

**BMS INSTITUTE OF TECHNOLOGY AND  
MANAGEMENT**

(An autonomous Institution, Affiliated to VTU)  
Doddaballapura Main Road, Avalahalli, Yelahanka,  
Bengaluru 560064



**DEPARTMENT OF MCA**

**LABORATORY MANUAL**

**SECOND SEMESTER MCA – AUTONOMOUS  
SCHEME**

**JAVA PROGRAMMING**

**22MCA204**

**(Revised Scheme 2021)**

**Prepared by:**

**Mr. Shivakumara T, Assistant Professor,  
Department of MCA, BMSITM**

**[shivakumarat@bmsit.in](mailto:shivakumarat@bmsit.in)**

**Department Vision**

To develop quality professionals in Computer Applications who can provide sustainable solutions to the societal and industrial needs

**Department Mission**

Facilitate effective learning environment through quality education, state-of-the-art facilities, and orientation towards research and entrepreneurial skills

**Programme Educational Objectives (PEOs)**

**PEO1:** Develop innovative IT applications to meet industrial and societal needs

**PEO2:** Adapt themselves to changing IT requirements through life-long learning

**PEO3:** Exhibit leadership skills and advance in their chosen career

**Programme Outcomes (POs)**

**PO1:** Apply knowledge of computing fundamentals, computing specialization, mathematics and domain

knowledge to provide IT solutions.

**PO2:** Identify, analyse and solve IT problems using fundamental principles of mathematics and computing sciences.

**PO3:** Design, Develop and evaluate software solutions to meet societal and environmental concerns.

**PO4:** Conduct investigations of complex problems using research based knowledge and methods to provide valid conclusions.

**PO5:** Select and apply appropriate techniques and modern tools for complex computing activities.

**PO6:** Understand professional ethics, cyber regulations and responsibilities.

**PO7:** Involve in life-long learning for continual development as an IT professional.

**PO8:** Apply and demonstrate computing and management principles to manage projects in multidisciplinary environments by involving in different roles

**PO9:** Comprehend & write effective reports and make quality presentations.

**PO10:** Understand and assess the impact of IT solutions on socio-environmental Issues.

**PO11:** Work collaboratively as a member or leader in multidisciplinary teams.

**PO12:** Identify potential business opportunities and innovate to create value to the society and seize that opportunity.

**IPCC Course – Java Programming**

Course Code – 22MCA204

Course Coordinator – Shivakumara T

Credits – 03

**Table of Contents**

1. Hardware and Software Requirements.....	4
2. Introduction to Java and Advanced Java Programming.....	5
3. Index of Lab Concepts.....	16
4. Concepts Related Programs.....	17
5. Proposed Viva Questions.....	53
6. Self-Assessment Quiz.....	54
7. References.....	54

## **1. Hardware and Software Requirements**

### **Hardware:**

RAM : 512 MB

HDD : 40 GB

### **Software:**

OS : WINDOWS XP

SDK : JDK 1.8

Plugin : MySQL Connector

Editor : Notepad / Notepad++ / Netbeans / Eclipse

Database : MYSQL

Browser : Internet Explorer

### **Set the following Environmental Variables:**

User Variable-catalina\_home: C:\Program Files\Apache Software Foundation\Apache Tomcat 6.0.26

System Variable – path: C:\Program Files\Microsoft SQL Server\80\Tools\BIN;  
C:\Program Files\Java\jdk1.6.0\bin;

## **2. Introduction to Java and Advanced Java Programming**

### **Java Programming Fundamentals**

The Java Language, The Key Attributes of Object-Oriented Programming, The Java Development Kit, A First Simple Program, The Java Keywords, Identifiers in Java, The Java Class Libraries.

### **Introducing Data Types and Operators**

Java's Primitive Types, Literals, A Closer Look at Variables, The Scope and Lifetime of Variables, operators, Shorthand Assignments, Type conversion in Assignments, Using Cast.

### **Program Control Statements**

Input characters from the Keyboard, if statement, Nested ifs, if-else-if Ladder, Switch Statement, Nested switch statements, for Loop, Enhanced for Loop, While Loop, do-while Loop, Use break, Use continue, Nested Loops.

### **Introducing Classes, Objects and Methods**

Class Fundamentals, How Objects are Created, Reference Variables and Assignment, Methods, Returning from a Method, Returning Value, Using Parameters, Constructors, Parameterized Constructors, The new operator Revisited, Garbage Collection and Finalizers, The this Keyword.

### **More Data Types and Operators**

Arrays, Multidimensional Arrays, Alternative Array Declaration Syntax, Assigning Array References, Using the Length Member, The For-Each Style for Loop, Strings,

### **String Handling**

String Fundamentals, The String Constructors, Three String-Related Language Features, The Length() Method, Obtaining the characters within a string, String comparison, using indexOf() and lastIndexOf(), Changing the case of characters within a string, StringBuffer and String Builder.

### **Introduction to Java**

- Java is an object-oriented programming language developed by Sun Microsystems, a company best known for its high-end Unix workstations.
- Java is modeled after C++
- Java language was designed to be small, simple, and portable across platforms and operating systems, both at the source and at the binary level (more about this later).
- Java also provides for portable programming with applets. Applets appear in a Web page much in the same way as images do, but unlike images, applets are dynamic and interactive.

- Applets can be used to create animations, figures, or areas that can respond to input from the reader, games, or other interactive effects on the same Web pages among the text and graphics.

### The Java Language

The key features of Java are security and portability (platform-independent nature). When we download any application from the internet, there is a chance that the downloaded code contain virus. But, downloading the Java code assures security. Java program can run on any type of system connected to internet and thus provides portability

- Java is **object-oriented**:
- Structured in terms of **classes**, which group data with operations on that data Can construct new classes by **extending** existing ones
  - Java designed as A **core language** plus
  - A rich collection of **commonly available packages**
- Java can be embedded in Web pages



### The origins of Java

- James Gosling initiated Java language project in June 1991
- The language, initially called 'Oak' after an oak tree later was renamed as JAVA
- Sun released the first public implementation as Java 1.0 in 1995.
- It promised **Write Once, Run Anywhere** (WORA)



### Java Is Platform-Independent

- **Platform-independence is a program's capability of moving easily from one computer system to another**
- Platform independence is one of the most significant advantages that Java has over other programming languages, particularly for systems that need to work on many different platforms.
- Java is platform-independent at both the source and the binary level.

The Platform independent nature can be interpreted by two things:

- Operating System Independent: Independent of the operating system on which your source code is being run.

- **Hardware Independent:** Doesn't depend upon the hardware on which your java code is run upon i.e. it can run on any hardware configuration

These two points make it a platform independent language. Hence, the users do not have to change the syntax of the program according to the Operating System and do not have to compile the program again and again on different Operating Systems. The meaning of this point can be understood as you read further. C and C++ are platform dependent languages as the file which compiler of C, C++ forms is a .exe(executable) file which is operating system dependent. The C/C++ program is controlled by the operating system whereas, the execution of a Java program is controlled by JVM (Java Virtual Machine).

Java's contribution to the Internet

Java Innovated a new type of Internet programming called APPLET

Java Applets:

Java program designed to be transmitted over the internet and automatically executed by JAVA compatible Web browser

Java Applets:

The team returned to work up a Java technology-based clone of Mosaic they named "WebRunner" (after the movie *Blade Runner*), later to become officially known as the HotJava™ browser. It was 1994. WebRunner was just a [demo](#), but an impressive one: It [brought to life](#), for the first time, animated, moving objects and [dynamic executable](#) content inside a Web browser. That had never been done.

Security:

**Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Portability:

**Portable** – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

**The Evolution of JAVA**

First version released in 1995

Eight major versions released since then

JDK 1.0 (1996) JDBC, Distributed Objects

JDK 1.1 (1997) New Event Model

J2SE 1.2 (1998) Swing

J2SE 1.3 (2000) Cleanup

J2SE 1.4 (2002)

J2SE 5.0 (2004) Generics

**J2SE 6.0** (2006)

J2SE 7.0 (2011)

## The Key Attributes of Object-oriented programming

Java is purely object oriented programming (OOP) language. Here, we will discuss the basics of OOPs concepts.

### *Two Paradigms*

Every program consists of two elements viz. code and data. A program is constructed based on two paradigms: *a program written around **what is happening*** (known as **process-oriented model**) and *a program written around **who is being affected*** (known as **object-oriented model**). In process oriented model, the program is written as a series of linear (sequential) steps and it is thought of as **code acting on data**. Since this model fails to focus on real-world entities, it will create certain problems as the program grows larger.

The object-oriented model focuses on real-world data. Here, the program is organized as data and a set of well-defined interfaces to that data. Hence, it can be thought of as **data controlling access to code**. This approach helps to achieve several organizational benefits.

### *Abstraction*

Abstraction can be thought of as **hiding the implementation details from the end-user**. A powerful way to manage abstraction is through the use of hierarchical classifications. This allows us to layer the semantics of complex systems, breaking them into more manageable pieces. For example, we consider a car as a vehicle and can be thought of as a single object.



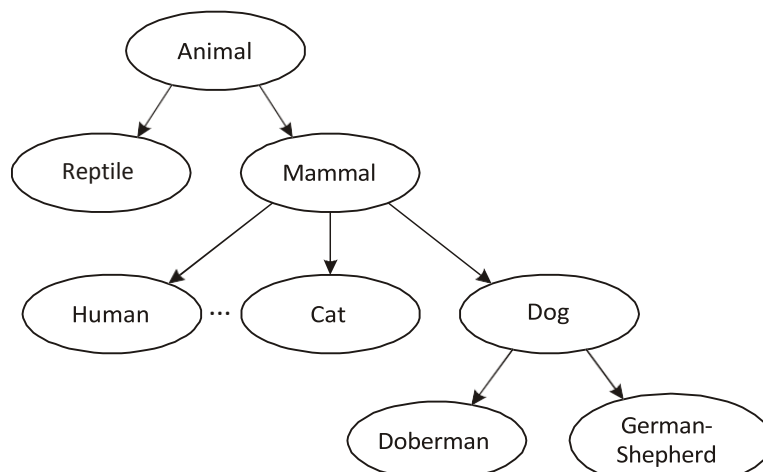
But, from inside, car is a collection of several subsystems viz. steering, brakes, sound system, engine etc. Again, each of these subsystems is a collection of individual parts (Ex. Sound system is a combination of a radio and CD/tape player). As an owner of the car, we manage it as an individual entity by achieving hierarchical abstractions.

Hierarchical abstractions of complex systems can also be applied to computer programs. The data from a traditional process-oriented program can be transformed by abstraction into its component objects. A sequence of process steps can become a collection of messages between these objects. Thus, each of these objects describes its own unique behavior. You can treat these objects as concrete entities that respond to messages telling them to *do something*. This is the essence of object-oriented programming.

**OOPs Principles:** Encapsulation, Inheritance and Polymorphism are the basic principles of any object oriented programming language.

**Encapsulation** is the mechanism to bind the data and code working on that data into a single entity. It provides the security for the data by avoiding outside manipulations. In Java, encapsulation is achieved using **classes**. A **class** is a collection of data and code. An **object** is an **instance** of a class. That is, several objects share a common structure (data) and behavior (code) defined by that class. A class is a logical entity (or prototype) and an object is a physical entity. The elements inside the class are known as **members**. Specifically, the data or variables inside the class are called as **member variables** or **instance variables** or **data members**. The code that operates on these data is referred to as **member methods** or **methods** (In C++, we term this as member function). The method operating on data will define the behavior and interface of a class.

Another purpose of the class is to hide the information from outside manipulation. Class uses **public** and **private** interfaces. The members declared as private can only be accessed by the members of that class, whereas, the public members can be accessed from outside the class. **Inheritance** allows us to have code re-usability. It is a process by which one object can acquire the properties of another object. It supports the concept of hierarchical classification. For example, consider a large group of animals having few of the abstract attributes like size, intelligence, skeletal structure etc. and having behavioral aspects like eating, breathing etc. Mammals have all the properties of Animals and also have their own specific features like



type of teeth, mammary glands etc. that make them different from Reptiles. Similarly, Cats and Dogs have all the characteristics of mammals, yet with few features which are unique for themselves. Though Doberman, German-shepherd, Labrador etc. have the features of Dog class, they have their own unique individuality. This concept can be depicted using following figure.

**Figure** Example of Inheritance

If we apply the above concept for programming, it can be easily understood that a code written is reusable. Thus, in this mechanism, it is possible for one object to be a specific instance of a more general case. Using inheritance, an object need only define those qualities that make it a unique object within its class. It can inherit its general attributes from its parent. Hence, through inheritance, we can achieve *generalization-specialization* concept. The top-most parent (or base class or **super class**) class is the generalized class and the bottom-most child (or derived class or **subclass**) class is a more specialized class with specific characteristics.

Inheritance interacts with encapsulation as well. If a given class encapsulates some attributes, then any subclass will have the same attributes *plus* any that it adds as part of its specialization. This is a key concept that lets object-oriented programs grow in complexity linearly rather than geometrically. A new subclass inherits all of the attributes of all of its ancestors. It does not have unpredictable interactions with the majority of the rest of the code in the system.

**Polymorphism** can be thought of as *one interface, multiple methods*. It is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. Consider an example of performing stack operation on three different types of data viz. integer, floating-point and characters. In a non-object oriented programming, we write functions with different names for push and pop operations though the logic is same for all the data types. But in Java, the same function names can be used with data types of the parameters being different.

### Java Development Kit (JDK)- Byte code

- *Bytecodes* are a set of instructions that look a lot like machine code, but are not specific to any one processor
- Platform-independence doesn't stop at the source level, however. Java binary files are also platform-independent and can run on multiple platforms without the need to recompile the source. Java binary files are actually in a form called bytecodes.

### Introduction to Java – **JDK Tools**

Tool in JDK	Purpose
<b>appletviewer</b>	This tool helps us to execute a special type of Java program known as applets.
<b>java</b>	This tool is the Java <b>interpreter</b> . It interprets the bytecode into machine code and then executes it
<b>javac</b>	This tool is Java <b>compiler</b> . It compiles the Java source code into the bytecode, which can be understood by JVM.
<b>javadoc</b>	This tool is used to create documentation in HTML.
<b>javah</b>	This tool produces C header files for use with a special type of Java methods, known as native methods.

### A First Simple Program

Here, we will discuss the working of a Java program by taking an example –

#### Program Illustration of First Java Program

```
class Prg1
{
    public static void main(String args[ ])
    {
        System.out.println("Hello World!!!");
    }
}
```

Save this program as **Prg1.java**. A java program source code is a text file containing one or more class definitions is called as **compilation unit** and the extension of this file name should be **.java**.

To compile above program, use the following statement in the command prompt –

**javac Prg1.java**

Now, the **javac** compiler creates a file **Prg1.class** containing bytecode version of the program, which can be understandable by JVM. To run the program, we have to use Java application launcher called **java**. That is, use the command –

**java Prg1**

The output of the program will now be displayed as –

Hello World!!!

**Note:** When java source code is compiled, each class in that file will be put into separate output file having the same name as of the respective class and with the extension of **.class**.

To run a java code, we need a class file containing *main()* function (Though, we can write java program without *main()*, for the time-being you assume that we need a *main()* function!!!). Hence, it is a tradition to give the name of the java source code file as the name of the class containing *main()* function.

Let us have closer look at the terminologies used in the above program now –

<b>class</b>	is the keyword to declare a class.
<b>Prg1</b>	is the name of the class. You can use any valid identifier for a class name.
<b>main()</b>	is name of the method from which the program execution starts.
<b>public</b>	is a keyword indicating the access specifier of the method. The <i>public</i> members can be accessed from outside the class in which they have been declared. The <i>main()</i> function must be declared as <i>public</i> as it needs to be called from outside the class.
<b>static</b>	The keyword <i>static</i> allows <i>main()</i> to be called without having to instantiate a particular instance of the class. This is necessary since <i>main()</i> is called by the Java Virtual Machine before any objects are made.
<b>void</b>	indicates that <i>main()</i> method is not returning anything.
<b>String args[ ]</b>	The <i>main()</i> method takes an array of <i>String</i> objects as a command-line argument.
<b>System</b>	is a predefined class (present in java.lang package) which gives access to the system. It contains pre-defined methods and fields, which provides facilities like standard input, output, etc.
<b>out</b>	is a static final (means not inheritable) field (ie, variable)in System class which is of the type <i>PrintStream</i> (a built-in class, contains methods to print the different data values). Static fields and methods must be accessed by using the class name, so we need to use <i>System.out</i> .

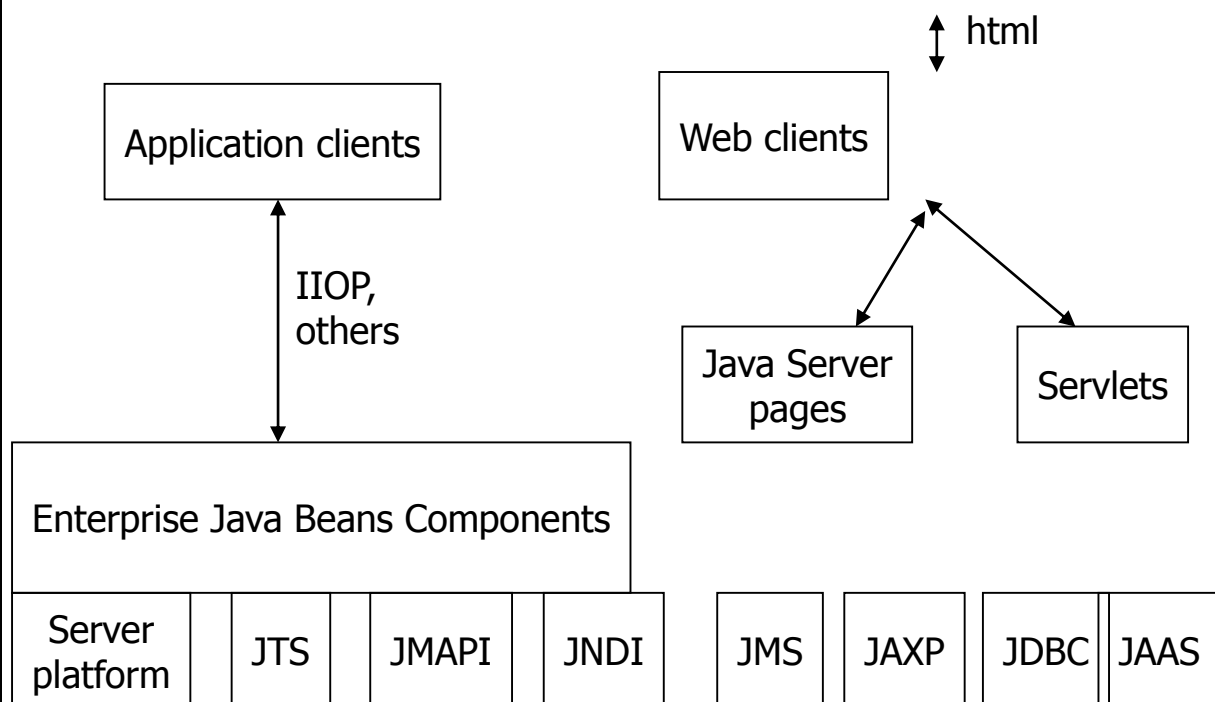
**println is a public method in *PrintStream* class to print the data values. After printing the data.**

### Advanced Java Programming Fundamentals

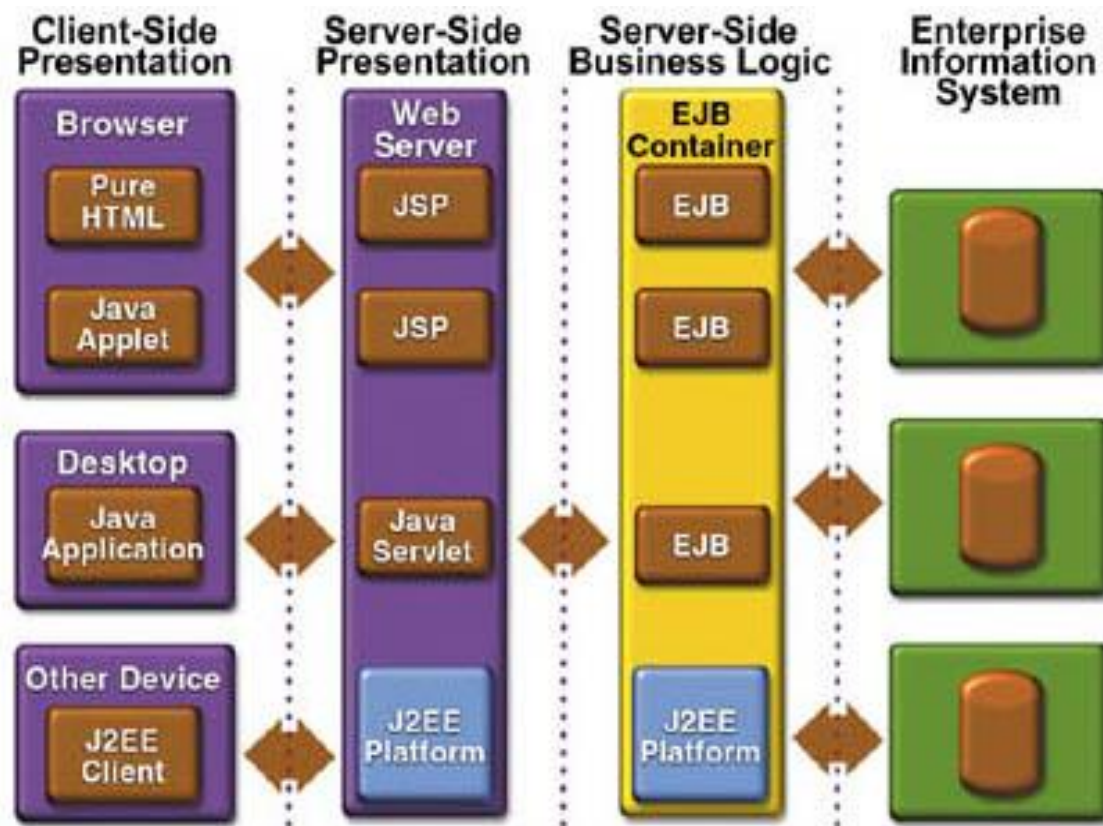
#### J2EE TECHNOLOGIES:

- Core technology: Container infrastructure, language and environment support
- XML technology
  - The Java API for XML Processing (JAXP)
  - The Java API for XML-based RPC (JAX-RPC)
  - SOAP with Attachments API for Java (SAAJ)
  - The Java API for XML Registries (JAXR)
- Web Technology
  - Java Servlets
  - JavaServer Pages
  - JavaServer Pages Standard Tag Library
- Enterprise Java Bean (EJB) technology
  - Session beans
  - Entity beans
    - Enterprise JavaBeans Query Language
  - Message-driven beans
- Platform services
  - Security
  - Transactions
  - Resources
  - Connectors
  - Java Message Service

### J2EE ARCHITECTURE:



# Enterprise Application Model



## J2EE Clients:

- **Web Clients**
  - Dynamic web pages with HTML, rendered by web browsers.
  - Can include applets.
  - Communicates with server typically using HTTP.
- **Application clients**
  - User interface using GUI components such as Swing and AWT.
  - Directly accesses the business logic tier.

## Web-tier Components:

- Client can communicate with the business tier either directly or through servlets or JSP that are located in the web-tier.
- Web-tier can help in pre-processing and allows distribution of the functionality.
- Servlets are special classes to realize the request-response model (get, post of HTTP).
- JSP is a developer-friendly wrapper over the servlet classes.

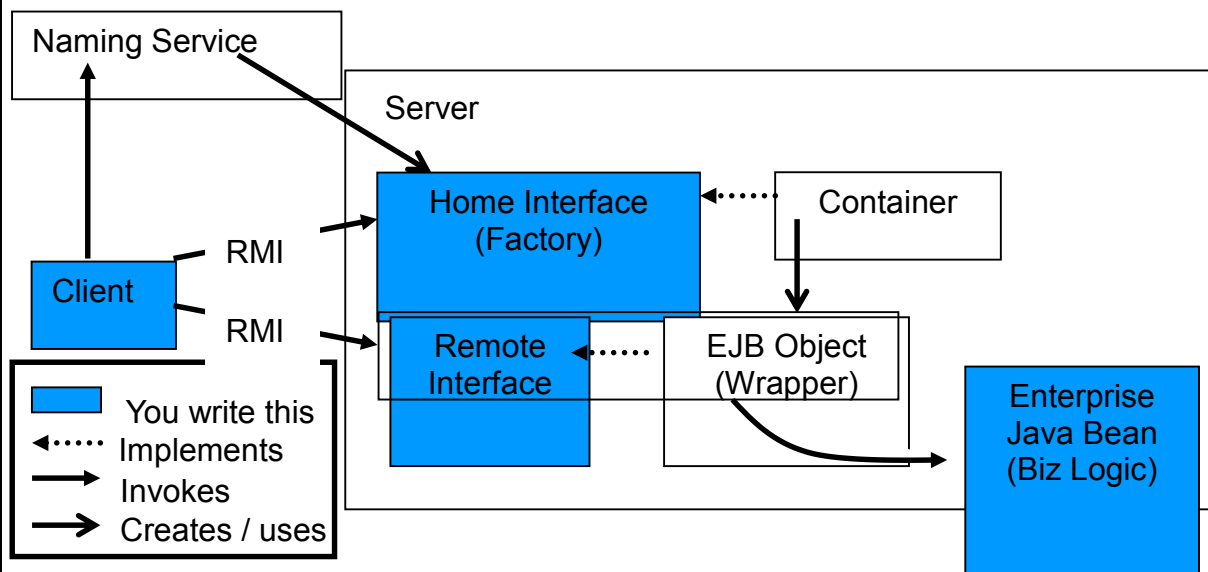
## Business-tier Components:

- This is defined by the logic that pertains to the (business) application that is being developed.
- Enterprise Java Beans (EJB) can be used to implement this tier.

- This tier receives the data from the web-tier and processes the data and sends it to the EIS-tier and takes the data from the EIS and sends it to the web-tier.

**Enterprise-Information System (EIS) tier:**

- In general this corresponds to the database ( relational database) and other information management system.
- The other information management systems may include Enterprise Resource Planning (ERP) and legacy system connected through open database connectivity.

**EJB Architecture**

### **3. Index of Lab Concepts**

#### **Program List - Basic Java Programs**

##### **1. Demonstration of Command Line Based Java Programming – Small Finance Application**

- i. Declaration of variables and Initialization,
- ii. Conditional and Iterative Statements - If, else, for, while
- iii. Strings,
- iv. Arrays and
- v. Getting user input through Java Stream Classes.

##### **2. Demonstration of Java Methods, Scope and Recursion –**

- i. Method parameters
- ii. Method overloading
- iii. Scope
- iv. Basic OOPS concepts
  - a. Class
  - b. Object
  - c. Constructors
  - d. Modifiers

##### **3. Demonstration of OOPs concepts**

- i. Inheritance – Single, multiple, multilevel
- ii. polymorphism
- iii. Inner Classes,
- iv. Abstraction,
- v. Interface
- vi. Enums.

##### **4. Demonstration of the following**

- i. Java Exceptions
- ii. Threads
- iii. RegEx
- iv. String Handling

#### **Advanced Java Programs**

- 1. Demonstration of JDBC Client-Server Program using JavaServlets / JavaServerPages. (Enquiry Form)
- 2. Demonstration of Cookies and Session Handling Using JavaServlets / JavaServerPages. (2-Page Website)
- 3. Demonstration of Java Beans – Student Markscard.
- 4. Demonstration of Entity Java Bean includes Session Bean – Student Registration Form.



**1. Program Statement:****Demonstration of Command Line Based Java Programming – Small Finance Application**

- i. Declaration of variables and Initialization,
- ii. Conditional and Iterative Statements - If, else, for, while
- iii. Strings,
- iv. Arrays and
- v. Getting user input through Java Stream Classes.

**Solution:**

```
import java.util.Scanner;
public class SmallFinance {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int termination = 0;
        int size = 0;
        double[] accountBalances = new double[250];
        String[] accountNames = new String[250];
        while(termination != -1){
            System.out.println("Bank Admin Menu");
            System.out.println("Please Enter a Menu Option");
            System.out.println("(1): Add Customer to Bank");
            System.out.println("(2): Change Customer Name");
            System.out.println("(3): Check Account Balance");
            System.out.println("(4): Modify Account Balance");
            System.out.println("(5): Summary of All Accounts");
            System.out.println("(-1): Quit");

            int userInput = input.nextInt();
            if(userInput == 1){
                System.out.println("Bank Add Customer Menu");
                System.out.println("Please Enter an account balance");
                double balance = input.nextDouble();
                accountBalances[size] = balance;
                System.out.println("Please enter an account holder name: ");
                input.nextLine();
                String name = input.nextLine();
                accountNames[size] = name;
                System.out.println("Customer's ID is :" + size);
                size = size + 1;
            }
            else if(userInput == 2){
                System.out.println("Bank Change Name Menu");
                System.out.println("Please Enter a customer ID to change their name");
                int index = input.nextInt();
                System.out.println("What is the customer's new name?");
```

```
        input.nextLine();
        accountNames[index] = input.nextLine();
    }
    else if(userInput == 3){
        System.out.println("Bank Check Balance Menu");
        System.out.println("Please Enter a customer ID to check their balance");
        int index = input.nextInt();
        double balance = accountBalances[index];
        System.out.println("This customer has $" + balance + " in their account");
    }
    else if(userInput == 4){
        System.out.println("Bank Modify Balance Menu");
        System.out.println("Please Enter a customer ID to check their balance");
        int index = input.nextInt();
        System.out.println("What is the customer's new account balance");
        accountBalances[index] = input.nextDouble();
    }
    else if(userInput == 5){
        System.out.println("Bank All Customer Summary Menu");
        double total = 0;
        for(int i = 0; i < size; i++){
            total = total + accountBalances[i];
        }
        System.out.println(accountNames[i] + " has $" + accountBalances[i] + " in their
account");
    }
    System.out.println("In total, there is $" + total + " in the bank");
    }
    else if(userInput == -1){
        System.exit(-1);
    }
    else{
        System.out.println("Error: Invalid input entered");
    }
}

}
```

**Sample Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac SmallFinance.java
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java SmallFinance
```

```
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
1
Bank Add Customer Menu
Please Enter an account balance
5000
Please enter an account holder name:
Shivakumara T
Customer's ID is :1
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
2
Bank Change Name Menu
Please Enter a customer ID to change their name
1
What is the customer's new name?
SKT
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
3
Bank Check Balance Menu
Please Enter a customer ID to check their balance
1
This customer has $5000.0 in their account
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
1
Bank Add Customer Menu
Please Enter an account balance
5380
Please enter an account holder name:
Ramu
Customer's ID is :2
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
5
Bank All Customer Summary Menu
null has $0.0 in their account
SKI has $8000.0 in their account
Ramu has $5380.0 in their account
In total, there is $13380.0 in the bank
Bank Admin Menu
Please Enter a Menu Option
(1): Add Customer to Bank
(2): Change Customer Name
(3): Check Account Balance
(4): Modify Account Balance
(5): Summary of All Accounts
(-1): Quit
```

Activate Windows  
Go to PC settings to activate Windows.

**2. Program Statement:****Demonstration of Java Methods, Scope and Recursion – Student Marksheet**

- i. Method parameters
- ii. Method overloading
- iii. Scope
- iv. Basic OOPS concepts
  - a. Class
  - b. Object
  - c. Constructors
  - d. Modifiers

**Solution:**

```
import java.util.Scanner;
class Marksheet
{
    private String usn;
    String sname;
    protected char grade;
    protected double m1,m2,m3;
    public double total;

    // Default Constructor
    public Marksheet()
    {
    }

    //Parametrized Constructor
    public Marksheet(String usn, String sname, char grade, double m1, double m2, double
m3, double total)
    {
        this.usn=usn;
        this.sname=sname;
        this.grade=grade;
        this.m1=m1;
        this.m2=m2;
        this.m3=m3;
        this.total=total;
    }

    //Getter and Setter methods
    public String getUSN()
    { return this.usn;
    }

    public void setUSN(String usn)
    {
        this.usn=usn;
    }

    public void setGRADE(String grade)
```

```
{
this.grade=grade;
}
public void setM1(double m1, double m2, double m3)
{
this.m1 = m1;
this.m2 = m2;
this.m3 = m3
}

//Method
public void display()
{
System.out.println("USN:"+usn);
System.out.println("NAME:"+sname);
System.out.println("M1:"+m1+",M2:"+m2+",M3:"+m3);
System.out.println("Total: "+(m1+m2+m3));
}
//Method Overloading
public void display(char msg)
{
System.out.println("Grade of the Student"+msg);
}
}

class Markcal
{
public static void main(String[] args)
{
Marksheet arjun = new Marksheet();
System.out.println("Default Constructor Called\n\n");
arjun.display();
Marksheet rambha = new Marksheet("1BY23MC001","AIO Rambha",'W',1,4,3,8);
rambha.display();
Marksheet sophia = new Marksheet();
sophia.setUSN("1BY23MC002");
sophia.sname="SOPHIA ROBO";
sophia.m1=sophia.m2=sophia.m3=90;
sophia.grade='s';
sophia.total = sophia.m1 + sophia.m2 + sophia.m3;
sophia.display();
sophia.display('S');
} }
```

**3. Program Statement:****Demonstration of OOPs concepts**

- i. Inheritance – Single, multiple, multilevel
- ii. polymorphism
- iii. Inner Classes,
- iv. Abstraction,
- v. Interface
- vi. Enums.

**Solution:**

- i. Inheritance – Single Level Inheritance (is-A & has-A relationship)

**Is-A relationship****//Parent-Son - Single Level Inheritance**

```
class Parent
{
String name;
int age;
Parent(String name, int age)
{
this.name=name;
this.age = age;
}
public void disp()
{
System.out.println("Parent Details are");
System.out.println("Name:\t"+name);
System.out.println("Age:\t"+age);
}
}
class Son extends Parent
{
String gender;
Son(String name, int age, String gender)
{
super(name,age);
this.gender = gender;
}

public void disp()
{
System.out.println("Son Details are below:");
System.out.println("Name:\t"+name);
System.out.println("Age:\t"+age);
System.out.println("Gender:\t"+gender);
}
}
class SingleLevelInheritance
{
```

```
public static void main(String[ ] args)
{
    Son s1 = new Son("Ajith",06,"Male");
    s1.disp();
}
}
```

### **Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac SingleLevelInheritance.java
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java SingleLevelInheritance
```

Son Details are below:

Name: Ajith

Age: 6

Gender: Male

Activate Windows

Go to PC settings to activate Windows.

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

### **Has-A relationship**

```
public class Address {
    String city,state,country;
```

```
    public Address(String city, String state, String country) {
        this.city = city;
        this.state = state;
        this.country = country;
    }
}
```

```
public class Emp {
    int id;
    String name;
    Address address;
```

```
    public Emp(int id, String name,Address address) {
        this.id = id;
        this.name = name;
        this.address=address;
    }
}
```

```
    void display(){
        System.out.println(id+" "+name);
        System.out.println(address.city+" "+address.state+" "+address.country);
    }
}
```

```
    public static void main(String[] args) {
        Address address1=new Address("gzb","UP","india");
        Address address2=new Address("gno","UP","india");
    }
}
```

```

Emp e=new Emp(111,"varun",address1);
Emp e2=new Emp(112,"arun",address2);

e.display();
e2.display();

}
}

```

**Output:**

```

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac Emp.java
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java Emp
111 varun
gzb UP india
112 arun
gno UP india
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>

```

**ii. Inheritance – Multilevel Inheritance**

//Multilevel Inheritance

```

class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class MultilevelInheritance{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}

```

**Output:**

```

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac MultilevelInheritance.java
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java MultilevelInheritance
weeping...
barking...
eating...
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>

```

**Inheritance – Multiple Inheritance through Interface****iii. Polymorphism**

//Runtime polymorphism

```

class Bank{
float getRateOfInterest(){return 0;}
}
class SBI extends Bank{

```



```
float getRateOfInterest(){return 8.4f;}
}
class ICICI extends Bank{
float getRateOfInterest(){return 7.3f;}
}
class AXIS extends Bank{
float getRateOfInterest(){return 9.7f;}
}
class Polymorphism{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
```

**Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac Polymorphism.java

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java Polymorphism
SBI Rate of Interest: 8.4
ICICI Rate of Interest: 7.3
AXIS Rate of Interest: 9.7

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

**iv. Inner Classes**

```
class Outerclass
{
public void callInner()
{
Innerclass inr = new Innerclass();
System.out.println("Now you are in outer class");
inr.callfromouterclass();
}
class Innerclass
{
public void callfromouterclass()
{
System.out.println("Now you are in Inner class");
}
}
}
class OuterInner
{
public static void main(String[] args)
{
Outerclass outer = new Outerclass();
outer.callInner();
}
```

```
}  
}
```

**Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac OuterInner.java  
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java OuterInner  
Now you are in outer class  
Now you are in Inner class  
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

**v. Abstraction**

//Example of an abstract class that has abstract and non-abstract methods

```
abstract class Bike{  
    Bike(){System.out.println("bike is created");}  
    abstract void run();  
    void changeGear(){System.out.println("gear changed");}  
}
```

//Creating a Child class which inherits Abstract class

```
class Honda extends Bike{  
    void run(){System.out.println("running safely..");}  
}
```

//Creating a Test class which calls abstract and non-abstract methods

```
class TestAbstraction2{  
    public static void main(String args[]){  
        Bike obj = new Honda();  
        obj.run();  
        obj.changeGear();  
    }  
}
```

**Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac TestAbstraction2.  
java  
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java TestAbstraction2  
bike is created  
running safely..  
gear changed  
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

**vi. Enums**

```
class EnumExample1{  
    //defining enum within class  
    public enum Season { WINTER, SPRING, SUMMER, FALL }  
    //creating the main method  
    public static void main(String[] args) {  
        //printing all enum
```

```
for (Season s : Season.values()){  
    System.out.println(s);  
}  
System.out.println("Value of WINTER is: "+Season.valueOf("WINTER"));  
System.out.println("Index of WINTER is:  
"+Season.valueOf("WINTER").ordinal());  
System.out.println("Index of SUMMER is:  
"+Season.valueOf("SUMMER").ordinal());  
  
}}
```

**Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac EnumExample1.java
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java EnumExample1  
WINTER  
SPRING  
SUMMER  
FALL  
Value of WINTER is: WINTER  
Index of WINTER is: 0  
Index of SUMMER is: 2
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

**4. Program Statement:****Demonstration of the following**

- i. Java Exceptions
- ii. Threads
- iii. RegEx
- iv. String Handling

**Solution:****i. Java Exceptions**

class LessBalExe extends Exception

```
{
    float b;
    public LessBalExe(float amt)
    {
        b=amt;
        System.out.println("Withdrawing amount is invalid"+b);
    }
}
class Account
{
    float bal;
    public Account(float b)
    {
        if(b<500.00f)
        {
            System.out.println("No Sufficient balance\n");
        }
        else
        {
            bal=b;
        }
    }
    public void deposit(float amt)
    {
        bal=bal+amt;
        System.out.println("The Balance is="+bal);
    }
    public void withdraw(float amt) throws LessBalExe
    {
        float x;
        x=bal-amt;
        if(x>=500.00f)
        {
            bal=bal-amt;
            System.out.println("Remaining balance is="+bal);
        }
        else
        {
            throw new LessBalExe(amt);
        }
    }
}
```

```
        throw new LessBalExe(x);
    }
}
class Transact
{
    public static void main(String args[])
    {
        try
        {
            Account ob1=new Account(500.00f);
            ob1.deposit(100.00f);
            ob1.withdraw(100.00f);
            Account ob2=new Account(1500.00f);
            ob2.deposit(200.00f);
            ob2.withdraw(1000.00f);
        }
        catch(Exception e)
        {
        }
    }
}
```

**Output:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac LessBalException.
java
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java LessBalException
The Balance is=600.0
Remaining balance is=500.0
The Balance is=1700.0
Remaining balance is=700.0
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac LessBalException.
java
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java LessBalException
The Balance is=600.0
Withdrawing amount is invalid100.0
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

**ii. Threads**

```
class Shop
{
    private static int materials;
    private boolean available=false;
    public synchronized void put(int value)
    {
        while(available==true)
        {
            try
            {
```

```
        wait();
    } catch (InterruptedException ie)
    {
        ie.printStackTrace();
    }
}
materials=value;
available=true;
notifyAll();
System.out.println("Produced val " +materials);
}
public synchronized void get()
{
    while(available==false)
    {
        try
        {
            wait();
        } catch (InterruptedException ie)
        {
        }
    }
    available=false;
    notifyAll();
    System.out.println("Consumed val " +materials);
}
}

class Producer extends Thread
{
    private Shop s;

    public Producer(Shop c)
    {
        s=c;
    }
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            s.put(i);
        }
    }
}

class Consumer extends Thread
{
    private Shop s;
```

```
public Consumer(Shop c)
{
    s=c;
}
public void run()
{
    int value=0;
    for(int i=0;i<10;i++)
    {
        s.get();
    }
}
class ProdCons
{
    public static void main(String args[])
    {
        Shop c=new Shop();
        Producer p1=new Producer(c);
        Consumer c1=new Consumer(c);
        p1.start();
        c1.start();
    }
}
```

**Output:****iii. Regular Expression (RegEx)**

```
//Java Regex Finder Example
import java.util.regex.Pattern;
import java.util.Scanner;
import java.util.regex.Matcher;
public class RegexExample8{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        while (true) {
            System.out.println("Enter regex pattern:");
            Pattern pattern = Pattern.compile(sc.nextLine());
            System.out.println("Enter text:");
            Matcher matcher = pattern.matcher(sc.nextLine());
            boolean found = false;
            while (matcher.find()) {
                System.out.println("I found the text "+matcher.group()+" starting at index "+
                    matcher.start()+" and ending at index "+matcher.end());
                found = true;
            }
            if(!found){

```

```
        System.out.println("No match found.");
    }
}
}
}

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac RegexExample8.java
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java RegexExample8
Enter regex pattern:
is
Enter text:
Java Programming is the very best programming to develop web based applications
I found the text is starting at index 17 and ending at index 19
Enter regex pattern:
java
Enter text:
java is kava
I found the text java starting at index 0 and ending at index 4
Enter regex pattern:

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac ProdCons.java
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java ProdCons
Produced val 0
Consumed val 0
Produced val 1
Consumed val 1
Produced val 2
Consumed val 2
Produced val 3
Consumed val 3
Produced val 4
Consumed val 4
Produced val 5
Consumed val 5
Produced val 6
Consumed val 6
Produced val 7
Consumed val 7
Produced val 8
Consumed val 8
Produced val 9
Consumed val 9

D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

#### iv. String Handling

```
class StringHandling{
    public static void main(String []args)
    {
        String s1="BMSITM";
        String s2="Bmsitm";
        String s3="BmS";
        boolean x=s1.equals(s2);
        System.out.println("Compare s1 and s2:"+x);
        System.out.println("Character at given position is:"+s1.charAt(5));
        System.out.println(s1.concat(" the author"));
        System.out.println(s1.length());
        System.out.println(s1.toLowerCase());
        System.out.println(s1.toUpperCase());
        System.out.println(s1.indexOf('a'));
        System.out.println(s1.substring(0,4));
        System.out.println(s1.substring(4));
    }
}
```



```
}  
}
```

**Ouptut:**

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>javac StringHandling.java
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>java StringHandling
```

```
Compare s1 and s2:false
```

```
Character at given position is:M
```

```
BMSITM the author
```

```
6
```

```
bmsitm
```

```
BMSITM
```

```
-1
```

```
BMSI
```

```
TM
```

```
D:\Academic\June-2023 to Dec 2023\Java Practice Programs>
```

**5. Program Statement: (Advanced Java)**

**Demonstration of JDBC Client-Server Program using JavaServlets / JavaServerPages. (Enquiry Form). The below code is sample of CRUD operations, accordingly fit your client-server program.**

**Solution:**

```
Studentdb.java
package studentdb;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.sql.*;

public class Studentdb {
    private Connection con=null;
    private Statement st=null;
    private ResultSet rs=null;
    private PreparedStatement pst=null;

    Studentdb() throws ClassNotFoundException, SQLException, IOException
    {
        Class.forName("com.mysql.jdbc.Driver");
        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/mysql", "root",
"root");

        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br =new BufferedReader(isr);
        System.out.println("MENU\n1.CREATE\n2.INSERT\nUPDATE\nDELETE\n");
        for(;;)
        {
            System.out.println("Enter your choice\n");
            int ch=Integer.parseInt(br.readLine());
            switch(ch)
            {
                case 1:
                    System.out.println("Enter your query to create a table\n");
                    String q1=br.readLine();
                    st=con.createStatement();
                    st.execute(q1);
                    System.out.println("Query executed successfully");
                    break;
                case 2:
                    System.out.println("Enter your query to Update the table\n");
                    String q2=br.readLine();
                    pst=con.prepareStatement(q2);
                    pst.executeUpdate(q2);
                    System.out.println("Query executed successfully");
                    break;
```

```
        case 3:
            System.out.println("Enter your query for selection");
            String q3=br.readLine();
            st=con.createStatement();
            rs=st.executeQuery(q3);
            DisplayRS(rs);
            System.out.println("Query executed successfully");
            break;
        default:
            System.exit(0);

    }
}
}

public static void main(String[] args) throws ClassNotFoundException,
SQLException, IOException
{
    final Studentdb s1=new Studentdb();
    System.out.println("MENU\n1.INSERT\n2.UPDATE\n3.DELETE");
}

private void DisplayRS(ResultSet rs) throws SQLException {
    ResultSet rs1=rs;
    ResultSetMetaData rsm=rs1.getMetaData();
    int col=rsm.getColumnCount();
    int count=1;
    boolean b=rs1.next();
    if(!b)
    {
        System.out.println("Data not found");
    }
    else
    do
    {
        System.out.println("Record" +(count++)+"=>");
        for(int j=0;j<col;j++)
            System.out.println(rs1.getString(j+1)+"\t");
        System.out.println("");
    } while(rs1.next()); } }
```

**OUTPUT:**

run:

MENU

1.CREATE

2.INSERT

UPDATE

DELETE

Enter your choice

2

```
Enter your query to Update the table
insert into tab values('1by13mca50','hisham');
Query executed successfully
Enter your choice
3
Enter your query for selection
select * from tab
Record1=>
1by22mca32
sudha
Record2=>
1by22mca50
hisham
Query executed successfully
Enter your choice 4
```

**6. Program Statement: (Advanced Java)****Demonstration of Cookies and Session Handling Using JavaServlets /  
JavaServerPages. (2-Page Website)****Solution:**

```
// HttpSession Program [login,logout,profilepage]
package yourpack;
```

LoginSessionServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class SessionServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String s1 = request.getParameter("uname");
        String s2 = request.getParameter("pwd");
        PrintWriter out = response.getWriter();
        try {
            if(s1.equals("skt") && s2.equals("skt"))
            {
                out.println("Welcome :"+s1);
                HttpSession ss1 = request.getSession();
                ss1.setAttribute("name", s1);
                request.getRequestDispatcher("link.html").include(request, response);
            }
            else
            {
                out.println("Sorry, username or password error!");
                request.getRequestDispatcher("Login.html").include(request, response);
            }
        } finally {
            out.close();
        }
    }
}
```

Login.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
```

```
<a href="Login.html"> Login </a> |  
<a href="LogoutServlet"> Logout </a> |  
<a href="ProfileServlet"> Profile </a> <hr>  
<form name="loginform" action="SessionServlet" method="post">  
  <h1 align="center"> User Login </h1>  
  Username:<input type="text" name="uname" value="" size="25" />  
  Password:<input type="password" name="pwd" value="" size="25" />  
  <input type="submit" value="Login" name="login" />  
</form>  
</body>
```

#### Link.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
  <head>  
    <title></title>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
  </head>  
  <body>  
    <a href="Login.html"> Login </a> |  
    <a href="LogoutServlet"> Logout </a> |  
    <a href="ProfileServlet"> Profile </a> <hr>  
  </body>  
</html>
```

#### LogoutSessionServlet.java

```
package yourpack;  
import java.io.IOException;  
import java.io.PrintWriter;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
public class LogoutServlet extends HttpServlet {  
  protected void doPost(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException {  
    response.setContentType("text/html;charset=UTF-8");  
    PrintWriter out = response.getWriter();  
    try {  
      request.getRequestDispatcher("link.html").include(request, response);  
      HttpSession sess = request.getSession();  
      sess.invalidate();  
      out.println("You are successfully logged out");  
    } finally {  
      out.close();  
    }  
  }  
}
```

#### ProfileServlet.java

```
package yourpack;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class ProfileServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            request.getRequestDispatcher("link.html").include(request,response);
            HttpSession sess2=request.getSession(false);
            if(sess2!=null)
            {
                String name=(String)sess2.getAttribute("name");
                out.println("Hello,"+name+" <br/>Welcome to Short-Profile");
                request.getRequestDispatcher("profilepage.html").include(request, response);
            }
        } else
        {
            out.println("please login first");
            request.getRequestDispatcher("Login.html").include(request, response);
        }
        } finally {
            out.close();
        }
    }
}
```

### Profilepage.html

Short-Resume of **Shivakumara T**

#### 1. Educational Information

MCA - 2003

MTech - 2013

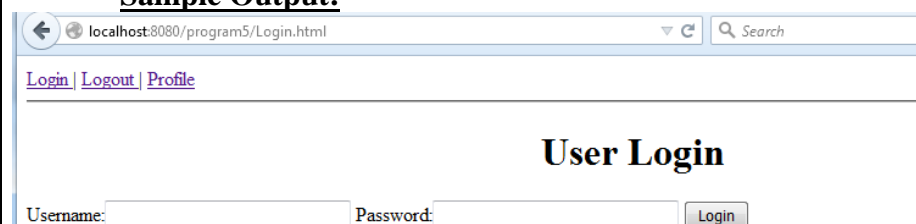
PhD - Pursuing

#### 2. Skills

Java, C#, PHP, JavaScript, html5

SQL, Android, Python, R-Programming

### Sample Output:

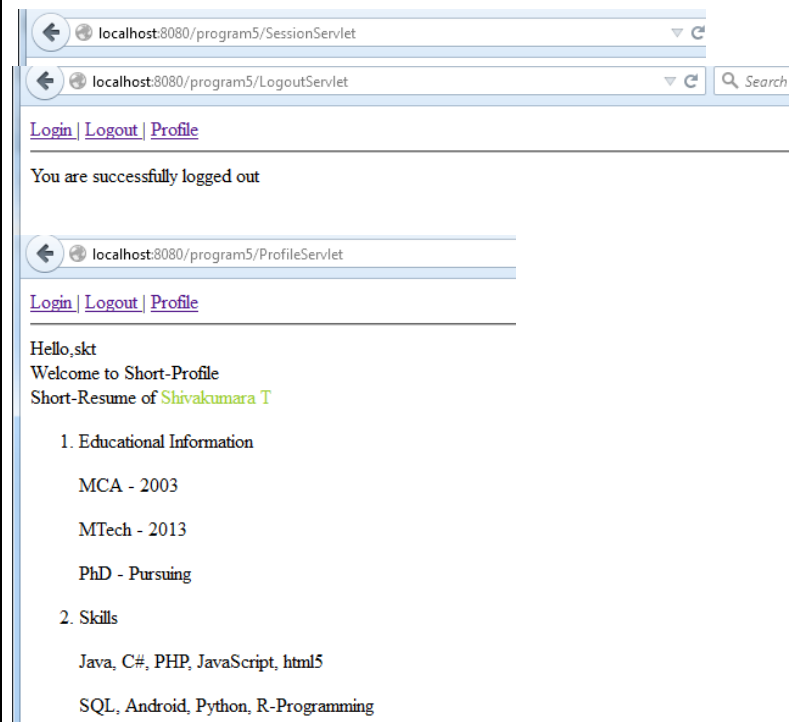


localhost:8080/program5/Login.html

[Login](#) | [Logout](#) | [Profile](#)

## User Login

Username:  Password:



## B. Write a Java Servlet program using Cookies to remember user preferences

### Front.html

```
<!DOCTYPE html>
<html>
  <head>
    <title> Details </title>
  </head>
  <body>
    <form action="helloform" method="get"><br><br>
      <b>First Name : </b><input type="text" name="first_name"> <br>
      <b>Last Name : </b><input type="text" name="last_name"> <br>
      <b>Phone no : </b><input type="text" name="phone"> <br>
      <b>Address : </b><input type="text" name="add"> <br>
      <b>E-mail ID : </b><input type="text" name="mail"> <br>
      <input type="submit" value="submit">
    </body>
  </html>
```



```
Helloform.java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class helloform extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        Cookie firstName=new Cookie("first_name",request.getParameter("first_name"));
        Cookie lastName=new Cookie("last_name",request.getParameter("last_name"));
        firstName.setMaxAge(60);
        lastName.setMaxAge(60);
        response.addCookie(firstName);
        response.addCookie(lastName);
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String title="Setting cookie Example";
        out.println("<html>\n"+
            "<head>" +
            "<title>"+title+"</title>" +
            "</head>" +
            "<body>" +
            "<h1>"+title+"</h1>\n"+
            "<ul>\n"+
            "<li> <b>First Name</b> : "+request.getParameter("first_name")+"\n"+
            "<li> <b>Last Name</b> : "+request.getParameter("last_name")+"\n"+
            "<li> <b>Phone no</b> : "+request.getParameter("phone")+"\n"+
            "<li> <b>Address</b> : "+request.getParameter("add")+"\n"+
            "<li> <b>E-mail ID</b> : "+request.getParameter("mail")+"\n"+
            "</body></html>");
    }
}
```

### **Readcookie.java**

```
// Import required java libraries
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
// Extend HttpServlet class
public class ReadCookies extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException
    {
        Cookie cookie = null;
        Cookie[] cookies = null;
        // Get an array of Cookies associated with this domain
        cookies = request.getCookies();
        // Set response content type
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Reading Cookies Example";
        String docType =
            "<!doctype html public \"-//w3c//dtd html 4.0 \" +
            \"transitional//en\">\n";
        out.println(docType +
            "<html>\n" +
            "<head><title>" + title + "</title></head>\n" +
            "<body bgcolor=\"#f0f0f0\">\n" );
        if( cookies != null ){
            out.println("<h2> Found Cookies Name and Value</h2>");
            for (int i = 0; i < cookies.length; i++){
                cookie = cookies[i];
                out.print("Name : " + cookie.getName( ) + ", ");
                out.print("Value: " + cookie.getValue( )+" <br/>");
            } else{
                out.println(
                    "<h2>No cookies founds</h2>");
            }
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```

Details

localhost:8080/programs/foont.html

First Name : sudha  
Last Name : shree  
Phone no : 85538888  
Address : moore road  
E-mail ID : shashree92@gmail.com  
submit

Setting cookie Example

localhost:8080/programs/foont.html?first\_name=sudha&last\_name=shree&phone=85538888&add=moore+road&email=sudhasree92%40gmail.com

### Setting cookie Example

- First Name : sudha
- Last Name : shree
- Phone no : 85538888
- Address : moore road
- E-mail ID : sudhasree92@gmail.com

Setting cookie Example    Reading Cookies Example

localhost:8080/programs/ReadCookies

### Found Cookies Name and Value

Name : first\_name, Value: sudha  
Name : last\_name, Value: shree

Setting cookie Example    Reading Cookies Example

localhost:8080/programs/ReadCookies

### No cookies found

**7. Program Statement: (Advanced Java)****Demonstration of Java Beans – Student Markscard.****Solution:****StudentInput.html**

```
<html>
  <head>
    <title> User login Page</title>
  </head>
  <body bgcolor="gold">
    <form action="Firstpage.jsp" method="Post">
      <center>
        <h1>Welcome to Student Information System </h1> </center>
        Please Enter Student Name
        <input type="text" name="sname"><br>
        <br>Please enter Student USN
        <input type="text" name="usn" > <br>
        <br>Please Enter Student marks1
        <input type="text" name="m1"><br>
        <br>Please Enter Student marks2
        <input type="text" name="m2"><br>
        <br>Please Enter Student marks3
        <input type="text" name="m3"><br>
        <br> <input type="submit" value="submit">
      </form>
    </body>
  </html>
```

**StudentBean.java**

```
package Bean;
import java.io.Serializable;
public class StudentBean implements Serializable
{
  private String sname;
  private String usn;
  private String m1;
  private String m2;
  private String m3;
  public void setsname(String sname)
  { this.sname=sname; }
  public void setusn(String usn)
  { this.usn=usn; }
  public void setm1(String m1)
  { this.m1=m1; }
  public void setm2(String m2)
  { this.m2=m2; }
  public void setm3(String m3)
  { this.m3=m3; }
```

```
public String getsname()
{ return sname; }
public String getusn()
{ return usn; }
public String getm1()
{ return m1; }
    public String getm2()
{ return m2; }
    public String getm3()
{ return m3; }
}
```

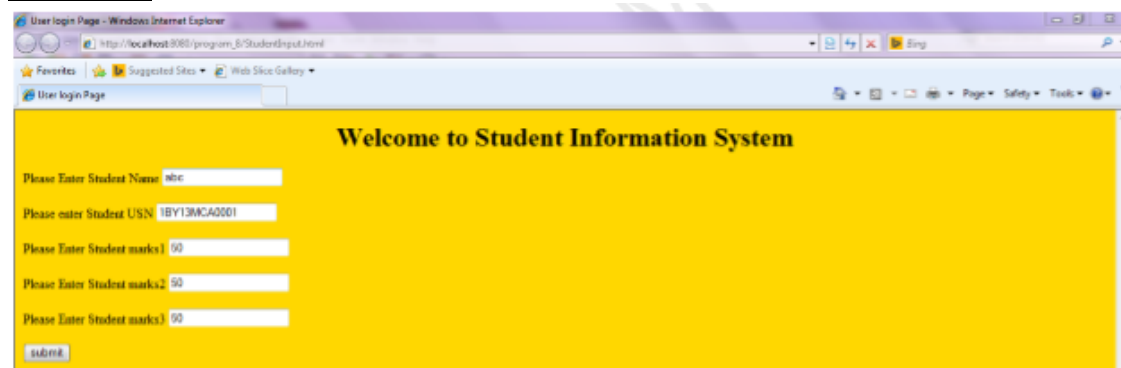
**Firstpage.jsp**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<jsp:useBean id="StudentBean" scope="request" class="Bean.StudentBean">
    <jsp:setProperty name="StudentBean" property="*" />
</jsp:useBean>
<html>
    <body bgcolor="gold"> <br>
        <jsp:forward page="Secondpage.jsp" />
    </body>
</html>
```

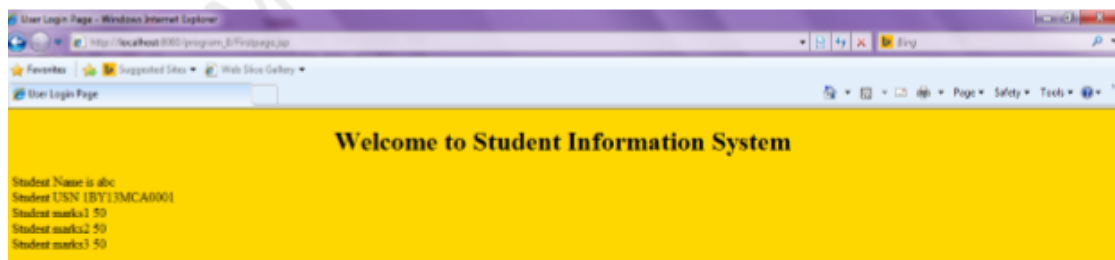
**Secondpage.jsp**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>User Login Page</title>
    </head>
    <body bgcolor="gold">

        <center>
            <h1>Welcome to Student Information System </h1> </center>
            <jsp:useBean id="StudentBean" scope="request" class="Bean.StudentBean" />
            Student Name is
            <jsp:getProperty name="StudentBean" property="sname" /> <br>
            Student USN
            <jsp:getProperty name="StudentBean" property="usn" /> <br>
            Student marks1
            <jsp:getProperty name="StudentBean" property="m1" /> <br>
            Student marks2
            <jsp:getProperty name="StudentBean" property="m2" /> <br>
            Student marks3
            <jsp:getProperty name="StudentBean" property="m3" /> <br>
        </body>
</html>
```

**OUTPUT:**

A screenshot of a web browser window titled "User Login Page - Windows Internet Explorer". The address bar shows "http://localhost:8080/program\_5/StudentInput.html". The page has a yellow background and the title "Welcome to Student Information System". On the left side, there are five input fields with labels: "Please Enter Student Name", "Please enter Student USN", "Please Enter Student marks1", "Please Enter Student marks2", and "Please Enter Student marks3". The first field contains "abc", the second contains "IBY13MCA0001", and the third, fourth, and fifth fields contain "50". A "submit" button is at the bottom left.



A screenshot of the same web browser window after the form has been submitted. The page title and background remain the same. The input fields are now populated with the submitted data: "Student Name is abc", "Student USN IBY13MCA0001", "Student marks1 50", "Student marks2 50", and "Student marks3 50".

**8. Program Statement:**

**Demonstration of Entity Java Bean includes Session Bean – Student Registration Form.**

**Solution:****index.jsp**

```
<html>
  <head>
  </head>
  <body>
    <form action="stservlet" method="post" >
      Enter name : <input type="text" name="nam">
      Enter age : <input type="text" name="age">
      <input type="submit">
    </form>
  </body>
</html>
```

**stservlet.java**

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import p2.StudFacadeLocal;

public class stservlet extends HttpServlet {
    @EJB
    private StudFacadeLocal studFacade;
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String name=request.getParameter("nam");
            String ag=request.getParameter("age");
            int age=Integer.parseInt(ag);
            studFacade.addvalue(name,age);
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet stservlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1> Inserted!! "+ "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
```

```
    } finally {  
        out.close();  
    }  
}  
}
```

### **AbstractFacade.java**

```
package p2;  
import java.util.List;  
import javax.persistence.EntityManager;  
public abstract class AbstractFacade<T> {  
    private Class<T> entityClass;  
    public AbstractFacade(Class<T> entityClass) {  
        this.entityClass = entityClass;  
    }  
    protected abstract EntityManager getEntityManager();  
    public void create(T entity) {  
        getEntityManager().persist(entity);  
    }  
    public void edit(T entity) {  
        getEntityManager().merge(entity);  
    }  
    public void remove(T entity) {  
        getEntityManager().remove(getEntityManager().merge(entity));  
    }  
    public T find(Object id) {  
        return getEntityManager().find(entityClass, id);  
    }  
    public List<T> findAll() {  
        javax.persistence.criteria.CriteriaQuery cq =  
getEntityManager().getCriteriaBuilder().createQuery();  
        cq.select(cq.from(entityClass));  
        return getEntityManager().createQuery(cq).getResultList();  
    }  
    public List<T> findRange(int[] range) {  
        javax.persistence.criteria.CriteriaQuery cq =  
getEntityManager().getCriteriaBuilder().createQuery();  
        cq.select(cq.from(entityClass));  
        javax.persistence.Query q = getEntityManager().createQuery(cq);  
        q.setMaxResults(range[1] - range[0]);  
        q.setFirstResult(range[0]);  
        return q.getResultList();  
    }  
    public int count() {  
        javax.persistence.criteria.CriteriaQuery cq =  
getEntityManager().getCriteriaBuilder().createQuery();  
        javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
```



```
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

### **Stud.java**

```
package p2;
import java.io.Serializable;
import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
@Entity
@Table(name = "STUD", catalog = "", schema = "APP")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Stud.findAll", query = "SELECT s FROM Stud s"),
    @NamedQuery(name = "Stud.findByName", query = "SELECT s FROM Stud s
WHERE s.name = :name"),
    @NamedQuery(name = "Stud.findByAge", query = "SELECT s FROM Stud s
WHERE s.age = :age"))
public class Stud implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 10)
    @Column(name = "NAME", nullable = false, length = 10)
    private String name;
    @Column(name = "AGE")
    private Integer age;
    public Stud() {
    }
    public Stud(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
```

```
        this.age = age;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (name != null ? name.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Stud)) {
            return false;
        }
        Stud other = (Stud) object;
        if ((this.name == null && other.name != null) || (this.name != null &&
!this.name.equals(other.name))) {
            return false;
        }
        return true;
    }
    @Override
    public String toString() {
        return "p2.Stud[ name=" + name + " ]";
    }
}
```

### **StudFacade.java**

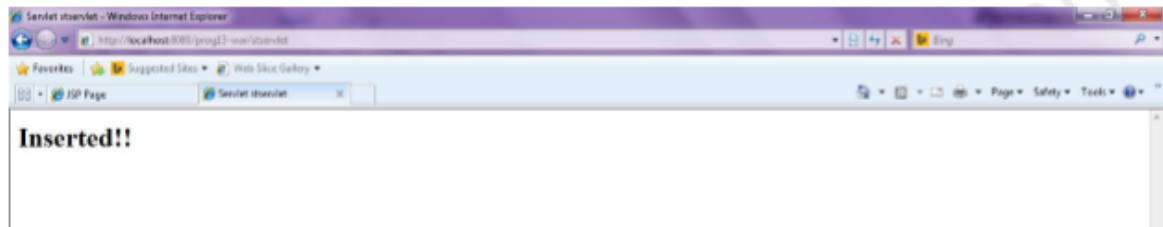
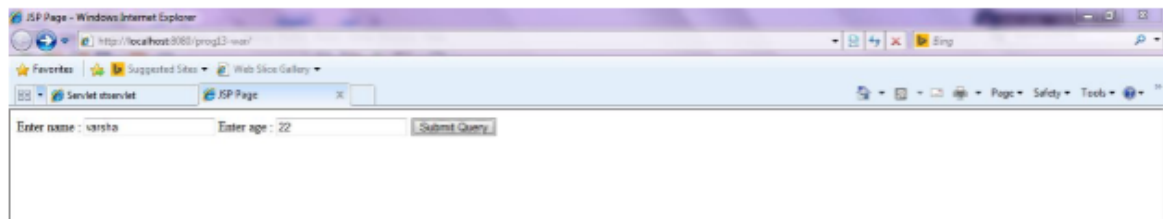
```
package p2;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
@Stateless
public class StudFacade extends AbstractFacade<Stud> implements StudFacadeLocal {
    @PersistenceContext(unitName = "prog13-ejbPU")
    private EntityManager em;
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    @Override
    public void persist(Object object)
    {
        em.persist(object);
    }
    public StudFacade() {
        super(Stud.class);
    }
}
```

```
}  
@Override  
public void addvalue(String name, int age) {  
    Stud obj=new Stud();  
    obj.setName(name);  
    obj.setAge(age);  
    persist(obj);  
}  
}
```

**StudFacadeLocal.java**

```
package p2;  
import java.util.List;  
import javax.ejb.Local;  
@Local  
public interface StudFacadeLocal {  
    void create(Stud stud);  
    void edit(Stud stud);  
    void remove(Stud stud);  
    Stud find(Object id);  
    List<Stud> findAll();  
    List<Stud> findRange(int[] range);  
    int count();  
    void addvalue(String name, int age);  
    public void persist(java.lang.Object object);  
}
```

**OUTPUT:**



select \* from APP.STUD \*

Page Size: 2

#	NAME	AGE
1	sudha	22
2	pri	22
3	varsha	22
4	abc	22

**Proposed Viva Questions**

1. Draw a neat diagram for J2EE architecture
2. Define JDBC, & how many JDBC driver types?
3. List Data types supported by JDBC
4. Which package is used for JDBC program?
5. Define class.forName() method?
6. Write the JDBC Connection code in java?
7. Define Statement Object?
8. How many types of Statement Objects?
9. Define Statement, PreparedStatement and CallableStatement?
10. What do you mean by ResultSet?
11. Define execute(), executeQuery(), and executeUpdate() methods?
12. Define Meta Data in JDBC?
13. How to create index in JDBC program?
14. Write the example query for Existence Test, Membership Test, and Any Test?
15. How Servlets different from CGI?
16. Write the life cycle of Servlet?
17. Which package mandatory for Servlet program?
18. Give the example for deployment descriptor of Servlet?
19. What the HTTP header carries the information along with client request?
20. What is Status code represents? And Define 400, 401, 403, 404, 405, 415, 500, 501, 503, & 505?
21. Define cookie & its types?
22. Write example code for cookie?
23. Define Session?
24. What are different Tracking mechanisms?
25. Define JSP?
26. How JSP different from Servlet?
27. What are Standard JSP tags?
28. Write the example code for JSP include, import, sql, and custom tags?
29. Define Web Server, Application Server?
30. Define JavaBeans?
31. What are the two design patterns in JavaBeans?
32. Define Introspection?
33. Define Customizer?
34. Which package is mandatory for JavaBeans?
35. Write a snippet code to add colour as property to JavaBean?
36. Write the hierarchy of classes for JavaBeans?
37. Which package is mandatory for EJB?
38. Explain bound and constraint properties of JavaBeans?
39. Define EJB?
40. Write the architecture diagram of EJB?
41. Write the entity java bean deployment descriptor?
42. How many environment elements of an EJB?
43. What is the Transaction attributes in EJB?
44. Define Session Java Bean?
45. What are the types of Session Beans?

46. Define Entity Java Bean?
47. What are the types of Entity Java Bean?
48. Define Message Driven Bean?
49. How Session Bean is different from Entity Bean and Message-Driven Bean?
50. Define the components of EJB jar file?

Reference:

- [1] <https://www.javatpoint.com/java>
- [2] <https://www.educative.io/answers/string-handling-functions-using-java>

Self-Assessment Quiz

- [1]: <https://www.interviewBit.com/java-mcq/amp/>
- [2] <https://www.javatpoint.com/java-mcq>
- [3] <https://www.sanfoundry.com/java-qquestions-answers-freshers-experienced/>