Savitribai Phule Pune University

WORKBOOK

CS 369 Object Oriented Programming using Java - II

T. Y. B. Sc. (Computer Science)

SEMESTER VI

(From Academic Year 2021)

Student Name:			
College:			
Roll No:		Exam Seat No:	
Year:		Division:	

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

BOARD OF STUDIES

1. Dr. Bedekar Smita
2. Dr. Dhole Sanjay
3. Dr. Bharambe Manisha
4. Dr. Ponde Poonam
5. Dr. Sardesai Anjali
6. Dr. Mulay Prashant
7. Dr. Sayyad Razzak
8. Dr. Wani Vilas
9. Dr. Shinde Sahebrao
10. Dr. Kolhe Satish
11. Dr. Patil Ranjeet
12. Dr. Sonar Deepak
13. Dr. Yadav Jyoti
14. Dr. Kumbhojkar Nilesh
15. Dr. Dasari Abhay

Co-ordinators

➢ **Dr. Prashant Mulay**, Annasaheb Magar College, Hadapsar , Pune.
   Member, BOS Computer Science, Savitribai Phule Pune University

➢ **Dr. Manisha Bharambe** , MES Abasaheb Garware college, Pune.
   Member, BOS Computer Science, Savitribai Phule Pune University

Editor

Dr. Manisha Bharambe , MES Abasaheb Garware college, Pune.
   Member, BOS Computer Science, Savitribai Phule Pune University

Prepared by:

Ms. Gadekar Manisha J.	Annasaheb Magar College, Hadapsar, Pune.
Mrs. Kulkarni Roopali G.	Ahmednagar College , Ahmednagar

Table of Contents

Introduction

About the workbook

This workbook is intended to be used by T. Y. B. Sc (Computer Science) students for the Laboratory Course – III CS – 369 based on Programming in JAVA CS- 365. Semester – VI.

The objectives of this book are

- Defining clearly the scope of the course

- Bringing uniformity in the way the course is conducted across different colleges

- Continuous assessment of the Students.

- Bring variation and variety in experiments carried out by different students in a batch

- Providing ready reference for students while working in the lab

- Catering to the need of slow paced as well as fast paced learners

How to use this workbook

The Object Oriented Programming using Java, practical syllabus is divided into five assignments. Each assignment has problems divided into three sets A, B and C.

- Set A is used for implementing the basic algorithms or implementing data structure along with its basic operations. **Set A is mandatory.**

- Set B is used to demonstrate small variations on the implementations carried out in set A to improve its applicability. Depending on the time availability the students should be encouraged to complete set B.

- Set C prepares the students for the viva in the subject. Students should spend additional time either at home or in the Lab and solve these problems so that they get a deeper understanding of the subject.

Instructions to the students

Please read the following instructions carefully and follow them.

- Students are expected to carry workbook during every practical.

- Students should prepare oneself before hand for the Assignment by reading the relevant material.

- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However student should spend additional hours in Lab and at home to cover as many problems as possible given in this work book.

- Students will be assessed for each exercise on a scale from 0 to 5
  - ➢ Not done   0
  - ➢ Incomplete   1
  - ➢ Late Complete   2
  - ➢ Needs improvement   3
  - ➢ Complete   4
  - ➢ Well Done   5

Instruction to the Practical In-Charge

- Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating the software.

- Choose appropriate problems to be solved by students. Set A is mandatory. Choose problems from set B depending on time availability. Discuss set C with students and encourage them to solve the problems by spending additional time in lab or at home.

- Make sure that students follow the instruction as given above.

- You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.

- The value should also be entered on assignment completion page of the respective Lab course.

Instructions to the Lab administrator and Exam guidelines

- You have to ensure appropriate hardware and software is made available to each student.

- Do not provide Internet facility in Computer Lab while examination

- Do not provide pen drive facility in Computer Lab while examination.

The operating system and software requirements are as given below:

- Operating system: Linux
- Editor: Any linux based editor like vi, gedit , eclipse etc.
- Database : Postgresql
- Tomcat , Spring Tool Suite.
- Compiler: javac

Assignment Completion Sheet

Sr. No	Assignment Name	Marks (Out of 5)	Signature
1	Collections		
2	Multithreading		
3	Database Programming		
4	Servlets and JSP		
5	Spring		
Total out of 25			
Total out of 5 (Viva)			
Total out of 30			
Total (Out of 15)			

This is to certify that Mr/Ms _____

University Exam Seat Number _____ has successfully completed the course work

for Lab Course III and has scored _____ Marks out of 15.

Instructor   **Head**

Internal Examiner   **External Examiner**

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

CS – 369

Object Oriented Programming Using Java -II

Assignment 1: Collections

Objectives

- Study the Collections framework in java
- To store and manipulate group of objects
- Use various collections

Reading

You should read the following topics before starting this exercise:
- Concept of Collection
- Classes and interfaces in the Collections framework
- Concept of iterator.
- Creating and using collections objects.

Ready Reference

What is Collection?

Collection is a group of objects.
Collection is a container object. It is used for storing multiple homogeneous and heterogeneous, unique and duplicate objects without size limitation.

What is the need of Collection?

In Java we can store and transfer the data in the following ways.
1. **Primitive data types**
   We can use primitive data types for storing only one element data and only one type of data.
2. **Class objects**
   Can store multiple fixed number of values of different type and can store multiple data elements of multiple types.
3. **Array Object**
   Can store multiple fixed number of values of same typefor storing many values of same data type.
4. **Collection Object**
   Can store multiple objects of same and different types without size limitations
   Thus, if our requirement is store and process multiple objects then we go for Collection Framework.

Limitations of Arrays -

1. Type(It is homogeneous in nature)
2. Size
3. Storing order
4. Operation Problem

Arrays vs Collections

Arrays	Collection
Fixed in size	Growable
Can hold only homogeneous data	Can hold both homogeneous & heterogeneous data
No predefined method support	For every requirement methods are available
Arrays can hold both primitives and objects	Collections can hold only objects

When to use collection?

Hence if our requirement is representing group individual objects as a single entity then the better option is 'Collection framework'.

Collection Framework

- The Collection Framework in Java is a collection of interfaces and classes to store, process and transfer the data efficiently.
- Collections are growable in nature. i.e., based on run time requirement we can store any number of elements.
- Collections can hold both homogeneous and heterogeneous data elements.
- We can transfer the data from one method to another of any type and any number of elements.
- For every requirement ready made method support is available. Hence being a programmer, we just have to know how to use the predefined methods
- A collection framework provides built-in interfaces, classes and methods which we can use to directly create and use a collection instead of writing long code manually.
- A collections framework is a unified architecture for representing and manipulating collections.
- All collections frameworks contain the following

Interfaces : These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.

Implementations : These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.

Algorithms : These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces. The algorithms are said to be

polymorphic: that is, the same method can be used on many different implementations of the appropriate collection interface.

In addition to collections, the framework defines several map interfaces and classes. Maps store key/value pairs. Although maps are not collections in the proper use of the term, but they are fully integrated with collections.

Collection's framework in Java supports **two types of containers**:

One for storing a collection of elements (objects), that is simply called a collection.
The other, for storing key/value pairs, which is called a map.

Collection Hierarchy in Java
The hierarchy of the entire collection framework consists of four core interfaces such as Collection, List, Set, Map, and two specialized interfaces named SortedSet and SortedMap for sorting.

All the interfaces and classes for the collection framework are located in java.util package. The diagram of Java collection hierarchy is shown in the below figure**.**
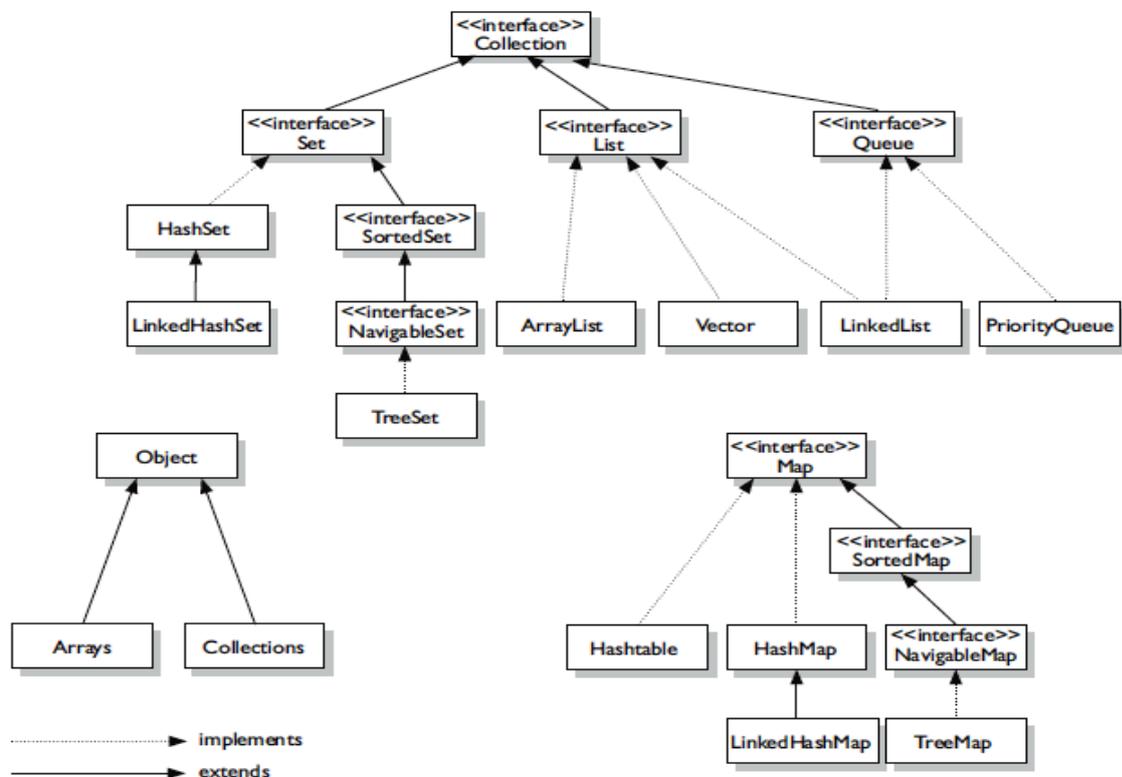
Interfaces
Collection(I)
      It is at the top of collection hierarchy and must be implemented by any class

that defines a collection. Following are some of the commonly used methods in this interface.

Methods	Description
boolean add(Object e)	Used to add objects to a collection. Returns true if obj was added to the collection. Returns false if obj is already a member of the collection, or if the collection does not allow duplicates.
boolean addAll(Collection C)	Add all elements of collection C to the invoking collection. Returns true if the element were added. Otherwise, returns false.
boolean remove(Object obj)	To remove an object from collection. Returns true if the element was removed. Otherwise, returns false.
boolean removeAll(Collection C)	Removes all element of collection C from the invoking collection. Returns true if the collection's elements were removed. Otherwise, returns false.
boolean contains(Object obj)	To determine whether an object is present in collection or not. Returns true if obj is an element of the invoking collection. Otherwise, returns false.
booleanisEmpty()	Returns true if collection is empty, else returns false.
int size()	Returns number of elements present in collection.
void clear()	Removes total number of elements from the collection.
Object[] toArray()	Returns an array which consists of the invoking collection elements.
boolean retainAll(Collection c)	Deletes all the elements of invoking collection except the specified collection.
Iterator iterator( )	Returns an iterator for the invoking collection.
boolean equals(Object obj)	Returns true if the invoking collection and obj are equal. Otherwise, returns false.
Object[] toArray(Object array[])	Returns an array containing only those collection elements whose type matches of the specified array.

The List Interface

- List(I) is a child interface of Collection.
- Here duplicates are allowed and insertion order is preserved
- Implemented class of List are
  ArrayList, LinkList, Vector (legacy classes)

Apart from methods of Collection Interface, it adds following methods of its own

Methods	Description
Object get(int index)	Returns object stored at the specified index
Object set(int index, E obj)	Stores object at the specified index in the calling

	collection
int indexOf(Object obj)	Returns index of first occurrence of obj in the collection
int lastIndexOf(Object obj)	Returns index of last occurrence of obj in the collection
List subList(int start, int end)	Returns a list containing elements between start and end index in the collection

The Set Interface

- Set(I) is a child interface of Collection.
- This interface defines a Set.
- It extends Collection interface and doesn't allow insertion of duplicate elements.
- Implemented class of Set are HashSet, LinkedHashSet, TreeSet.
- Duplicates are not allowed and insertion order is not preserved.
- It doesn't define any method of its own.
- It has two sub interfaces, SortedSet and NavigableSet.

SortedSet Interface

- SortedSet interface extends Set interface and arranges added elements in an ascending order.
- SortedSet(I) is a child interface of Set.
- Implemented class of Set are TreeSet
- Duplicates are not allowed but all objects should be inserted according to some sorting order.

The Queue Interface

- It extends collection interface and defines behavior of queue, that is first-in,first- out.
- Queue(I) is a child interface of Collection.

Methods	Description
Object poll()	removes element at the head of the queue and returns null if queue is empty
Object remove()	removes element at the head of the queue and throws NoSuchElementException if queue is empty
Object peek()	returns the element at the head of the queue without removing it. Returns null if queue is empty
Object element()	same as peek(), but throws NoSuchElementException if queue is empty
booleanoffer(E obj)	Adds object to queue.

Map Interface

- Not child interface of Collection.
- Map and Collection both are different
- A Map stores data in key and value pair.

SortedMap Interface

- It is a child interface of map.
- If we want to represent a group of key value pairs according to some sorting order of keys then should go for SortedMap

Java Collection Framework Classes
This table contains abstract and non-abstract classes that implements collection interface.

Class	Description
AbstractCollection	Implements most of the Collection interface.
AbstractList	Extends AbstractCollection and implements most of the List interface.
AbstractQueue	Extends AbstractCollection and implements parts of the Queue interface.
AbstractSequentialList	Extends AbstractList for use by a collection that uses sequential rather than random access of its elements.
LinkedList	Implements a linked list by extending AbstractSequentialList
ArrayList	Implements a dynamic array by extending AbstractList
ArrayDeque	Implements a dynamic double-ended queue by extending AbstractCollection and implementing the Deque interface(Added by Java SE 6).
AbstractSet	Extends AbstractCollection and implements most of the Set interface.
EnumSet	Extends AbstractSet for use with enum elements.
HashSet	Extends AbstractSet for use with a hash table.
LinkedHashSet	Extends HashSet to allow insertion-order iterations.
PriorityQueue	Extends AbstractQueue to support a priority-based queue.
TreeSet	Implements a set stored in a tree. Extends AbstractSet.

Implementations

The general-purpose implementations are summarized in the following table.

General – Purpose Implementation					
Interface			Implementation		
	Hash table	**Resizable Array**	**Tree**	**Linked List**	**Hash table + Linked List**
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

List Implementations

Lists are further classified into the following:

- ArrayList
- LinkedList
- Vectors

ArrayList class

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed. **.**

Important points to note

- Underlying data structure for ArrayList is Resizable Array.
- Duplicates are allowed.
- Insertion order is preserved.
- Heterogeneous objects insertion is allowed .
- 'Null' insertion is possible.
- ArrayList is best choice if our frequent operation is retrieval.
- ArrayList is worst choice if our frequent operation is insertion or deletion in middle

Methods of ArrayList

Method	Description
void add(int index, Object element)	It is used to insert the specified element at the specified position in a list.
boolean addAll(int index, Collection c)	It is used to append all the elements in the specified collection, starting at the specified position of the list.
void clear()	It is used to remove all of the elements from this list.
void ensureCapacity(int requiredCapacity)	It is used to enhance the capacity of an ArrayList instance.
Eget(int index)	It is used to fetch the element from the particular position of the list.

boolean isEmpty()	It returns true if the list is empty, otherwise false.
boolean contains(Object o)	It returns true if the list contains the specified element
boolean remove(Object o)	It is used to remove the first occurrence of the specified element.
boolean removeAll(Collection c)	It is used to remove all the elements from the list.
void replaceAll(Collection c)	It is used to replace all the elements from the list with the specified element.
void retainAll(Collection c)	It is used to retain all the elements in the list that are present in the specified collection.
int size()	It is used to return the number of elements present in the list.

Linked List

Java LinkedList class provides implementation of linked-list data structure.

Important points to note

- It uses doubly linked list to store the elements.
- Duplicates are allowed.
- Insertion order is preserved.
- Heterogeneous objects insertion allowed.
- 'Null' insertion is possible.
- .LinkList is best choice if our frequent operation is insertion or deletion in middle.
- LinkList is worst choice if our frequent operation is retrieval operation.

Methods of Java LinkedList

Method	Description
void add(int index, Object element)	It is used to insert the specified element at the specified position index in a list.
boolean addAll(Collection c)	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
void addFirst(Object element)	It is used to insert the given element at the beginning of a list.
void addLast(Object element)	It is used to append the given element to the end of a list.
void clear()	It is used to remove all the elements from a list.
boolean contains(Object o)	It is used to return true if a list contains a specified element.
Object element()	It is used to retrieve the first element of a list.
Object get(int index)	It is used to return the element at the specified position in a list.
Object peek()	It retrieves the first element of a list
void push(Object e)	It pushes an element onto the stack represented by a list.
Object remove()	It is used to retrieve and removes the first element of a list.
Object remove(int index)	It is used to remove the element at the specified position in a list.

| int size() | It is used to return the number of elements in a list. |

Vector
- Vector uses a dynamic array to store the data elements.
- It is similar to ArrayList.

Set Implementation

Sets are further classified into the following:
- **HashSet**
- **LinkedHashSet**
- **TreeSet.**

HashSet
- Underlying data structure for HashSet is hash table.
- Duplicates are not allowed
- Insertion order is not preserved
- Heterogeneous objects insertion is allowed
- 'Null' insertion is possible
- HashSet is best choice if our frequent operation is search

LinkedHashSet
- Underlying data structure for LinkedHashSetis hash table and Linked List.
- Duplicates are not allowed.
- Insertion order is not preserved.
- Heterogeneous objects insertion is allowed.
- 'Null' insertion is possible.
- LinkedHashSet is best choice to develop catche based application.

TreeSet
- Underlying data structure is balanced tree.
- Duplicates are not allowed
- Insertion order is not preserved but all objects will be inserted according to some
- sorting order
- Heterogeneous objects insertion is not allowed
- 'Null' insertion is possible but only once
- HashSet is best choice if our frequent operation is search

Map

A Map is useful if you have to search, update or delete elements on the basis of a key.

There are two interfaces for implementing Map in java: **Map and SortedMap**, and

Three classes: **HashMap, LinkedHashMap, and TreeMap.**

Class	Description
HashMap	HashMap is the implementation of Map, but it doesn't maintain any order.
LinkedHashMap	LinkedHashMap is the implementation of Map. It inherits HashMap class. It maintains insertion order.
TreeMap	TreeMap is the implementation of Map and SortedMap. It

	maintains ascending order.

HashMap

Map contains its own methods. collection terminology is not applicable

- Underlying datastructure forHashMapis hash table
- Duplicates keys are not allowed but values are allowed
- Insertion order is not preserved and it is based on hashcode of keys
- Heterogeneous keys and values are allowed
- 'Null' insertion is possible
- Best choice for searching

LinkedHashMap

- Underlying datastructure forLinkedHashMap is hash table and Linked List
- Duplicates keys are not allowed but values are allowed
- Insertion order is preserved and it is based on hashcode of keys
- Heterogeneous keys and values allowed
- 'Null' insertion is possible
- Best choice for searching

SortedMap

It is child interface of Map.

- Underlying data structure forSortedMap is hash table
- Duplicates keys are not allowed but values
- Insertion order is not preserved and it is based on hashcode of keys
- Heterogeneous keys and values allowed
- 'Null' insertion is possible
- Best choice for searching

TreeMap

- Underlying data structure is RED-BLACK TREE
- Duplicates keys are not allowed but values can be duplicated
- Insertion order is not preserved preserved and it is based on some sorting order of keys
- Heterogeneous keys and values not allowed
- Null acceptance is not there

Hash table

- Underlying data structure is hash table.
- Duplicates keys are not allowed but values can be duplicated.
- Insertion order is not preserved.
- Heterogeneous keys and values allowed.
- Null not allowed.
- Thread safe .
- Best choice for Searching.
- Default initial capacity is 11.

Cursors

A Java Cursor is an Iterator, which is used to iterate or traverse or retrieve a Collection or Stream object's elements one by one.

Java supports the following three different cursors.

- **Enumeration(I)**
- **Iterator(I)**
- **ListIterator(I)**

Enumeration(I)

We can use Enumeration to get objects one by one from the legacy collection objects.
We can create Enumeration object by using elements() method.
Enumeration interface defines the following two methods
public boolean hasMoreElements();
public Object nextElement();

Iterator

We can apply Iterator concept for any collection object hence it is universal cursor. By using this we can perform both read and remove operations.We can create Iterator by using Iterator() of collection interface.

        public Iterator iterator();
        Iterator itr = c.Iterator();

Where c is any collection object

Iterator Methods

Methods	Description
next()	Returns the next object
boolean hasNext()	This returns a true value if a high number of elements are encountered during iteration.
remove()	This method removes the current element. Throws *IllegalStateException* if an attempt is made to call remove( ) that is not preceded by a call to next( ).

ListIterator

By using this we can move either to f/w or b/w direction and hence it is bidirectional cursor. We can perform replacement and addition of new objects in addition to read and remove operation.
Note - It is most powerful cursor but its limitation is,it is applicable only for list implemented class objects and it is not universal cursor.

ListIterator methods

Methods	Description
void add(Object obj)	Inserts obj into the list in front of the element that will be returned by the next call to next( ).
boolean hasNext( )	Returns true if there is the next element. Otherwise, returns false.

boolean hasPrevious( )	Returns true if there is a previous element. Otherwise, returns false.
object next( )	Returns the next element. A NoSuchElementException is thrown if there is not the next element.
int nextIndex( )	Returns the index of the next element. If there is not the next element, returns the size of the list.
Object previous( )	Returns the previous element. A NoSuchElementException is thrown if there is not a previous element.
int previousIndex( )	Returns the index of the previous element. If there is not a previous element, returns -1.
void remove( )	Removes the current element from the list. An IllegalStateException is thrown if remove( ) is called before next( ) or previous( ) is invoked.
void set(Object obj)	Assigns obj to the current element. This is the element last returned by a call to either next( ) or previous( ).

Comparator

- Comparator interface is used to order the objects of a user-defined class.
- This interface is found in java.util package and contains 2 methods **compare(Object obj1,Object obj2)** and **equals(Object element).**
- It provides multiple sorting sequences, i.e., you can sort the elements on the basis of any data member, for example, rollno, name, age or anything else

Methods of Java Comparator Interface

There are two methods of Comparators in, namely:

Methods	Description
compare(Object obj1,Object obj 2)	Compares the first object with another
equals(Object obj)	Compares current object with specified obj

Self Activity

Note: To use any Collection class in your program, you need to import java.util package.

Whenever you print any Collection class, it gets printed inside the square brackets [] with its elements.

Sample Program1 : Program to demonstrate ArrayList

```
/* Program to demonstrate ArrayList*/

import java.util.ArrayList;
class ArrayListDemo
{
    public static void main(String[] args)
    {
        // creating an Array List named colors
```

```
ArrayList  Al = new ArrayList();
      // add elements in the Array List
Al.add("Red");
Al.add(5);
Al.add("Null");
Al.add("Orange");
Al.add("Red");
      // printing the ArrayList
System.out.println(Al);
   }
}
```

Sample Program2 : Program to demonstrate LinkedList

```
/* Program to demonstrate LinkedList  */

import java.util.*;
public class LinkedList1{
 public static void main(String args[]){
  LinkedList<String> al=new LinkedList<String>();
al.add("Ravi");
al.add("Vijay");
al.add("Ravi");
al.add("Ajay");
  Iterator<String>itr=al.iterator();
  while(itr.hasNext()){
System.out.println(itr.next());
 }
 }
}
```

Sample Program3 : Program to demonstrate HashSet

```
/* Program to demonstrate HashSet  */
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        // Create HashSet object

    HashSet hs = new HashSet(5, 0.5f);
    System.out.println(hs.add("one"));
    System.out.println(hs.add("two"));
    System.out.println(hs.add("three"));
    System.out.println(hs.add("four"));
    System.out.println(hs.add("five"));
        // Print out the HashSet object
    System.out.println(hs);
        // Add a duplicate item to the HashSet
```

```
        Boolean b = hs.add("one");
        System.out.println("Duplicate item allowed = " + b);
        System.out.println(hs);

    }
  }
```

Sample Program4 : Program to demonstrate LinkedHashSet

```
/* Program to demonstrate LinkedHashSet  */
import java.util.*;
public class Test4{
public static void main(String args[]){
LinkedHashSet<String> set=new LinkedHashSet<String>();
set.add("Java");
set.add("ML");
set.add("Python");
set.add("AI");
Iterator<String>itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

Sample Program5 :Program to demonstrate TreeSet

```
/* Program to demonstrate TreeSet*/

import java.util.Set;
import java.util.TreeSet;
public class Main {
    public static void main(String[] args) {
        Set ts = new TreeSet();
ts.add("one");
ts.add("two");
ts.add("three");
ts.add("four");
ts.add("three");
System.out.println("Members from TreeSet = " + ts);
        Set ts2 = new TreeSet();
ts2.add(1);
ts2.add(2);
ts2.add(3);
ts2.add(4);
ts2.add(2);
System.out.println("Members from TreeSet = " + ts2);
    }
  }
```

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

Sample Program6 :Program to demonstrate HashTable

```
/* Program to demonstrate HashTable*/

import java.util.*;
class Demo
{
 public static void main(String args[])  {
   // Creating Hashtable
Hashtable<String,Integer>hashtable = new Hashtable<String,Integer>();
   // Adding elements
hashtable.put("a",100);
hashtable.put("b",200);
hashtable.put("c",300);
hashtable.put("d",400);
   // Displaying Hashtable
System.out.println(hashtable);
   // Search for a value
booleanval = hashtable.contains(400);
System.out.println("is 400 present: "+val);
   // Search for a key
val = hashtable.containsKey("d");
System.out.println("is d present: "+val);
 }
}
```

Sample Program7 : Program to demonstrate Iterator

```
/* Program to demonstrate Iterator  */

import java.util.ArrayList;
import java.util.Iterator;
 public class Test
{
public static void main(String[] args)
  {
 ArrayList al = new ArrayList();
    for (int i = 0; i< 10; i++)
    al.add(i);
      System.out.println(al);
      // at beginning itr(cursor) will point to
     // index just before the first element in al
  Iterator itr = al.iterator();
      // checking the next element availabilty
     while (itr.hasNext())
     {
       //  moving cursor to next element
   int i = (Integer)itr.next();
         // getting even elements one by one
    System.out.print(i + " ");
```

```
      // Removing odd elements
      if (i % 2 != 0)
        itr.remove();
    }
    System.out.println();
    System.out.println(al);
  }
}
```

Sample Program8 : Program to demonstrate ListIterator

```
/* Program to demonstrate ListIterator  */

import java.util.ArrayList;
import java.util.ListIterator;
public class Main {
    public static void main(String[] args) {
     // Create ArrayList object with capacity of 2 elements
ArrayList al = new ArrayList(2);
System.out.println(al+", size = "+al.size());
     // Add items to the ArrayList
al.add("R");
al.add("U");
al.add("O");
al.add(new String("x"));
al.add(2, new Integer(10));
System.out.println(al+", size = " + al.size());
     // Remove item
al.remove("U");
System.out.println(al+", size = " + al.size());
     // Check if the list contains the specified element
     Boolean b = al.contains("x");
System.out.println("The list contains x = " + b);
     b = al.contains("p");
System.out.println("The list contains p = " + b);
     b = al.contains(new Integer(10));
System.out.println("The list contains Integer of 10 = " + b);
     // Create ListIterator and iterate entries in it
ListIterator li = al.listIterator();
     while (li.hasNext())
System.out.println("From ListIterator = " + li.next());

     // Create Object array from ArrayList
     Object a[] = al.toArray();
     for (int i=0; i<a.length; i++)
System.out.println("From an Array = " + a[i]);
  }
}
```

Lab Assignments

Set A

a) Write a java program to accept names of 'n' cities, insert same into array list collection and display the contents of same array list, also remove all these elements.

b) Write a java program to read 'n' names of your friends, store it into linked list, also display contents of the same.

c) Write a program to create a new tree set, add some colors (string) and print out the tree set.

d) Create the hash table that will maintain the mobile number and student name. Display the contact list.

Set B

a) Accept 'n' integers from the user. Store and display integers in sorted order having proper collection class. The collection should not accept duplicate elements.

b) Write a program to sort HashMap by keys and display the details before sorting and after sorting.

c) Write a program that loads names and phone numbers from a text file where the data is organized as one line per record and each field in a record are separated by a tab (\t).it takes a name or phone number as input and prints the corresponding other value from the hash table (hint: use hash tables)

Set C

a) Create a java application to store city names and their STD codes using an appropriate collection. The GUI should allow the following operations:
      i.   Add a new city and its code (No duplicates)
      ii.   Remove a city from the collection
      iii.   Search for a city name and display the code

b) Write a program to create link list of integer objects. Do the following:
      i.   add element at first position
      ii.   delete last element
      iii.   display the size of link list

c) Read a text file, specified by the first command line argument, into a list. The program should then display a menu which performs the following operations on the list:

1. Insert line   2. Delete line   3. Append line 4. Modify line 5. Exit
When the user selects Exit, save the contents of the list to the file and end the
program.

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-charge

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

Assignment 2: Multithreading

Objectives

- To create and use threads in java
- To demonstrate multithreading

Reading

You should read the following topics before starting this exercise:
- Thread class
- Runnable interface
- Thread lifecycle
- Thread methods

Ready Reference

Introduction:

A program can be divided into a number of small processes. Each small process can be addressed as a single thread (a lightweight process).

Multithreaded programs contain two or more threads that can run concurrently and each thread defines a separate path of execution. This means that a single program can perform two or more tasks simultaneously. For example, one thread is writing content on a file at the same time another thread is performing spelling check.

Why use Threads in Java
The Java run-time system depends on threads for many things. Threads reduce inefficiency by preventing the waste of CPU cycles.

Why Multithreading?
Thread has many advantages over the process to perform multitasking. Process is heavy weight, takes more memory and occupy CPU for longer time that may lead to performance issue with the system. To overcome this issue process is broken into small unit of independent sub-process. These sub-process are called threads that can perform independent task efficiently. So nowadays computer systems prefer to use thread over the process and use multithreading to perform multitasking.

The main thread
When we run any java program, the program begins to execute its code starting from the main method. Therefore, the JVM creates a thread to start executing the code present in main method. This thread is called as main thread. Although the main thread is automatically created, you can control it by obtaining a reference to it by calling currentThread() method. Two important things to know about main thread are,

- It is the thread from which other threads will be produced.
- It must be always the last thread to finish execution

Life cycle of a Thread

1) New
The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable
The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running
The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)
This is the state when the thread is still alive, but is currently not eligible to run.

5)Terminated
A thread is in terminated or dead state when its run() method exits.

How to Create a Java Thread
Java lets you create thread in following two ways:-

 1.By implementing the Runnable interface.
 2.By extending the Thread

Method 1: Thread creation by extending Thread class

```
class MultithreadingDemo extends Thread
  {
   public void run()
    {
     System.out.println("My thread is in running state.");
    }
   public static void main(String args[])
   {
     MultithreadingDemo obj=new MultithreadingDemo();
     obj.start();
   }
  }
```

  Output:

  My thread is in running state.

Method 2: Thread creation by implementing Runnable Interface

```
class MultithreadingDemo implements Runnable
  {
   public void run()
   {
    System.out.println("My thread is in running state.");
   }
```

```
    public static void main(String args[])
    {
      MultithreadingDemo obj=new MultithreadingDemo();
      Thread tobj =new Thread(obj);
      tobj.start();
    }
  }
```

Output:

My thread is in running state.

Important methods of Thread class

Method	Description
setName()	To give thread a name
getName()	Return thread's name
getPriority()	Return thread's priority
isAlive()	Checks if thread is still running or not
join()	Wait for a thread to end
run()	Entry point for a thread
sleep()	Suspend thread for a specified time
start()	Start a thread by calling run() method
activeCount()	Returns an estimate of the number of active threads in the current thread's thread group and its subgroups.
checkAccess()	Determines if the currently running thread has permission to modify this thread.
currentThread()	Returns a reference to the currently executing thread object.
getId()	Returns the identifier of this Thread.
getState()	Returns the state of this thread.
interrupt()	Interrupts this thread.
isAlive()	Tests if this thread is alive.
isDaemon()	Tests if this thread is a daemon thread.
isInterrupted()	Tests whether this thread has been interrupted.
setPriority(int newPriority)	Changes the priority of this thread.
yield()	A hint to the scheduler that the current thread is willing to yield its current use of a processor.

Priority of a Thread

Each thread have a priority. Priorities are represented by a number between 1 and 10.

Three constants defined in Thread class:
public static int **MIN_PRIORITY**
public static int **NORM_PRIORITY**
public static int **MAX_PRIORITY**

Default priority of a thread is **5 (NORM_PRIORITY)**. The value of
MIN_PRIORITY is **1** and the value of **MAX_PRIORITY is 10**.

Thread Synchronization

- If multiple threads are trying to operate simultaneously on same java object, then there may be a chance of data inconsistency problem. To overcome this problem, we should go for Synchronization
- If a method or block declared as synchronized then at a time only one thread is allowed to execute that method or block on the given object so that data inconsistency prob will be resolved.

Inter Thread Communication in Java

- Inter-thread communication in Java is a technique through which multiple threads communicate with each other
- There are several situations where communication between threads is important. For example, suppose that there are two threads A and B. Thread B uses data produced by Thread A and performs its task.
- If Thread B waits for Thread A to produce data, it will waste many CPU cycles. But if threads A and B communicate with each other when they have completed their tasks, they do not have to wait and check each other's status every time.
- Inter thread communication in Java can be achieved by using three methods
  1. **wait**()
  2. notify()
  3. notifyAll()
- Note-These methods can be called only from within a synchronized method or synchronized block of code

Self - Activity

Execute all the sample programs

Sample Program1:Below is a program that illustrates instantiation and running of threads using the Runnable interface.

```
class RunnableThread implements Runnable
  {
   Thread runner;
   public RunnableThread()
    {
    }
public RunnableThread(String threadName)
{
    runner = new Thread(this, threadName); // Create a new thread.
    System.out.println(runner.getName());
    runner.start(); // Start the thread.
}
```

```
public void run()
 {
//Display info about this particular thread
        System.out.println(Thread.currentThread());
  }
}
public class RunnableExample
{
public static void main(String[] args)
{
    Thread thread1 = new Thread(new RunnableThread(), "thread1");
    Thread thread2 = new Thread(new RunnableThread(), "thread2");
    RunnableThread thread3 = new RunnableThread("thread3");
     //Start the threads
        thread1.start();
        thread2.start();
  try
  {
    //delay for one second
      Thread.currentThread().sleep(1000);
  }
  catch (InterruptedException e)
  {
  }
//Display info about the main thread
System.out.println(Thread.currentThread());
}
}
```

Sample Program2:Creating multiple threads using the Thread class.

```
class MyThread extends Thread
{
String message;
MyThread(String message)
{
this.message = message;
}
public void run()
{
try
{
for(int i=1; i<=5; i++)
```

```
{
System.out.println(message + "-" + i);
Thread.sleep(5000); //sleep for 5 seconds
}
}
catch(InterruptedException ie) { }
}
}
public class MultipleThreadDemo
{
public static void main( String[] args)
{
MyThread t1 = new MyThread("One");
MyThread t2 = new MyThread("Two");
System.out.println(t1);
System.out.println(t2); t1.start();
t2.start();
}
}
```

Sample Program3:Demonstrating Priority of a Thread

```
class PriorityDemo extends Thread
 {
 public void run()
  {
  System.out.println("running thread name is:"+Thread.currentThread().getName());
  System.out.println("running thread priority is:"+Thread.currentThread().getPriority());
  }
 public static void main(String args[])
{
    PriorityDemo m1=new PriorityDemo();
    PriorityDemo m2=new PriorityDemo();
    m1.setPriority(Thread.MIN_PRIORITY);
    m2.setPriority(Thread.MAX_PRIORITY);
    m1.start();
    m2.start();
 }
}
```

```
class First
{
  synchronized public void display(String msg)
  {
   System.out.print ("["+msg);
   try
   {
    Thread.sleep(1000);
   }
   catch(InterruptedException e)
   {
    e.printStackTrace();
   }
   System.out.println ("]");
  }
}
class Second extends Thread
{
  String msg;
  First fobj;
  Second (First fp,String str)
  {
   fobj = fp;
   msg = str;
   start();
  }
  public void run()
  {
   fobj.display(msg);
  }
}
public class MyThread
{
  public static void main (String[] args)
  {
   First fnew = new First();
   Second ss = new Second(fnew, "welcome");
   Second ss1= new Second(fnew,"new");
   Second ss2 = new Second(fnew, "programmer");
  }
}
```

Sample Program4 : Demonstrating Synchronization of a Thread

```
class mythread extends Thread
{
    String msg[]={"Java", "Supports", "Multithreading", "Concept"};
    mythread(String name)
      {
        super(name);
      }
    public void run()
    {
        display(getName());
        System.out.println("Exit from "+getName());
    }
    synchronized void display(String name ) //Synchrinized method
    {
        for(int i=0;i<msg.length;i++)
        {
            System.out.println(name+msg[i]);
        }
    }
    } /* Main class */
class MySynchro
{
    public static void main(String args[])
    {
        mythread t1=new mythread("Thread 1: ");
        mythread t2=new mythread("Thread 2: ");
        t1.start();
        t2.start();
        System.out.println("Main thread exited");
    }
}
```

Sample Program5 : Demonstrating Inter-thread communication of a Thread

```
class SampleThread extends Thread
  {
        int tBal = 0;
        public void run()
        {
          synchronized (this)
              {
```

```
            System.out.println("Thread calculation for total balance");
                for (int i = 0; i <= 30; i++)
                {
                        tBal = tBal + i;
                }
                System.out.println("Thread gives notification call");
                this.notify();
            }
        }
}
public class DemoThread
{
        public static void main(String[] args) throws InterruptedException
        {

                SampleThread st = new SampleThread ();
                st.start();
                synchronized (st)
                {
                        System.out.println("Thread calling wait() Method");
                        st.wait();
                        System.out.println("Thread got notification");
                        System.out.println("Totol Balance " + st.tBal);
                }
        }
}
```

Lab Assignments

Set A

a) Program to define a thread for printing text on output screen for 'n' number of times. Create 3 threads and run them. Pass the text 'n' parameters to the thread constructor. Example:
   i.   First thread prints "COVID19" 10 times.
   ii.   Second thread prints "LOCKDOWN2020" 20 times
   iii. Third thread prints "VACCINATED2021" 30 times

b) Write a program in which thread sleep for 6 sec in the loop in reverse order from 100 to 1 and change the name of thread.

c) Write a program to solve producer consumer problem in which a producer produces a value and consumer consume the value before producer generate the next value. (Hint: use thread synchronization)

Set B

a) Write a program to calculate the sum and average of an array of 1000 integers (generated randomly) using 10 threads. Each thread calculates the sum of 100 integers. Use these values to calculate average. [Use join method].

b) Write a program for a simple search engine. Accept a string to be searched. Search for the string in all text files in the current folder. Use a separate thread for each file. The result should display the filename, line number where the string is found.

c) Write a program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

Set C

a) Write a program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with "stop" or "ready" or "go"should appear above the buttons in a selected color. Initially there is no message shown.

b) Write a program to create a thread for moving a ball inside a panel vertically. The ball should be created when the user clicks on the start button.

c) Using the concepts of thread synchronization create two threads as sender and receiver. Sender thread will set a message to the receiver thread that will display the message on console. The sender thread accepts the input message from console. Continue this process until sender sets the message as "Good Bye Corona".

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Needs Improvement		4: Complete		5: Well, Done	

Practical In-charge

Assignment 3: Database Programming

Objectives

- To communicate with a database using java.
- To execute queries on tables.
- To obtain information about the database and tables

Reading

You should read the following topics before starting this exercise:

- The JDBC driver types
- The design of JDBC
- Statement, PreparedStatement, ResultSet
- DatabaseMetaData and ResultSetMetaData

Ready Reference

JDBC : Java Database Connectivity

This API contains of a set of **classes** and **interfaces** to enable programmers to communicate with a database using java. These classes and interfaces are in the java.sql package.

The JDBC API makes it possible to do three things:
      i.   Establish a connection with a data source.
      ii.   Send queries and update statements to the data source.
      iii.   Process the results.

The classes and interfaces in the java.sql package are given below.

Interface Name	Description
Connection	Represents a connection session with the database
DatabaseMetaData	Information about the database
Driver	Interface that every driver class must implement
ParameterMetaData	Information about parameters in PreparedStatement object
PreparedStatement	Represents precompiled SQL statement
Statement	For executing a static SQL statement and returning the results it produces.
CallableStatement	To execute SQL stored procedures.
Ref	Maps to SQL REF type
ResultSet	Table of data generated by executing a database query
ResultSetMetaData	Information about columns in a ResultSet
Savepoint	The representation of a savepoint, which is a point within the current transaction.
Array	Maps to the SQL type ARRAY
Blob	Represents SQL BLOB Value
Clob	Represents SQL CLOB type
SQLData	For custom mapping of an SQL user-defined type (UDT) to a class in

	the Java programming language.
SQLInput	An input stream that contains a stream of values representing an instance of an SQL structured type.
SQLOutput	The output stream for writing the attributes of a user-defined type back to the database.
Struct	Maps to an SQL structured type

Classes Name	Description
DriverManager	The basic service for managing a set of JDBC drivers.
DriverPropertyInfo	Driver properties for making a connection
Date	Represents an SQL DATE value.
SQLPermission	The permission for which the SecurityManager will check when code that is running in an applet calls the DriverManager.setLogWriter method or the DriverManager.setLogStream (deprecated) method.
Time	Represents an SQL TIME value.
Timestamp	Represents an SQL TIMESTAMP value.
Types	Defines constants that are used to identify generic SQL types, called JDBC types.

JDBC Drivers

To communicate with a database, you need a database driver.
There are four types of drivers :
       1. Type 1: JDBC-ODBC Bridge driver.
       2. Type 2: Native-API partly-Java driver.
       3. Type 3: JDBC-Net pure Java driver.
       4. Type 4: Native-protocol pure Java driver.

Load Driver

For postgresql, use the driver :
        org.postgresql.Driver

To load the driver, use the following command :
Syntax :
       Class.forName("DiverName");

Example :
       Class.forName("org.postgresql.Driver");

Establishing a connection

To establish a connection with the database, use the getConnection method of the DriverManager class.

This method returns a Connection object.

       DriverManager.getConnection("url", "user", "password");

Example :

Connection conn = DriverManager.getConnection
("jdbc:postgresql:TestDB", "postgres", "");

Methods of Connection class:

Methods	Description
void close()	Releases this Connection object's database and JDBC resources immediately instead of waiting for them to be automatically released.
void commit()	Makes all changes made since the previous commit/rollback permanent and releases any database locks currently held by this Connection object.
Statement createStatement()	Creates a Statement object for sending SQL statements to the database.
Statement createStatement(int resultSetType, int resultSetConcurr ency)	Creates a Statement object that will generate ResultSet objects with the given type and concurrency.
Boolean getAutoCommit()	Retrieves the current auto-commit mode for this Connection object.
DatabaseMetaData getMetaData()	Retrieves a DatabaseMetaData object that contains metadata about the database to which this Connection object represents a connection.
CallableStatement prepareCall(String s ql)	Creates a CallableStatement object for calling database stored procedures.
CallableStatement prepareCall(String s ql, int resultSetType, int resultSetConcurr ency)	Creates a CallableStatement object that will generate ResultSet objects with the given type and concurrency.
PreparedStatement prepareStatement(Str ing sql)	Creates a PreparedStatement object for sending parameterized SQL statements to the database.
PreparedStatement prepareStatement(Str ing sql, int resultSetType, int resultSetConcurr ency)	Creates a PreparedStatement object that will generate ResultSet objects with the given type and concurrency.
void rollback()	Undoes all changes made in the current transaction and releases any database locks currently held by this Connection object.
void setAutoCommit(Boolean autoCommit)	Sets this connection's auto-commit mode to the given state.

Executing Queries

To execute an SQL query, you have to use one of the following classes :

- Statement
- PreparedStatement
- CallableStatement

A Statement represents a general SQL statement without parameters. The method **createStatement()** creates a Statement object.

A PreparedStatement represents a precompiled SQL statement, with or without parameters. The method **prepareStatement(String sql)** creates a PreparedStatement object.

CallableStatement objects are used to execute SQL stored procedures. The method **prepareCall(String sql)** creates a CallableStatement object.

Executing a SQL statement with the Statement object, and returning a jdbc resultSet.

To execute a query, call an execute method from Statement such as the following :

- execute : Use this method if the query could return one or more ResultSet objects.
- executeQuery : Returns one ResultSet object.
- executeUpdate : Returns an integer representing the number of rows affected by the SQL statement. Use this method if you are using INSERT, DELETE, or UPDATE SQL statements.

Examples

 ResultSet rs = stmt.executeQuery("SELECT * FROM book");

 int result = stmt.executeUpdate("Update authors SET name = 'xxx' WHERE id = 1");

boolean bol = stmt.execute("DROP TABLE IF EXISTS DBTest");

ResultSet provides access to a table of data generated by executing a Statement.
The table rows are retrieved in sequence.
A ResultSet maintains a cursor pointing to its current row of data.
The next() method is used to successively step through the rows of the tabular results.

Examples :

```
Statement stmt =  conn.prepareStatement();
ResultSet rs = stmt.executeQuery("Select * from student");
while(rs.next())
 {
   //access resultset data
 }
```

To access these values, there are getXXX() methods where XXX is a type for example, getString(), getInt() etc.

There are two forms of the getXXX methods:
i.   Using columnName: getXXX(String columnName)
ii.   Using columnNumber: getXXX(int columnNumber)

Example
        rs.getString("studname")); rs.getString(1);
                        //where name appears as column 1 in the ResultSet

Using PreparedStatement

These are precompiled sql statements. For parameters, the SQL commands in a PreparedStatement can contain **placeholders** which are represented by **'?'** in the SQL command.

Example
                String sql = "UPDATE authors SET name = ? WHERE id = ?";
                PreparedStatement ps = conn.prepareStatement(sql);

Before the sql statement is executed, the placeholders have to be replaced by actual values.
This is done by calling a **setXXX(int n, XXX x)** method,
      where XXX is the appropriate type for the parameter
For example, **setString, setInt, setFloat, setDate** etc,
n is the placeholder number and x is the value which replaces the placeholder.

Example
                String sql = "UPDATE authors SET name = ? WHERE id = ?";
                PreparedStatement ps = conn.prepareStatement(sql);
                ps.setString(1,'abc');
                                //assign 'abc' to first placeholder
                ps.setInt(2,123);
                                //assign '123' to second placeholder

ResultSet Scroll Types and Concurrency
The scroll type indicates how the cursor moves in the ResultSet. The concurrency type affects concurrent access to the resultset. The types are given in the table below.

Scroll Type	
TYPE_FORWARD_ONLY	The result set is not scrollable.
TYPE_SCROLL_INSENSITIVE	The result set is scrollable but not sensitive to database changes.
TYPE_SCROLL_SENSITIVE	The result set is scrollable and sensitive to database changes.
Concurrency Type	
CONCUR_READ_ONLY	The result set cannot be used to update the database.
CONCUR_UPDATABLE	The result set can be used to update the database.

Example

Statement stmt = conn.createStatement
(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);

ResultSet Interface The ResultSet interface provides methods for retrieving and manipulating the results of executed queries.

Methods	Description
beforeFirst()	Default position. Puts cursor before 1st row of ResultSet.
first()	Puts cursor on 1st row of ResultSet.
last()	Puts cursor on last row of ResultSet.
afterLast()	Puts cursor after/beyond last row of ResultSet.
absolute (int pos)	Puts cursor at row number position where absolute (1) is a 1st row and absolute (-1) is last row of ResultSet.
relative (int pos)	Puts cursor at row no. position relative from current position.
next()	To move to the next row in ResultSet
previous()	To move to the previous row in ResultSet.
void close()	To close the ResultSet.
deleteRow()	Deletes the current row from the ResultSet and underlying database.
getRow()	Retrieves the current row number.
insertRow()	Inserts the contents of the insert row into the ResultSet object and into the database.
refreshRow()	Refreshes the current row with its most recent value in the database.
updateRow()	Updates the underlying database with the new contents of the current row of this ResultSet object.
getXXX(String columnName)	Retrieves the value of the designated column in the current row as a corresponding type in the Java programming language. XXX represents a type: Int, String, Float, Short, Long, Time etc.
moveToInsertRow()	Moves the cursor to the insert row.
close()	Disposes the ResultSet.
isFirst()	Tests whether the cursor is at the first position.
isLast()	Tests whether the cursor is at the last position.
isBeforeFirst()	Tests whether the cursor is before the first position.
isAfterLast()	Tests whether the cursor is after the last position.
updateXXX(int columnNumber, XXX value)	Updates the value of the designated column in the current row as a corresponding type in the Java programming language. XXX represents a type: Int, String, Float, Short, Long, Time etc.

DatabaseMetaData

This interface provides methods that tell you about the database for a given connection object.

Methods	Description
getDatabaseProductName()	Retrieves the name of this database product.
getDatabaseProductVersion()	Retrieves the version number of this database product.
getDriverName()	Retrieves the name of this JDBC driver.

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

getDriverVersion()	Retrieves the version number of this JDBC driver as a String.
getUserName()	Retrieves the user name as known to this database.
getCatalogs()	Retrieves the catalog names available in this database.
getSchemas(String catalog, String schemaPattern)	Retrieves the schema names available in this database.
getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	Retrieves a description of the tables available in the given catalog.
getPrimaryKeys(String catalog, String schema, String table)	Retrieves a description of the given table's primary key columns.
getExportedKeys(String catalog, String schema, String table)	Retrieves a description of the foreign key columns that reference the given table's primary key columns (the foreign keys exported by a table).
getImportedKeys(String catalog, String schema, String table)	Retrieves a description of the primary key columns that are referenced by a table's foreign key columns (the primary keys imported by a table).
getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	Retrieves a description of table columns available in the specified catalog.
getProcedures(String catalog, String schemaPattern, String procedureNamePattern)	Retrieves a description of the stored procedures available in the given catalog.
getFunctions(String catalog, String schemaPattern, String functionNamePattern)	Retrieves a description of the system and user functions available in the given catalog.

Example

```
DatabaseMetaData dbmd = conn.getMetaData();
ResultSet rs = dbmd.getTables(null, null, null,new String[] {"TABLE"});
while (rs.next())
    System.out.println( rs.getString("TABLE_NAME"));
```

ResultSetMetaData

The ResultSetMetaData interface provides information about the structure of a particular ResultSet.

Methods	Description
getColumnCount()	Returns the number of columns in the current ResultSet object.

getColumnDisplaySize(int column)	Gives the maximum width of the column specified by the index parameter.
getColumnLabel(int column)	Gives the suggested title for the column for use in display and printouts.
getColumnName(int column)	Gives the column name associated with the column index.
getColumnTypeName(int column)	Gives the designated column's SQL type. isReadOnly(int column) Indicates whether the designated column is read-only.
isWritable(int column)	Indicates whether you can write to the designated column.
isNullable(int column)	Indicates the nullability of values in the designated column.

Example

```
ResultSet rs = stmt.executeQuery(query);
ResultSetMetaData rsmd = rs.getMetaData();
int noOfColumns = rsmd.getColumnCount();
    System.out.println("Number of columns = " + noOfColumns);
for(int i=1; i<=noOfColumns; i++)
  {
        System.out.println("Column No : " + i);
        System.out.println("Column Name : " + rsmd.getColumnName(i));
        System.out.println("Column Type : " + rsmd.getColumnTypeName(i));
         System.out.println("Column display size : " +
                            rsmd.getColumnDisplaySize(i));
  }
```

Self - Activity

Execute all the sample programs

Sample Program1 : Sample program to display employee data (empid, empname, empsalary)

```
import java.sql.*;
import java.io.*;
class JDBCDemo
  {
    public static void main(String[] args) throws SQLException
    {
        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;
        try
        {
            Class.forName("org.postgresql.Driver");
            con = DriverManager.getConnection("jdbc:postgresql:empDB","postgres","");
            if(conn==null)
              System.out.println("Connection failed ");
            else
```

```
                {
                    System.out.println("Connection successful..");
                    stmt = conn.createStatement();
                    rs = stmt.executeQuery("Select * from emp");
                    while(rs.next())
                     {
                        System.out.print("EmpID = " + rs.getInt(1));
                        System.out.println("EmpName = " + rs.getString(2));
                        System.out.println("Salary = " + rs.getInt(3));
                     }
                  con.close();
                }
         }
     catch(Exception e)
     {
            System.out.println("ERROR"+e);
     }
 }//end of main
}// end of class
```

Sample Program2 : To perform insert and delete operations on employee table using PreparedStatement (Empid, Empname, Empsalary)

```
import java.sql.*;
import java.io.*;
 class JDBCDemoOp
  {
     public static void main(String[] args) throws SQLException
      {
         Connection con = null;
         Statement st =  null;
         ResultSet rs = null;
         PreparedStatement ps1 = null, ps2=null;
         int eid,esal;
         String ename;
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         Class.forName("org.postgresql.Driver");
         con = DriverManager.getConnection("jdbc:postgresql:EmpDB","postgres","");
         st = conn.createStatement();
         ps1 = con.prepareStatement("Insert into employee values(?,?,?)");
         ps2 = con.prepareStatement("Delete employee where ID = ?");

           if(con!=null)
              System.out.println("Connection successful..");
              System.out.println("Enter the employee ID, employee name and employee salary
                             to be inserted ");
           eid = Integer.parseInt(br.readLine());
           ename = br.readLine();
           esal = Integer.parseInt(br.readLine());
           ps1.setInt(1,eid);
           ps1.setString(2,ename);
```

```
            ps1.setInt(3,esal);
            ps1.executeUpdate();

            System.out.println("Enter the employee ID to be deleted ");
            eid = Integer.parseInt(br.readLine());
            ps2.setInt(1,eid);
            ps2.executeUpdate();
            conn.close();
      }//end of main
 }// end of class
```

Lab Assignments

Set A

    a) Create a PROJECT table with fields project_id, Project_name, Project_description, Project_Status. etc. Insert values in the table. Display all the details of the PROJECT table in a tabular format on the screen.(using swing).

    b) Write a program to display information about the database and list all the tables in the database. (Use DatabaseMetaData).

    c) Write a program to display information about all columns in the DONAR table using ResultSetMetaData.

Set B

    a) Create a MOBILE table with fields Model_Number, Model_Name, Model_Color, Sim_Type, NetworkType, BatteryCapacity, InternalStorage, RAM and ProcessorType. Insert values in the table. Write a menu driven program to pass the input using Command line argument to perform the following operations on MOBILE table.
        1. Insert 2. Modify 3. Delete 4. Search 5. View All   6. Exit

    b) Design a following Registration form and raise an appropriate exception if invalid information is entered like Birth Year '0000'

```
                    Co-WIN Registration
AdharCard No. :  [                    ]

Birth Year : [            ]

Mobile No. : [          ]

Age Group :   ○ 18 & above  ○ 45 & above

Select Hospital : [          ]

Vaccines :   ● Covishield,  ○ Covaxin  ○ Sputnik V.

Time Slot :  ● Morning ○ Afternoon ○ Evening

[ ADD ]  [ UPDATE ]  [ DELETE ]  [ VIEW ]  [ SEARCH ]
```

Set C

a) Create tables : Course (courseid, coursename, courseinstructor) and Student (studentid, studentname, studentclass). Course and Student have a many to many relationship. Create a GUI based system for performing the following operations on the tables:

    Course : Add Course, View All students of a specific course

    Student : Add Student, Delete Student, View All students, Search student.

b) Create the following tables and relations, for an INVESTMENT firm EMP(empid ,empname, empaddress, empcontact, empage) INVESTOR(invno, invname , invdate, invamt) An employee may invest in one or more investments, hence he can be an investor. But an investor need not be an employee of the firm. Insert sufficient number of records in the relations / tables with appropriate values.

    i. Display the List the distinct names of person who are either employees, or investors or both.

    ii. List the names of employees who are not investors

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-charge

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

Assignment 4: Servlets and JSP

Objectives

- To understand server-side programming.
- Defining and executing servlets.
- To demonstrate the use of JSP

Reading

You should read the following topics before starting this exercise:

- Concept of servlet.
- Introduction to Servlet (HTTP Servlet).
- Lifecycle of a Servlet and JSP.
- Handling Get and Post requests (HTTP).
- Data Handling using Servlet.
- Creating Cookies.
- Session Tracking using HTTP Servlet.
- JSP Directives.
- Scripting elements.
- Actions in JSP.

Ready Reference

What are servlets?
Servlets are small programs that execute on the server side. Servlets are pieces of Java source code that add functionality to a web server

Servlet provides full support for sessions, a way to keep track of a particular user over time as a website's pages are being viewed. They also can communicate directly with a web server using a standard interface.

Servlets can be created using the javax.servlet and javax.servlet.http packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

Running servlets requires a server that supports the technologies. Several web servers, each of which has its own installation, security and administration procedures, support Servlets. The most popular one is the Tomcat- an open source server developed by the Apache Software Foundation in cooperation with Sun Microsystems version 5.5 of Tomcat supports Java Servlet.

Getting Tomcat
The software is available a a free download from Apache's website at the address http://jakarta.apache.org/tomcat. Several versions are available: Linux users should download the rpm of Tomcat.

The javax.servlet package The important interfaces and classes are described in the table below.

The javax.servlet package
The important interfaces and classes are described in the table below.

Interface	Description
Servlet	A java servlet must implement the Servlet interface. This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods.
ServletConfig	The ServletConfig interface is used by the server to pass configuration information to a servlet. Its methods are used by the servlet to retrieve this information.
taglib	Allows programmers to use new tags from tag libraries that encapsulate more complex functionality and simplify the coding of a JSP.
ServletRequest	The ServletRequest interface encapsulates a client request for service. It defines a number of methods for obtaining information about the server, requester, and request.
ServletResponse	The ServletResponse interface is used by a servlet to respond to a request by sending information back to the client.
ServletContext	The ServletContext interface defines the environment in which an applet is executed. It provides methods that are used by applets to access environment information.
SingleThreadModel	The SingleThreadModel interface is used to identify servlets that must be thread-safe. If a servlet implements this interface, the Web server will not concurrently execute the service() method of more than one instance of the servlet.

Class	Description
GenericServlet	The GenericServlet class implements the Servlet interface. You can subclass this class to define your own servlets.
ServletInputStream	The ServletInputStream class is used to access request information supplied by a Web client. An object of this class is returned by the getInputStream() method of the ServletRequest interface.
ServletOutputStream	The ServletOutputStream class is used to send response information to a Web client. An object of this class is returned by the getOutputStream() method of the ServletResponse interface.

The javax.servlet.http package

Interface	Description
HttpServletRequest	The HttpServletRequest interface extends the ServletRequest interface and adds methods for accessing the details of an HTTP request.
HttpServletResponse	The HttpServletResponse interface extends the ServletResponse interface and adds constants and methods for returning HTTP-specific responses
HttpSession	This interface is implemented by servlets to enable them to support browserserver sessions that span multiple HTTP request-response pairs. Since HTTP is a stateless protocol, session state is maintained

	externally using client-side cookies or URL rewriting. This interface provides methods for reading and writing state values and managing sessions.
HttpSessionContext	This interface is used to represent a collection of HttpSession objects that are associated with session IDs.

Class	Description
HttpServlet	Used to create HTTP servlets. The HttpServlet class extends the GenericServlet class.
Cookie	This class represents an HTTP cookie. Cookies are used to maintain session state over multiple HTTP requests. They are named data values that are created on the Web server and stored on individual browser clients. The Cookie class provides the method for getting and setting cookie values and attributes.

Servlet Life Cycle

A servlet's life cycle methods function similarly to the life cycle methods of applets.
- The init(ServletConfig) method is called automatically when a web server first begins a servlet to handle the user's request. The init() method is called only once. ServletConfig is an interface in the javax.servlet package, containing the methods to find out more about the environment in which a servlet is running.
- The servlet action is in the service() method. The service() method checks the HTTP request type (GET, POST, PUT, DELETE etc.) and calls doGet(), doPost(), doPut(), doDelete() etc. methods. A GET request results from normal request for a URL or from an HTML form that has no METHOD specified. The POST request results from an HTML form that specifically lists POST as the METHOD.
- The destroy() method is called when a web server takes a servlet offline.

Using Servlets

One of the main tasks of a servlet is to collect information from a web user and present something back in response. Collection of information is achieved using form, which is a group of text boxes, radio buttons, text areas, and other input fields on the web page. Each field on a form stores information that can be transmitted to a web server and then sent to a Java servlet. web browsers communicate with servers by using Hypertext Transfer Protocol (HTTP).

- Form data can be sent to a server using two kinds of HTTP requests: get and post. When web page calls a server using get or post, the name of the program that handles the request must be specified as a web address, also called uniform resource locator (URL). A get request affixes all data on a form to the end of a URL. A post request includes form data as a header and sent separately from the URL. This is generally preferred, and it's required when confidential information is being collected on the form.
- Java servlets handle both of these requests through methods inherited from the HTTPServlet class: doGet(HttpServletRequest, HttpServletResponse) and doPost(HttpServletRequest, HttpServletResponse). These methods throw two kinds of exceptions: ServletException, part of javax.servlet package, and IOException, an exception in the java.io package.

- The getparameter(String) method is used to retrieve the fields in a servlet with the name of the field as an argument. Using an HTML document a servlet communicates with the user.
- While preparing the response you have to define the kind of content the servlet is sending to a browser. The setContentType(String) method is used to decide the type of response servlet is communicating. Most common form of response is written using an HTML as: setContentType("text/html").
- To send data to the browser, you create a servlet output stream associated with the browser and then call the println(String) method on that stream. The getWriter() method of HttpServletResponse object returns a stream. which can be used to send a response back to the client.

Example

```
import java.io.*;
import javax.servlet.* ;
import javax.servlet.http.*;
public class MyHttpServlet extends HttpServlet
 {
    public void doGet(HttpServletRequest req,HttpServletResponse res)
            throws ServletException, IOException
      {
                // Use "req" to read incoming request
              // Use "res" to specify the HTTP response status
            //Use req.getParameter(String) or getParameterValues(String) to obtain
                 parameters
                 PrintWriter out = res.getWriter();//stream for output
                 // Use "out" to send content to browser
      }
}
```

Request and Response methods

ServletRequest methods	Description
String getParameter(String name)	Obtains the value of a parameter sent to the servlet as part of a get or post request. The name argument represents the parameter name.
Enumeration getParameterNames()	Returns the names of all the parameters sent to the servlet as part of a post request.
String[]getParameterValues(String name)	For a parameter with multiple values, this method Returns an array of strings containing the values for a specified servlet parameter.
String getProtocol()	Returns the name and version of the protocol the request uses in the form protocol/majorVersion.minorVersion for example HTTP/1.
String getRemoteAddr()	Returns the Internet Protocol (IP) address of the client that sent the request.
String getRemoteHost()	Returns the fully qualified name of the client that sent the request.
String getServerName()	Returns the host name of the server that received the request.

| int getServerPort() | Returns the port number on which this request was received. |

HttpServletRequest methods	Description
Cookie[] getCookies()	Cookie[] getCookies() Returns an array of Cookie objects stored on the client by the server.
HttpSession getSession(boolean create)	Returns an HttpSession object associated with the client's current browsing session. This method can create an HttpSession object (True argument) if one does not already exist for the client.
String getServletPath()	Returns the part of this request's URL that calls the servlet
String getMethod()	Returns the name of the HTTP method with which this request was made for example GET, POST, or PUT.
String getQueryString()	Returns the query string that is contained in the request URL after the path.
String getRemoteUser()	Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.

ServletResponse methods	Description
ServletOutputStream getOutputStream()	Obtains a byte-based output stream for sending binary data to the client.
PrintWriter getWriter()	Obtains a character based output stream for sending text data (usually HTML formatted text) to the client.
void setContentType(String type)	Specifies the content type of the response to the browser. The content type is also known as MIME (Multipurpose Internet Mail Extension) type of the data. For examples, "text/html" , "image/gif" etc.
String setContentLength(int len)	Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header

HttpServletResponse methods	Description
void addCookie(Cookie cookie)	Used to add a Cookie to the header of the response to the client.
void sendError(int ec	Sends an error response to the client using the specified status.
void sendError(int ec String msg)	Sends an error response to the client using the specified status code and descriptive message.
void sendRedirect(Stirng url)	Sends a temporary redirect response to the client using the specified redirect location URL.
void setHeader(String name, String value)	Sets a response header with the given name and value.

Writing, Compiling and Running Servlet

Type the first sample program of the self-activity section.

After saving this servlet,compile it with the Java compiler as: javac MyServlet.java.

After compilation a class file with name MyServlet.class is created.

To make the servlet available, you have to publish this class file in a folder on your web server that has been designated for Java servlets. Tomcat provides the classes sub-folder to deploy this servlet's class file. Copy this class file in this classes sub-folder, which is available on the path: tomcat/webapps/ WEB-INF/classes.

Now edit the web.xml file available under WEB-INF sub-folder with the following lines:

```
<servlet>
  <servlet-name>MyServlet</servlet-name>
  <servlet-class> MyServlet</servlet-class>
 </servlet>

<servlet-mapping>
  <servlet-name> MyServlet </servlet-name>
   <url-pattern>/ MyServlet </url-pattern>
</servlet-mapping>
```

Repeat the above sequence of line to run every newly created servlet. Remember, these line lines must be placed somewhere after the <web-app> tag and before the closing </web-app> tag.

After adding these lines, save web.xml file. Restart the Tomcat service and run the servlet by loading its address with a web browser as: **http://localhost:8080/MyServlet.**

Using PostgreSQL – Database Connectivity tool with servlets

Java's Servlet also provides support for data handling using **PostgreSQL** database. For this you have to do few simple steps.
1. Copy the jar file mentioned in Database Connectivity assignment into the subfolder: tomcat/lib/common.
2. Edit the file .bash_profile of your login using command: vi .bash_profile.
3. Add the following line without removing any line.
export CLASSPATH=$CLASSPATH:/$HOME/tomcat/common/lib/<jar file> used in database connectivity assignment.
Example: if I have postgresql-9.3-1104.jdbc4.jar file, I will type the line as export CLASSPATH=$CLASSPATH:/$HOME/tomcat/common/lib/postgresql-9.3-1104.jdbc4.jar

 4. Save this file. Logout from the terminal and re-login.

 5. Create the table student(sno, sname) in your database. Insert few records into this table.

Session Handling
1. Using cookies
2. Using HttpSession class

1. Using Cookies

To keep the track of information about you and the features you want the site to display. This customization is possible because of a web browser features called cookies, small files containing information that a website wants to remember about a user, like username, number of visits, and other. The files are stored on the user's computer, and a website can read only the cookies on the user's system that the site has created. The default behavior of all the web browsers is to accept all cookies.

The javax.servlet.http.Cookie class allows us to create a cookie and send it to the browser. **The methods are:**

Method	Description
int getMaxAge()	Returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.
String getName()	Returns the name of the cookie.
String getValue()	Returns the value of the cookie.
void setMaxAge(int s)	Sets the maximum age of the cookie in seconds.
void setValue (String value)	Assigns a new value to a cookie after the cookie is created.

- The Cookie class in the javax.servlet.http package supports cookies. To create a cookie, call the Cookie(String,String) constructor. The first argument is the name you want to give the Cookie, and the second is the cookie's value.
- To send a cookie, call the addCookie(Cookie) method of an HttpServletResponse object. You can add more than one cookie to a response.
- In a servlet,call the getCookies() method of an HttpServletRequest object to receive an array of Cookie objects. Use getName() and getValue() methods to find out about cookie.

2.HttpSession class

Servlet can retain the state of user through HttpSession, a class that represents sessions. There can be one session object for each user running your servlet.

- A user's session can be created or retrieved by calling the getSession(Boolean) method of the servlet's request object. Use an argument true if a session should be created when one doesn't already exist for the user.

Example

HttpSession hs=req.getSession(true);

```
public void doGet (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
 {
        HttpSession hs = req.getSession(true);
```

```
        // ….
    }
```

- Objects held by session are called its attributes. Call the session's setAttribute(String, Object) method with two arguments: a name to give the attribute and the object.
- To retrieve an attribute, call the getAttribute(String) method with its name as the only argument. It returns the object, which must be cast from object to the desired class, or null if no attribute of that name exists.
- To remove an attribute when it's no longer needed, call removeAttribute(String) with its name as the argument.

Method	Description
Object getAttribute(String name)	Returns the object bound with the specified name in this session, or null if no object is bound under the name.
Enumeration getAttributeNames()	Returns an Enumeration of String objects containing the names of all the objects bound to this session.
long getCreationTime()	Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
long getLastAccessedTime()	Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container received the request.
int getMaxInactiveInterval()	Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses
void RemoveAttribute(String name)	Removes the object bound with the specified name from this session.
void setAttribute(String name, Object value)	Binds an object to this session, using the name specified.
void setMaxInactiveInterval(int seconds)	Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.
void invalidate()	Invalidates this session then unbinds any objects bound to it.
Boolean isNew()	Returns true if it is a new session.
String getId()	Returns a string containing the unique identifier assigned to this session.

Self Activity

Sample Program1 : Program for simple servlet.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet
{
```

```
    public void service(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException
      {
            res.setContentType("text/html");
            PrintWriter pw = rs.getWriter();
            pw.println("<html>");
            pw.println("<body>");
            pw.println("<B> Welcome to My Servlet World… ");
            pw.println("</body>");
            pw.println("</html>");
            pw.close();
      }
 }
```

After saving this servlet, compile it with the Java compiler as: javac MyServlet.java. Run the servlet using **http://localhost:8080/MyServlet**

Sample Program2: To read two numbers and return their Subtraction.

```
// Save the following code as Sub.html

<html>
<head>
 <title>Subtraction of Two Number </title>
 </head>
  <body>
  <form method="post" action="http://server-ip or localhost:8080/SubServlet">
  Enter the Number1 <input type="text" name="No1">
  Enter the Number2 <input type="text" name="No2">
  <br> <input type="Submit" value ="SUB" >
  <br> <input type="Reset" value ="CLEAR" >

</form>
</body>
</html>

// Save the following code as SubServlet.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SubServlet extends HttpServlet
  {
     public void doGet(HttpServletRequest req,HttpServletResponse res)
          throws ServletException, IOException
      {

            int num1 = Integer.parseInt(req.getParameter("No1"));
            int num2 = Integer.parseInt(req.getParameter("No2"));
```

```
        int sub = num1 - num2;
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        pw.println("<h1> Subtraction  </h1> <h3>"+sub+"</h3>");
         pw.close();
      }
   }
```

Sample Program3 :For database handling using servlet

```
//Create a student table (sno, sname)
//The servlet displays all records from the student table on the
  client machine.

import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
public class ServletJdbc extends HttpServlet

{

   public void doGet(HttpServletRequest req,HttpServletResponse res)throws IOException,
ServletException
   {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      try
      {
        out.println("<html>");
        out.println("<body>");

           Class.forName("org.postgresql.Driver");
           out.println("<h1>Driver loaded</h1>");

           Connection c = DriverManager.getConnection
           ("jdbc:postgresql:m2","postgres","");
      out.println("<h1>Connection created</h1>");

           Statement st=c.createStatement();
           ResultSet rs=st.executeQuery("select * from student");
           while(rs.next())
        {
           out.print("<h3>"+rs.getInt(1)+" –
              "+rs.getString(2)+"</h3>");

           out.println("<br>");
          }
        }
```

```
        catch(SQLException e)
        {
         out.println("ERROR"+e);
        }

    out.println("<h1>Hi! Manisha</h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

Run this program as http://server-ip:8080/ServletJdbc

Sample Program4 : For Add the cookies

```
//Save this program as AddCookie.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AddCookie extends HttpServlet
   {
       public void doGet(HttpServletRequest req,HttpServletResponse res)
            throws ServletException, IOException
         {

            Cookie c1=new Cookie("Cookie1","1");
            res.addCookie(c1);
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.print("Cookie added with value 1);
           Cookie c2=new Cookie("Cookie2","2");
           res.addCookie(c2);
           out.print("Cookie added with value 2);
           out.close();
         }
   }

//Save this program as GetCookie.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class GetCookie extends HttpServlet
     {

         public void doGet(HttpServletRequest req,HttpServletResponse res)
              throws ServletException, IOException
```

```
          {

              Cookie [] c=req.getCookies();
              res.setContentType("text/html");
              PrintWriter out = res.getWriter();
              for(int i=0;i<c.length;i++)
                out.println("Cookie Name"+c[i].getName());   pw.close();
          }
     }
```

Run this program as http://server-ip:8080/AddCookie
Run this program as http://server-ip:8080/GetCookie

Sample Program5 : Program for Session using Servlet

```
//Save this program as DemoSession.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
 public class DemoSession extends HttpServlet
  {
      String result1="success";
      String result2="failure";
      public void doGet(HttpServletRequest req,HttpServletResponse res)
          throws ServletException, IOException
            {
                HttpSession hs=req.getSession(true);
                String uname = req.getParameter("txt1");
                String pwd = req.getParameter("txt2");
                res.setContentType("text/html");
                PrintWriter pw=res.getWriter();

                 if((uname.equals("covid"))&&(pwd.equals("covid19")))
                   {
                     pw.print("<a href=http://localhost:8080/NewInfo.html>
                         Login Success</a>");
                     hs.setAttribute("loginID",result1);
                   }
                  else
                   {
                      pw.print("<a href=http://localhost:8080/NewInfo.html> Kick Out</a>");
                      hs.setAttribute("loginID",result2);
                   }
                  pw.close();
            }
  }

 <!—HTML File for NewInfo.html --> <html>
  <head> <title></title>
  </head>
  <body> <form method="post" action="http://localhost:8080/SessionInfo">
```

```
 <input type="Submit" value="Read Session Value">
 </form>
 </body>
</html>
```

//Save this program as SessionInfo.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

  public class SessionInfo extends HttpServlet
    {
       String readloginid;
       public void doGet(HttpServletRequest req,HttpServletResponse res)
              throws ServletException, IOException
        {

              HttpSession hs = req.getSession(true);
              readloginid = hs.getId();
              res.setContentType("text/html");
              PrintWriter pw = res.getWriter();

               if(hs.getAttribute("loginID").equals("success"))
                 pw.print("Your Session ID " + readloginid);
              else
                  pw.print("<h1>Session Expired </h1>");
              pw.close();
        }
    }
```

Create an html file for login and password and use http://server-ip:8080/SessionDemo in the Form Action tag.

What is JSP?

JSP is Java Server Page, which is a dynamic web page and used to build dynamic websites. To run jsp, we need web server which can be tomcat provided by apache, it can also be jRun, jBoss(Redhat), weblogic (BEA) , or websphere(IBM).

JSP is dynamic file whereas Html file is static. HTML can not get data from database or dynamic data. JSP can be interactive and communicate with database and controllable by programmer. It is saved by extension of .jsp. Each Java server page is compiled into a servlet before it can be used. This is normally done when the first request to the JSP page is made.

A JSP contains 3 important types of elements

1. Directives:- these are messages to the JSP container that is the server program that executes JSPs.
2. Scripting elements:- These enables programmers to insert java code which will be a part of the resultant servlet.

3. Actions:- Actions encapsulates functionally in predefined tags that programmers can embedded in a JSP.

JSP Directives

Directives are message to the JSP container that enable the programmer to specify page setting to include content from other resources & to specify custom tag libraries for use in a JSP.

Syntax

<%@ name attribute1="….", attribute2="…"…%>

Directive	Description
page	Defines page settings for the JSP container to process.
include	Causes the JSP container to perform a translation-time insertion of another resource's content. The file included can be either static (HTML file) or dynamic (i.e., another tag file)
taglib	Allows programmers to use new tags from tag libraries that encapsulate more complex functionality and simplify the coding of a JSP.

Page Directive

The page directives specify global settings for the JSP in the JSP container. There can be many page directives, provided that there is only one occurrence of each attribute.

Syntax

```
<% @ page
[ language="java" ]
[ extends="package.class" ]
[ import="{package.class | package.*}, ..." ]
[ session="true|false" ]
[ buffer="none|8kb|sizekb" ]
[ autoFlush="true|false" ]
[ isThreadSafe="true|false" ]
[ info="text" ]
[ errorPage="relativeURL" ]
[ contentType="mimeType [ ; charset=characterSet ]" "text/html ; charset=ISO-8859-1" ]
[ isErrorPage="true|false" ]
[ pageEncoding="characterSet | ISO-8859-1" ]    %>
```

Scripting Elements

1. **Declarations**
   A declaration declares one or more variables or methods that you can use in Java code later in the JSP file.
   Syntax

   <%! Java declaration statements %>

Example

<%! private int count = 0; %> <%! int i = 0; %>

2. Expressions
An expression element contains a java expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.

Syntax

<%= expression %>

Example

Name is <%= request.getParameter("name") %>

3. Scriptlet
A scriptlet contains a set of java statements which is executed. A scriptlet can have java variable and method declarations, expressions, use implicit objects and contain any other statement valid in java.

Syntax

<% statements %>

Example

```
<%    String name = request.getParameter("userName");
      out.println("Hello  " + name); %>
```

Implicit objects used in JSP

Implicit object	Description
applicat ion	A javax.servlet.ServletContext object that represents the container in which the JSP executes. It allows sharing information between the jsp page's servlet and any web components with in the same application.
config	A javax.servlet.ServletConfig object that represents the JSP configuration options. As with servlets, configuration options can be specified in a Web application descriptor (web.xml). The method getinitparameter() is used to access the initialization parameters.
exception	A java.lang.Throwable object that represents an exception that is passed to a JSP error page. This object is available only in a JSP error page.
out	A javax.servlet.jsp.JspWriter object that writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
page	An Object that represents the current JSP instance.

pageContext	A javax.servlet.jsp.PageContext object that provides JSP programmers with access to the implicit objects discussed in this table.
request	An object that represents the client request and is normally an instance of a class that implements HttpServletRequest. If a protocol other than HTTP is used, this object is an instance of a subclass of javax.servlet.Servlet-Request. It uses the getParameter() method to access the request parameter.
response	An object that represents the response to the client and is normally an instance of a class that implements HttpServletResponse (package javax.servlet.http). If a protocol other than HTTP is used, this object is an instance of a class that implements javax.servlet.ServletResponse.
session	A javax.servlet.http.HttpSession object that represents the client session information. This object is available only in pages that participate in a session.

To run JSP files: all JSP code should be copied (Deployed) into webapps folder in the tomcat server. To execute the file, type: http://server-ip:8080/Programname.jsp

Self Activity

Sample Program1 : Simple display on browser.

```
/* type this as first.jsp */
<html>
<body>
 <% out.print("DREAMS Don't work UNLESS YOU DO!"); %>
</body>
</html>
```

Sample Program2 : To display current date.

```
<%@ page  language="java" import="java.util.*" %>
<html>
    <body> Current Date time: <%=new java.util.Date()%>
</body>
</html>
```

Sample Program3 : To multiplication of two numbers "MultNumbers.jsp"

```
<%@ page language="java"%>
<html>
<head>
<title>Add number program in JSP </title>
 </head>
 <body>
 <form method = "post" action = "MultNumbers.jsp">
 Enter Number 1 <input type ="text" name = "No1">
 Enter Number 2 <input type ="text" name = "No2">
 <input type="submit" value="RESULT"/>
    <%
       int p = Integer.parseInt(request.getParameter("No1"));
       int q = Integer.parseInt(request.getParameter("No2"));
       int result = p * q;
```

```
    out.print("Multiplication of p and q :"+result);
  %>

 </form>
 </body>
 </html>
```

Lab Assignments

Set A

a) Design a servlet that provides information about a HTTP request from a client, such as IP address and browser type. The servlet also provides information about the server on which the servlet is running, such as the operating system type, and the names of currently loaded servlets.

b) Write a Program to make use of following JSP implicit objects:
   i.   out: To display current Date and Time.
   ii.   request: To get header information.
   iii. response: To Add Cookie
   iv.   config: get the parameters value defined in <init-param>
   v.   application: get the parameter value defined in <context-param>
   vi.   session: Display Current Session ID
   vii. pageContext: To set and get the attributes.
   viii. page: get the name of Generated Servlet

c) Write a program to create a Online Book purchase. User must be login and then purchase the book. Each page should have a page total. The last page should display a total book and bill, which consists of a page total of what ever the purchase has been done and print the total. (Use HttpSession)

Set B

a) Design an HTML page which passes customer number to a search servlet. The servlet searches for the customer number in a database (customer table) and returns customer details if found the number otherwise display error message.

b) Design an HTML page containing option buttons (Maths, Physics, Chemistry and Biology) and buttons submit and reset. When the user clicks submit, the server responds by adding a cookie containing the selected subject and sends a message back to the client. Program should not allow duplicate cookies to be written.

c) Write a JSP program to display the details of PATIENT (PatientNo, PatientName, PatientAddress, Patientage,PatientDiease) in tabular form on browser

Set C

a) Create a JSP page for an online multiple choice test. The questions are randomly selected from a database and displayed on the screen. The choices are displayed

using radio buttons. When the user clicks on next, the next question is displayed. When the user clicks on submit, display the total score on the screen.

b) Consider the following entities and their relationships Movie (movie_no, movie_name, release_year) Actor(actor_no, name) Relationship between movie and actor is many – many with attribute rate in Rs. Create a RDB in 3 NF answer the following: a) Accept an actor name and display all movie names in which he has acted along with his name on top. b) Accept a movie name and list all actors in that movie along with the movie name on top.

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-charge

Assignment 5: Spring

Objectives

- To understand basic concept of Spring.
- To demonstrate the use of Spring.
- To create and understand the steps to develop Spring application

Reading

You should read the following topics before starting this exercise
- Concept of Spring and Framework.
- Types of Dependency Injection.
- Spring IoC(Inversion of Control)

Ready Reference

What is Spring?
Spring is an open source framework created to address the complexity of enterprise application development.Spring is a very lightweight framework which provides well-defined infrastructure support for developing Java application.

Why Spring?
Spring is considered to be a secure, low-cost and flexible framework. Spring improves coding efficiency and reduces overall application development time because it is lightweight and efficient at utilizing system resources. Spring removes tedious configuration work so that developers can focus on writing program logic. Spring handles the infrastructure so developers can focus on the application.

Spring Framework

The Spring Framework is one of the most popular Java-based application Frameworks. It is an application framework and Inversion of Control (IoC) container for the Java platform. The Spring Framework is a mature, powerful and highly flexible framework focused on building Web applications in Java. The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Spring Module
JDBC Module: This module provides the JDBC abstraction layer and helps to avoid tedious JDBC coding.
ORM Module: This module provides integration for object relational mapping APIs such as JPA, Hibernate, JDO, etc.
JMS (Java Messaging Service) Module: This module contains features for producing and consuming messages.
OXM Module: This module provides Object/XML binding.

Transaction Module: This model supports programmatic and declarative transaction management for classes that implement special interfaces and for all the POJOs.
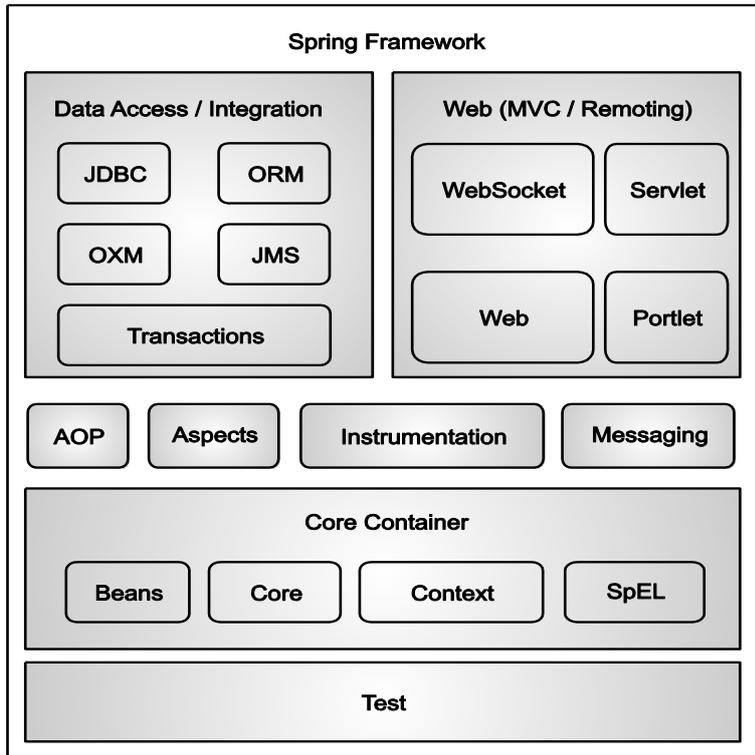
Spring MVC

Model: A model contains the data of the application. A data can be a single object or a collection of objects. The Model encapsulates the application data and in general they will consist of POJO.

View: View is responsible for presenting data to the end user. A view represents the provided information in a particular format. The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

Controller: The controller is a logic that is responsible for processing and acting on user requests. The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

Annotations of Spring MVC

Annotation	Description
@Controller	It represents the controller class
@RequestMapping	It can be used for the mapping of incoming requests.
@GetMapping	It is used to map HTTP Get requests.
@PostMapping	It is used to map HTTP Post requests.
@RequestParam	It reads the HTML form data.
@ModelAttribute	This annotation accesses elements present in the model.

Spring MVC Validation Annotations

| Annotation	Description
@NotNull	Checks that the annotated value is not null.
@Min	Must be a number >=value
@Max	Must be a number <=value
@Size	Total no of characters must match the given size.
@Pattern	Must match a regular expression pattern.
@Future	Date must be in the future of the given date.
@Past	Date must be in the past of the given date.

pom.xml

It stands for **Project Object Model.** POM is a fundamental unit of work in Maven. Project Object Model (POM) is a XML file that contains information about the project and configuration details used by Maven to build the project. When executing a task ,Maven looks for the POM in the current directory. It reads the POM, gets the required configuration and information, then executes the goal. Configurations specified in the POM are the project dependencies, the plugins or goals that can be executed, the build project.

Download the Spring Tool Suits 4 for that follow the link https://spring.io/tools

Example

Step 1: Create Java Project: The first step is to lunch the workspace and then create a simple spring starter project. Follow the option File → New → Spring Starter Project and finally select Java Project wizard from the wizard list. Lunch workspace window as shown in Fig.
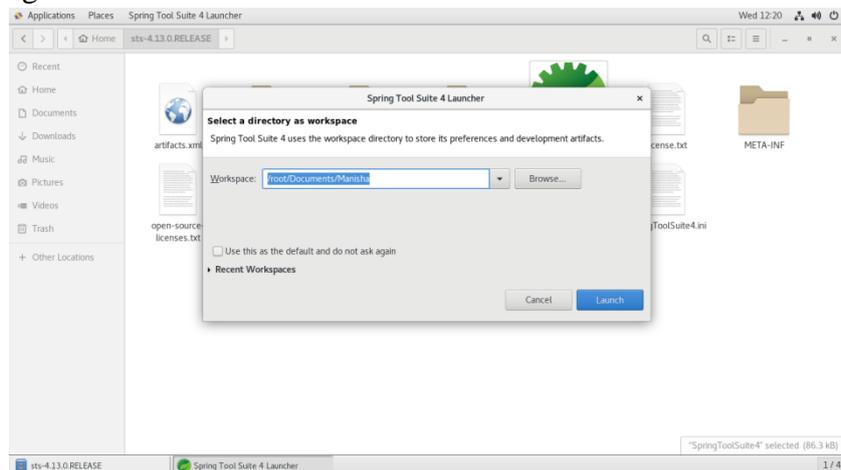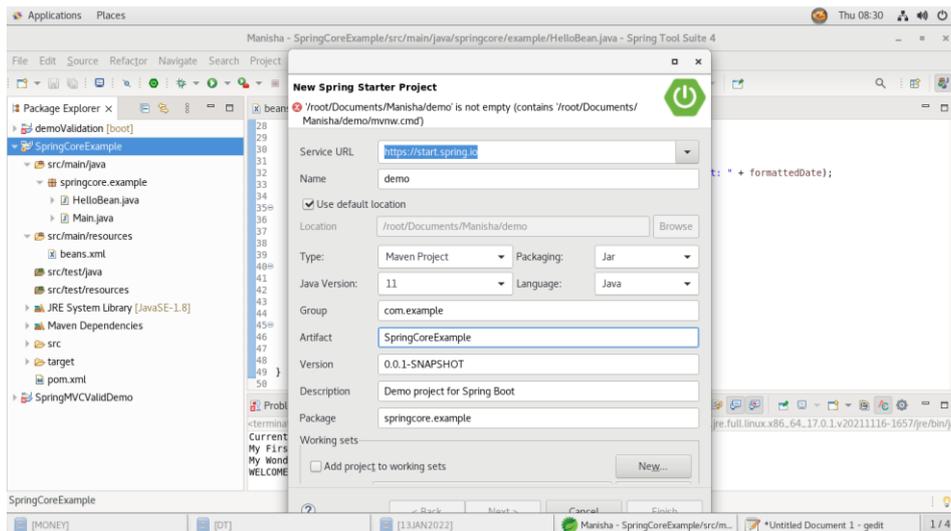
Now name the project as SpringCore is created successfully.

Step 2: Add Required Libraries: As a second step let us add Spring Framework and common logging API libraries in our project. To do this, right-click on the project name springcore and then follow the following option available in the context menu − Build Path → Configure Build Path to display the Java Build Path window.

Now use Add External JARs button available under the Libraries tab to add the following core JARs from Spring Framework and Common Logging installation directories:
o   commons-logging-1.1.1
o   spring-aop-4.1.6.RELEASE
o   spring-aspects-4.1.6.RELEASE
o   spring-beans-4.1.6.RELEASE
o   spring-context-4.1.6.RELEASE
o   spring-context-support-4.1.6.RELEASE
o   spring-core-4.1.6.RELEASE
o   spring-expression-4.1.6.RELEASE
o   spring-instrument-4.1.6.RELEASE
o   spring-instrument-tomcat-4.1.6.RELEASE
o   spring-jdbc-4.1.6.RELEASE
o   spring-jms-4.1.6.RELEASE
o   spring-messaging-4.1.6.RELEASE
o   spring-orm-4.1.6.RELEASE
o   spring-oxm-4.1.6.RELEASE
o   spring-test-4.1.6.RELEASE
o   spring-tx-4.1.6.RELEASE
o   spring-web-4.1.6.RELEASE
o   spring-webmvc-4.1.6.RELEASE
o   spring-webmvc-portlet-4.1.6.RELEASE
o   spring-websocket-4.1.6.RELEASE

Step 3: Create Source Files: Now let us create actual source files under the SpringCore project. First we need to create a package called springcore.example. To do this, right

click on src in package explorer section and follow the option − New → Package.
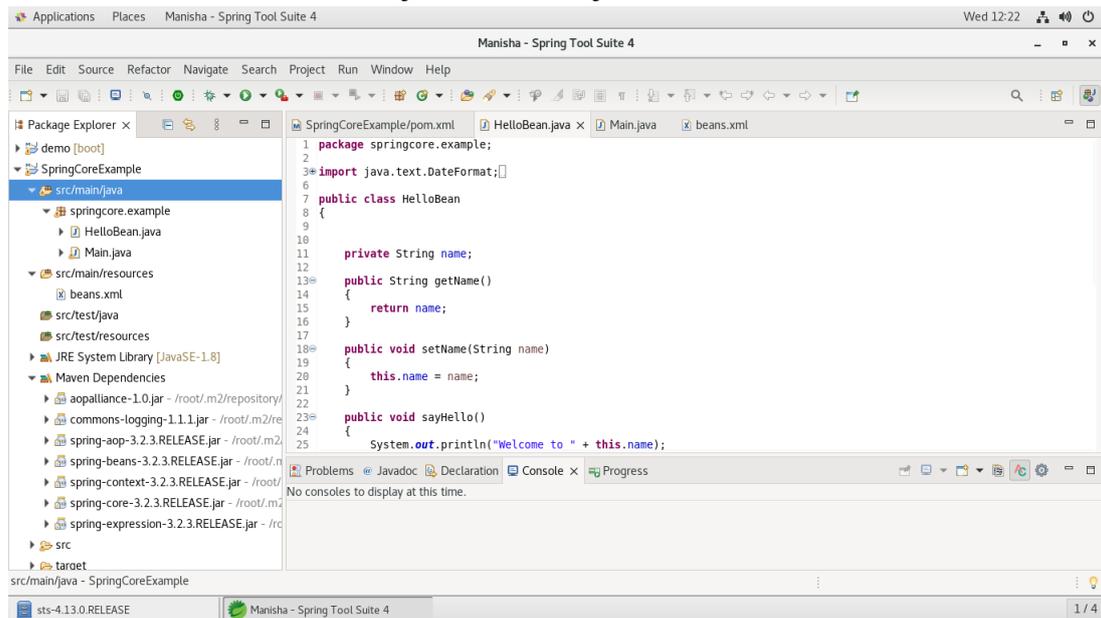Next we will create HelloBean.java and Main.java files.

Fig. 5.5

```
// Here is the content of HelloBean.java file:
package springcore.example;
public class HelloBean
{

        private String name;

        public String getName()
        {
                return name;
        }

        public void setName(String name)
        {
                this.name = name;
        }

        public void sayHello()
        {
                System.out.println("Hello" + this.name);
        }
}
```

Content of the second file Main.java

```
//Main.java

package springcore.example;
```

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main
{

        private static ApplicationContext context;
        public static void main(String[] args)
        {
                context = new ClassPathXmlApplicationContext("beans.xml");
                HelloBeanhelloBean = (HelloBean) context.getBean("HelloBean");
                helloBean.sayHello();

        }
}
```

Step 4: Create Bean Configuration File: We need to create a Bean Configuration file which is an XML file and acts as cement that glues the beans, i.e. the classes together. This file needs to be created under the src directory (Src/main/resources)beans.xml.
Usually developers name this file as Beans.xml, but we are independent to choose any name we like.

```
//beans.xml

<beansxmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"xmlns:context="http://www.springframework.org/schema/context"
        xmlns:jee="http://www.springframework.org/schema/jee"xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:task="http://www.springframework.org/schema/task"
        xsi:schemaLocation="http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-3.2.xsd">

        <context:component-scanbase-package="springcore.examples"/>

        <beanid="HelloBean"class="springcore.example.HelloBean">
                <propertyname="name"value="Spring Programe"/>
        </bean>
```

```
</beans>
```

```
//pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>springcore_example</groupId>
  <artifactId>SpringCoreExample</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <!-- JDK 8 configuration below -->

  <properties>
  <spring.version>3.2.3.RELEASE</spring.version>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

<!-- completed -->

  <dependencies>
            <dependency>
                    <groupId>org.springframework</groupId>
                    <artifactId>spring-core</artifactId>
                    <version>${spring.version}</version>
            </dependency>
            <dependency>
                    <groupId>org.springframework</groupId>
                    <artifactId>spring-context</artifactId>
                    <version>${spring.version}</version>
            </dependency>
       </dependencies>
</project>
```

Step 5: Running the Program: Once we are done with creating the source and beans configuration files, we are ready for this step, which is compiling and running the program. To do this, keep Main.Java file tab active and use either Run option. If everything is fine with the application, this will print the following message in console.

Spring MVC
Example

```java
package jbr.springmvc.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import jbr.springmvc.model.User;
import jbr.springmvc.service.UserService;

@Controller
public class RegistrationController {
  @Autowired
  public UserService userService;

  @RequestMapping(value = "/register", method = RequestMethod.GET)
  public ModelAndView showRegister(HttpServletRequest request, HttpServletResponse
response) {
    ModelAndView mav = new ModelAndView("register");
    mav.addObject("user", new User());

    return mav;
  }
```

```java
@RequestMapping(value = "/registerProcess", method = RequestMethod.POST)
public ModelAndView addUser(HttpServletRequest request, HttpServletResponse response,
    @ModelAttribute("user") User user) {

  userService.register(user);

  return new ModelAndView("welcome","name", user);
 }
}
package jbr.springmvc.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import jbr.springmvc.model.Login;
import jbr.springmvc.model.User;
import jbr.springmvc.service.UserService;

@Controller
public class LoginController {

  @Autowired
  UserService userService;

  @RequestMapping(value = "/login", method = RequestMethod.GET)
  public ModelAndView showLogin(HttpServletRequest request, HttpServletResponse
response) {
    ModelAndView mav = new ModelAndView("login");
    mav.addObject("login", new Login());

    return mav;
  }

  @RequestMapping(value = "/loginProcess", method = RequestMethod.POST)
  public ModelAndView loginProcess(HttpServletRequest request, HttpServletResponse
response,
      @ModelAttribute("login") Login login) {
    ModelAndView mav = null;

    User user = userService.validateUser(login);

    if (null != user) {
```

```
    mav = new ModelAndView("welcome", "firstname", login);
    //mav.addObject("firstname", user.getFirstname(),user.getPassword());
   } else {
    mav = new ModelAndView("login");
    mav.addObject("message", "Username or Password is wrong!!");
   }
  return mav;
 }
}


package jbr.springmvc.dao;
import jbr.springmvc.model.User;

publicinterface UserDao {
int register(User user);
}

package jbr.springmvc.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
importjava.util.List;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import jbr.springmvc.model.User;

publicclass UserDaoImpl implements UserDao {

        @Autowired
        DataSource datasource;

        @Autowired
        JdbcTemplate jdbcTemplate;

        publicint register(User user) {
                String sql = "insert into userss values(?,?,?)";

                returnjdbcTemplate.update(sql, new Object[] { user.getName(), user.getId(),
user.getAge() });
        }

}

class UserMapper implements RowMapper<User> {

        public User mapRow(ResultSet rs, intarg1) throws SQLException {
```

```java
                User user = new User();

                user.setName(rs.getString("name"));
                user.setId(rs.getString("id"));
                user.setAge(rs.getString("age"));
                returnuser;
        }
}

package jbr.springmvc.model;
public class User {

        private String name;
        private String age;
        private String id;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getAge() {
                return age;
        }

        public void setAge(Stringage) {
                this.age = age;
        }

        public String getId() {
                returnid;
        }

        public void setId(Stringid) {
                this.id = id;
        }

}
package jbr.springmvc.service;
import jbr.springmvc.model.User;
publicinterface UserService {
int register(User user);
}

package jbr.springmvc.service;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import jbr.springmvc.dao.UserDao;
import jbr.springmvc.model.User;

public class UserServiceImpl implements UserService {

  @Autowired
  public UserDao userDao;

  public int register(User user) {
   return userDao.register(user);
  }

}
```

Following is the content of Spring Web configuration file web.xml:
web.xml
```xml
<?xmlversion="1.0"encoding="UTF-8"?>
<web-appxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://java.sun.com/xml/ns/javaee"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
       version="3.0">

       <display-name>Archetype Created Web Application</display-name>

       <welcome-file-list>
               <welcome-file>home.jsp</welcome-file>
       </welcome-file-list>

       <servlet>
               <servlet-name>spring-mvc</servlet-name>
               <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
               <load-on-startup>1</load-on-startup>
       </servlet>
       <servlet-mapping>
               <servlet-name>spring-mvc</servlet-name>
               <url-pattern>/</url-pattern>
       </servlet-mapping>

       <!-- <context-param>
               <param-name>contextConfigLocation</param-name>
               <param-value>/WEB-INF/spring-mvc-servlet.xml</param-value>
       </context-param>

       <listener>
               <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
       </listener> -->
</web-app>
```

Pom.xml

```xml
<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>jbr</groupId>
<artifactId>springmvc-student-reg-login</artifactId>
<packaging>war</packaging>
<version>0.1</version>
<name>springmvc-student-reg-login</name>
<url>http://maven.apache.org</url>

<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<spring.version>5.2.4.RELEASE</spring.version>
<junit.version>4.12</junit.version>
<servlet.version>3.1.0</servlet.version>
<java.version>1.8</java.version>
<mysql.connector.version>8.0.17</mysql.connector.version>
</properties>

<dependencies>
<!-- Junit -->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>${junit.version}</version>
<scope>test</scope>
</dependency>

<!-- Spring Framework -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>${spring.version}</version>
</dependency>

<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-beans</artifactId>
<version>${spring.version}</version>
</dependency>

<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>${spring.version}</version>
</dependency>
```

```xml
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>${spring.version}</version>
</dependency>

<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>${spring.version}</version>
<scope>test</scope>
</dependency>

<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>${spring.version}</version>
</dependency>

<!-- Postgresql database driver -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>${mysql.connector.version}</version>
</dependency>

<!-- Servlet API -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>${servlet.version}</version>
</dependency>
</dependencies>

<build>
<finalName>springmvc-user-reg-login</finalName>
<sourceDirectory>src/main/java</sourceDirectory>
<plugins>
<plugin>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.5.1</version>
<configuration>
<source>${java.version}</source>
<target>${java.version}</target>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

Following is the content of another Spring Web configuration file HelloWeb-servlet.xml:

Spring-mvc-servlet.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beansxmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.0.xsd">

        <importresource="classpath:jbr/config/user-beans.xml"/>

        <context:component-scanbase-package="jbr.springmvc"/>
        <context:annotation-config/>

        <bean

        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <propertyname="prefix"value="/jsp/"/>
                <propertyname="suffix"value=".jsp"/>
        </bean>

</beans>
```

Following is the content of Spring view file hello.jsp:

Register.jsp

```
<%@taglibprefix="form"uri="http://www.springframework.org/tags/form"%>

<%@pagelanguage="java"contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=ISO-8859-1">
<title>Student Registration</title>
</head>
<body>
        <form:formid="regForm"modelAttribute="user"action="registerProcess"method="post">

                <tablealign="center">
                        <tr>
                                <td><form:label path="name">name</form:label></td>
                                <td><form:input path="name"name="name"id="name"/></td>
                        </tr>
                        <tr>
                                <td><form:labelpath="id">id</form:label></td>
```

```
                        <td><form:inputpath="id"name="id"id="id"/></td>
                </tr>
                <tr>

                        <td><form:labelpath="age">age</form:label></td>
                        <td><form:inputpath="age"name="age"id="age"/></td>
                </tr>

                <tr>

                        <td></td>

        <td><form:buttonid="register"name="register">Submit</form:button></td>
                </tr>
                <tr></tr>
                <tr>

                        <td></td>
                        <td><ahref="home.jsp">Home</a></td>
                </tr>
        </table>
    </form:form>

</body>
</html>
```

Welcome.jsp
```
<%@pagelanguage="java"contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01
Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=ISO-8859-1">
<title>Welcome</title>
</head>
<body>
        <table>
                <tr>
                        <td>Name : ${user.name}</td>
                </tr>
                <tr>
                        <td>Age : ${user.age}</td>
                </tr>
                <tr>
                        <td>ID  : ${user.id}</td>
                </tr>
                <tr>
                        <td><ahref="home.jsp">Home</a></td>
                </tr>
        </table>
</body>
</html>
```

Home.jsp
```
<%@page language="java"contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01
Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=ISO-8859-1">
<title>Welcome</title>
</head>
<body>
        <tablealign="center">
                <tr>
                        <td><ahref="register">Register</a></td>
                </tr>
        </table>
</body>
</html>
```

Now start the Tomcat server and make sure you are able to access other web pages from webapps folder using a standard browser.
Run the project File-Run As - Run on Server

Lab Assignments

Set A

    a) Create a Spring core example to display the message "If you can't explain it simply, you don't understand it well enough".

    b) Write a program to display the Current Date using spring.

Set B

    a) Design simple student information like Student_id, Student_Name and Student_Age using Spring Framework.

    b) Design the Employee login form application using spring form MVC validation.

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-charge

T.Y.B.Sc (Comp. Sc.) Lab – III, Sem – II