Advanced Java Programming LAB

(CIE-306P)

Faculty Name : Dr. Ashish Khanna

Name : Amit Singhal

Enrollment No. : 11614802722

Semester : 6

Group : AIML-II-B

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

"To attain global excellence through **education, innovation, research,** and **work ethics** with the commitment to **serve humanity."**

MISSION

M1. To promote diversification by adopting advancement in science, technology, management, and allied discipline through continuous learning

M2. To foster **moral values** in students and equip them for developing sustainable solutions to serve both national and global needs in society and industry.

M3. To **digitize educational resources and process** for enhanced teaching and effective learning.

M4. To cultivate an **environment** supporting **incubation, product development, technology transfer, capacity building and entrepreneurship**.

M5. To encourage **faculty-student networking with alumni, industry, institutions,** and other **stakeholders** for collective engagement.

Department of Computer Science and Engineering

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

"To attain global excellence through education, innovation, research, and work ethics in the field of Computer Science and engineering with the commitment to serve humanity."

MISSION

M1. To lead in the advancement of computer science and engineering through internationally recognized research and education.

M2. To prepare students for full and ethical participation in a diverse society and encourage lifelong learning.

M3. To foster development of problem solving and communication skills as an integral component of the profession.

M4. To impart knowledge, skills and cultivate an environment supporting incubation, product development, technology transfer, capacity building and entrepreneurship in the field of computer science and engineering.

M5. To encourage faculty, student's networking with alumni, industry, institutions, and other stakeholders for collective engagement.

ADVANCED JAVA PROGRAMMING LAB

PRACTICAL RECORD

Name : **Amit Singhal** Paper Code: **CIE - 306P**

Enrollment No. : **11614802722**

Branch : **CSE-I**

Semester/Group : **6 – AIML II - B**

a) LIST OF EXPERIMENTS (prescribed by G.G.S.I.P.U)

S.No.	Experiment Name	Date of Perf.	M R1	A R2	R R3	K R4	S R5	Total Marks	Signature

Experiment - 1

AIM :: Write a java program to demonstrate the concept of multiple inheritance.

CODE ::

```java
package lab2;

interface Animal {
    void sound();
}

interface Movable {
    void move();
}

// A class that implements both Animal and Movable interfaces
public class MultipleInheritanceExample implements Animal, Movable {

    public void sound() {
        System.out.println("The animal makes a sound");
    }

    public void move() {
        System.out.println("The animal moves");
    }

    public static void main(String[] args) {
        MultipleInheritanceExample obj = new MultipleInheritanceExample();

        obj.sound(); // Calls sound() from Animal
        obj.move(); // Calls move() from Movable
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit-11614802722> javac lab2\MultipleInheritanceExample.java
PS C:\Users\mait\Downloads\amit-11614802722> java lab2.MultipleInheritanceExample
The animal makes a sound
The animal moves
```

Experiment - 2

AIM :: Write a java program to demonstrate the concept of labeled loops.

CODE ::

```java
package lab2;

public class LabeledLoopExample {
    public static void main(String[] args) {

        outerLoop: for (int i = 1; i <= 3; i++) {
            System.out.println("Outer loop iteration " + i);

            innerLoop: for (int j = 1; j <= 3; j++) {
                System.out.println("  Inner loop iteration " + j);

                if (j == 2) {
                    System.out.println(" Skipping the rest of the inner
                                        loop (j == 2)");
                    continue outerLoop;
                }
            }
        }
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit-11614802722> javac lab2\LabeledLoopExample.java
PS C:\Users\mait\Downloads\amit-11614802722> java lab2.LabeledLoopExample
Outer loop iteration 1
  Inner loop iteration 1
  Inner loop iteration 2
    Skipping the rest of the inner loop (j == 2)
Outer loop iteration 2
  Inner loop iteration 1
  Inner loop iteration 2
    Skipping the rest of the inner loop (j == 2)
Outer loop iteration 3
  Inner loop iteration 1
  Inner loop iteration 2
    Skipping the rest of the inner loop (j == 2)
```

Experiment - 3

AIM :: Write a java program to show the concept of all types of inheritances among interfaces.

CODE ::

```
package Code;

// Single Inheritance among Interfaces
interface Animal {
    void sound();
}

interface Dog extends Animal {
    void breed();
}

class Labrador implements Dog {
    public void sound() {
        System.out.println("Labrador barks");
    }

    public void breed() {
        System.out.println("Breed: Labrador");
    }
}

// Hierarchical Inheritance among Interfaces
interface Vehicle {
    void start();
}

interface Car extends Vehicle {
    void speed();
}

interface Bike extends Vehicle {
    void mileage();
}

class Sedan implements Car {
    public void start() {
        System.out.println("Sedan starts");
    }

    public void speed() {
        System.out.println("Sedan has a speed of 120 km/h");
    }
}
```

```java
class SportsBike implements Bike {
  public void start() {
    System.out.println("Sports Bike starts");
  }

  public void mileage() {
    System.out.println("Sports Bike has a mileage of 30 km/l");
  }
}

public class InterfaceInheritance{
    public static void main(String[] args) {
        // Single Inheritance
        Labrador labrador = new Labrador();
        labrador.sound();
        labrador.breed();

        // Hierarchical Inheritance
        Sedan sedan = new Sedan();
        sedan.start();
        sedan.speed();

        SportsBike bike = new SportsBike();
        bike.start();
        bike.mileage();
    }
}
```

Output ::

```
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/Sem 6/LabWork/Advanced Java$ javac Code/InterfaceInheritance.java
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/Sem 6/LabWork/Advanced Java$ java Code.InterfaceInheritance
Labrador barks
Breed: Labrador
Sedan starts
Sedan has a speed of 120 km/h
Sports Bike starts
Sports Bike has a mileage of 30 km/l
```

Experiment - 4

AIM :: Write a java program to demonstrate the concept of multi-level inheritance.

CODE ::

```
class Appliance {

    void powerOn() {
        System.out.println("Appliance is powered on");
    }
}

class WashingMachine extends Appliance {

    void wash() {
        System.out.println("Washing Machine is washing clothes");
    }
}

class SmartWashingMachine extends WashingMachine {

    void connectToWifi() {
        System.out.println("Smart Washing Machine is connected to WiFi");
    }
}

class AdvancedWasher extends SmartWashingMachine {

    @Override
    void powerOn() {
        System.out.println("Advanced Washer is powered on");
    }

    @Override
    void wash() {
```

```
        System.out.println("Advanced Washer is washing clothes");
    }

    @Override
    void connectToWifi() {
        System.out.println("Advanced Washer is connected to WiFi");
    }
}

public class MultiLevelInheritance {
    public static void main(String[] args) {
        AdvancedWasher washer = new AdvancedWasher();
        washer.powerOn();
        washer.wash();
        washer.connectToWifi();
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit> javac .\MultiLevelInheritance.java
PS C:\Users\mait\Downloads\amit> java MultiLevelInheritance
Advanced Washer is powered on
Advanced Washer is washing clothes
Advanced Washer is connected to WiFi
```

Experiment - 5

AIM :: Write a java program to demonstrate the concept of dynamic method dispatch.

CODE ::

```
package Code;

// Superclass
class Animal {
    // Method in superclass
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

// Subclass 1
class Dog extends Animal {
    // Overriding sound() method
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

// Subclass 2
class Cat extends Animal {
    // Overriding sound() method
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class DynamicMethodDispatch {
```

```
    public static void main(String[] args) {

        // creating objects of subclasses
        Animal animal1 = new Dog(); // Reference type: Animal, Object type:
Dog
        Animal animal2 = new Cat(); // Reference type: Animal, Object type:
Cat


        // the method that is called depends on the actual object type
        animal1.sound(); // Output: Dog barks (since animal1 points to a Dog
object)
        animal2.sound(); // Output: Cat meows (since animal2 points to a Cat
object)
    }
}
```

Output ::

```
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/Sem 6/LabWork/Advanced Java$ javac Code/DynamicMethodDispatch.java
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/Sem 6/LabWork/Advanced Java$ java Code.DynamicMethodDispatch
Dog barks
Cat meows
```

Experiment - 6

AIM :: Write a java program to show the use of command line arguments.

CODE ::

```
public class CommandLineArgs {

    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("<name> <age> not defined in command line
arguments");
            return;
        }

        String name = args[0];
        String ageString = args[1];

        try {
            int age = Integer.parseInt(ageString);
            System.out.println("Hello, " + name + "! You are " + age + "
years old.");
        } catch (NumberFormatException e) {
            System.out.println("Error: The second argument should be a valid
number (age).");
        }
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit> javac .\CommandLineArgs.java
PS C:\Users\mait\Downloads\amit> java CommandLineArgs Amit 21
Hello, Amit! You are 21 years old.
```

Experiment - 7

AIM :: Write a java program to create a class, declare variables and methods and call those methods using the object of the class.

CODE ::

```
public class ObjectCall {
    String name;
    int age;

    public void greet() {
        System.out.println("Hello, my name is " + name + " and I am " + age +
" years old.");
    }

    public int getAge() {
        return age;
    }

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

    public static void main(String[] args) {
        ObjectCall person1 = new ObjectCall();

        person1.name = "John";
        person1.age = 25;

        person1.greet();
        System.out.println("Age: " + person1.getAge());

        person1.setAge(30);
        System.out.println("Updated Age: " + person1.getAge());
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit> javac .\ObjectCall.java
PS C:\Users\mait\Downloads\amit> java ObjectCall
Hello, my name is John and I am 25 years old.
Age: 25
Updated Age: 30
```

Experiment - 8

AIM :: Write a java program to show the concept of Single Inheritance.

CODE ::

```
class Animal {

    String name;

    public Animal(String name) {
        this.name = name;
    }

    public void makeSound() {
        System.out.println("The animal makes a sound.");
    }

    public String getName() {
        return name;
    }
}

class Dog extends Animal {

    public Dog(String name) {
        super(name);  // Calls the constructor of the parent class
    }

    @Override
    public void makeSound() {
        System.out.println("The dog barks.");
    }

    public void fetch() {
        System.out.println("The dog is fetching the ball.");
    }
}

public class SingleInheritance {
    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy");

        System.out.println("Dog's name: " + myDog.getName());
        myDog.makeSound();
        myDog.fetch();
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit> javac .\SingleInheritance.java
PS C:\Users\mait\Downloads\amit> java SingleInheritance
Dog's name: Buddy
The dog barks.
The dog is fetching the ball.
```

Experiment - 9

AIM :: Write a java program to show the concept of Constructors.

CODE ::

```
class Car {
    String model;
    int year;
    String color;

    public Car() {
        this.model = "Unknown";
        this.year = 0;
        this.color = "Unknown";
        System.out.println("Default constructor called");
    }

    public Car(String model, int year, String color) {
        // Initialize with the provided values
        this.model = model;
        this.year = year;
        this.color = color;
        System.out.println("Parameterized constructor called");
    }

    public void displayDetails() {
        System.out.println("Car Model: " + model);
        System.out.println("Car Year: " + year);
        System.out.println("Car Color: " + color);
    }
}

public class Constructor {
    public static void main(String[] args) {
        Car car1 = new Car();
        car1.displayDetails();

        System.out.println();

        Car car2 = new Car("Tesla Model S", 2023, "Red");
        car2.displayDetails();
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit> javac .\Constructor.java
PS C:\Users\mait\Downloads\amit> java Constructor
Default constructor called
Car Model: Unknown
Car Year: 0
Car Color: Unknown

Parameterized constructor called
Car Model: Tesla Model S
Car Year: 2023
Car Color: Red
```

Experiment - 10

AIM :: Write a java program to show the concept of Method Overloading.

CODE ::

```
class Calculator {

    public int add(int a, int b) {
        return a + b;
    }

    public int add(int a, int b, int c) {
        return a + b + c;
    }

    public double add(double a, double b) {
        return a + b;
    }

    public double add(int a, double b) {
        return a + b;
    }
}

public class MethodOverloading {

    public static void main(String[] args) {
        Calculator calc = new Calculator();

        System.out.println("Sum of 10 and 20: " + calc.add(10, 20));

        System.out.println("Sum of 10, 20, and 30: " + calc.add(10, 20, 30));

        System.out.println("Sum of 10.5 and 20.5: " + calc.add(10.5, 20.5));

        System.out.println("Sum of 10 and 20.5: " + calc.add(10, 20.5));
    }
}
```

OUTPUT ::

```
PS C:\Users\mait\Downloads\amit> javac .\MethodOverloading.java
PS C:\Users\mait\Downloads\amit> java MethodOverloading
Sum of 10 and 20: 30
Sum of 10, 20, and 30: 60
Sum of 10.5 and 20.5: 31.0
Sum of 10 and 20.5: 30.5
```

Experiment - 11

AIM :: WAP where a method with the same name is declared in both the parent and child classes.

- The method should have the same parameter list in one case and a different parameter list in another case.
- Also, provide a description of which category this program falls into.

Code ::

```
class Parent {
    void show(int a) {
        System.out.println("Parent class method with 1 parameter: " + a);
    }

    void show(int a, int b) {
        System.out.println("Parent class method with 2 parameters: " + a + ", " + b);
    }
}

class Child extends Parent {
    @Override
    void show(int a) {
        System.out.println("Child class method (Overriding) with 1 parameter: " + a);
    }

    void show(String msg) {
        System.out.println("Child class method (Overloading) with string parameter: "
+ msg);
    }
}

public class q11_specialquestion {
    public static void main(String[] args) {
        Child obj = new Child();
        obj.show(116);
        obj.show("Hello");
        obj.show(116, 105);
    }
}
```

Output ::

```
Code$ javac q11_specialquestion.java
Code$ java q11_specialquestion
Child class method (Overriding) with 1 parameter: 116
Child class method (Overloading) with string parameter: Hello
Parent class method with 2 parameters: 116, 105
```

Experiment - 12

AIM :: WAP To implement for Loop & Methods Of Thread like join, wait, sleep and suspend.

Code ::

```
class MyThread extends Thread {
    private volatile boolean suspended = false; // Flag to manage suspension

    public void run() {
        try {
            System.out.println(Thread.currentThread().getName() + " started.");

            // Using sleep() to pause the thread for 1 second
            Thread.sleep(1000);
            System.out.println(Thread.currentThread().getName() + " slept for 1
second.");

            // Simulating wait() (but in a synchronized block)
            synchronized (this) {
                wait(1000); // Makes the current thread wait for 1 second
                System.out.println(Thread.currentThread().getName() + " resumed after
waiting.");
            }

            // Check if the thread is suspended and wait
            synchronized (this) {
                while (suspended) {
                    wait(); // Wait until resumed
                }
            }
            System.out.println(Thread.currentThread().getName() + " is running after
resume.");

        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }

    // Suspend method using a flag
    public void suspendThread() {
        suspended = true;
        System.out.println(Thread.currentThread().getName() + " is suspended.");
    }

    // Resume method to wake up the suspended thread
    public synchronized void resumeThread() {
        suspended = false;
        notify(); // Notify the waiting thread
        System.out.println(Thread.currentThread().getName() + " is resumed.");
    }
}

public class q12_BasicThreadMethods {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();

        // Start both threads using a for loop
        for (int i = 0; i < 2; i++) {
            if (i == 0) {
                t1.start();
```

```
            } else {
                t2.start();
            }
        }

        // Using join() to ensure the main thread waits for both threads to finish
        try {
            t1.join();
            t2.join();
            System.out.println("Both threads have finished.");
        } catch (InterruptedException e) {
            System.out.println(e);
        }

        // Suspending and resuming thread (demonstration)
        t1.suspendThread();
        try {
            Thread.sleep(2000); // Wait for 2 seconds before resuming
        } catch (InterruptedException e) {
            System.out.println(e);
        }
        t1.resumeThread();
    }
}
```

Output ::

```
Code$ javac q12_BasicThreadMethods.java
Code$ java q12_BasicThreadMethods
Thread-0 started.
Thread-1 started.
Thread-0 slept for 1 second.
Thread-1 slept for 1 second.
Thread-0 resumed after waiting.
Thread-0 is running after resume.
Thread-1 resumed after waiting.
Thread-1 is running after resume.
Both threads have finished.
main is suspended.
main is resumed.
```

Experiment - 13

AIM :: WAP to implement synchronization in threads and remove deadlock.

Code ::

```
class Resource {
    // Synchronize methods to avoid thread interference and ensure thread safety.
    public synchronized void methodA(Resource other) {
        System.out.println(Thread.currentThread().getName() + " is in methodA.");
        try {
            Thread.sleep(1000);  // Simulate work being done
            other.methodB(this);  // Call methodB of the other resource
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }

    public synchronized void methodB(Resource other) {
        System.out.println(Thread.currentThread().getName() + " is in methodB.");
        try {
            Thread.sleep(1000);  // Simulate work being done
            other.methodA(this);  // Call methodA of the other resource
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

public class q13_DeadlockPrevention {
    public static void main(String[] args) {
        Resource resource1 = new Resource();
        Resource resource2 = new Resource();

        // Thread 1: Locking in a consistent order to prevent deadlock
        Thread t1 = new Thread(() -> {
            synchronized (resource1) {  // First lock
                System.out.println(Thread.currentThread().getName() + " locked
resource1.");
                try { Thread.sleep(100); } catch (InterruptedException e) {}
                synchronized (resource2) {  // Then lock resource2
                    System.out.println(Thread.currentThread().getName() + " locked
resource2.");
                    resource1.methodA(resource2);  // Call methodA on resource2
                }
            }
        });

        // Thread 2: Locking in a consistent order to prevent deadlock
        Thread t2 = new Thread(() -> {
            synchronized (resource1) {  // First lock
                System.out.println(Thread.currentThread().getName() + " locked
resource1.");
                try { Thread.sleep(100); } catch (InterruptedException e) {}
                synchronized (resource2) {  // Then lock resource2
                    System.out.println(Thread.currentThread().getName() + " locked
resource2.");
                    resource1.methodB(resource2);  // Call methodB on resource2
                }
            }
        });
```

```
            t1.start();
            t2.start();
        }
    }
```

Output ::

```
Code$ javac q13_DeadlockPrevention.java
Code$ java q13_DeadlockPrevention
Thread-0 locked resource1.
Thread-0 locked resource2.
Thread-0 is in methodA.
Thread-0 is in methodB.
Thread-0 is in methodA.
Thread-0 is in methodB.
Thread-0 is in methodA.
Thread-0 is in methodB.
```

Experiment - 14

AIM :: WAP To implement Producer-Consumer problem in java.

Code ::

```
import java.util.LinkedList;
                    import java.util.Queue;

class SharedBuffer {
    private final Queue<Integer> buffer = new LinkedList<>();
    private final int MAX_CAPACITY = 5;

    // Method for Producer to add items to the buffer
    public synchronized void produce() throws InterruptedException {
        while (buffer.size() == MAX_CAPACITY) {
            System.out.println("Buffer is full. Producer is waiting...");
            wait(); // Wait if the buffer is full
        }
        int item = (int) (Math.random() * 100); // Generate a random item
        buffer.add(item);
        System.out.println("Produced: " + item);
        notify(); // Notify the consumer that the buffer is no longer empty
    }

    // Method for Consumer to consume items from the buffer
    public synchronized void consume() throws InterruptedException {
        while (buffer.isEmpty()) {
            System.out.println("Buffer is empty. Consumer is waiting...");
            wait(); // Wait if the buffer is empty
        }
        int item = buffer.poll();
        System.out.println("Consumed: " + item);
        notify(); // Notify the producer that the buffer is no longer full
    }
}

class Producer implements Runnable {
    private final SharedBuffer sharedBuffer;

    public Producer(SharedBuffer sharedBuffer) {
        this.sharedBuffer = sharedBuffer;
    }

    @Override
    public void run() {
        try {
            while (true) {
                sharedBuffer.produce();
                Thread.sleep(1000); // Simulate time taken to produce an item
            }
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

class Consumer implements Runnable {
    private final SharedBuffer sharedBuffer;

    public Consumer(SharedBuffer sharedBuffer) {
        this.sharedBuffer = sharedBuffer;
```

```
    }

    @Override
    public void run() {
        try {
            while (true) {
                sharedBuffer.consume();
                Thread.sleep(1500); // Simulate time taken to consume an item
            }
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

public class q14_ProducerConsumerProblem {
    public static void main(String[] args) {
        SharedBuffer sharedBuffer = new SharedBuffer();

        Thread producerThread = new Thread(new Producer(sharedBuffer));
        Thread consumerThread = new Thread(new Consumer(sharedBuffer));

        producerThread.start();
        consumerThread.start();
    }
}
```

Output ::

```
Code$ javac q14_ProducerConsumerProblem.java
Code$ java q14_ProducerConsumerProblem
Produced: 51
Consumed: 51
Produced: 65
Consumed: 65
Produced: 1
Consumed: 1
Produced: 18
Produced: 50
Consumed: 18
Produced: 3
Consumed: 50
Produced: 39
Produced: 60
Consumed: 3
```

Experiment - 15

AIM :: WAP To implement try-catch-finally block in java.

Code ::

```
public class q15_TryCatch {
    public static void main(String[] args) {
        int arr[] = { 66, 69, 122, 105, 80085 };
        int n = arr.length;

        try {
            System.out.println("Currently In Try Block");
            checkArrayIndex(arr, n);
        } catch (Exception e) {
            System.out.println();
            System.out.println("\nCurrently In Catch Block");
            System.out.println(e.getMessage());
        } finally {
            System.out.println();
            System.out.println("This is finally block, Program Has Ended");
        }
    }

    public static void checkArrayIndex(int[] arr, int n) throws
ArrayIndexOutOfBoundsException {
        for (int i = 0; i ≤ n; i++) {
            if (i ≥ arr.length) {
                throw new ArrayIndexOutOfBoundsException("Array index out of bounds:
" + i);
            }
            System.out.print(arr[i] + " ");
        }
    }
}
```

Output ::

```
Code$ javac q15_TryCatch.java
Code$ java q15_TryCatch
Currently In Try Block
66 69 122 105 80085

Currently In Catch Block
Array index out of bounds: 5

This is finally block, Program Has Ended
```

Experiment - 16

AIM :: WAP to implement a basic applet.

Code ::

1. **q16_BasicApplet.java**

```
import java.applet.Applet;
import java.awt.Graphics;

public class q16_BasicApplet extends Applet {
    // Override the paint method to display the message
    @Override
    public void paint(Graphics g) {
        // Display the message on the applet window
        g.drawString("Hi, This is Amit Singhal", 50, 60);
    }
}
```

2. **q16_BasicApplet.html**

```
<html>
    <body>
        <applet code="q16_BasicApplet.class" width="400" height="150"></applet>
    </body>
</html>
```

Output ::

```
Code$ javac q16_BasicApplet.java
Code$ java q16_BasicApplet
```

Experiment - 17

AIM :: WAP to implement a moving banner in java using applet and threads.

Code ::

1. q17_MovingBanner.java

```
import java.applet.Applet;
import java.awt.*;

public class q17_MovingBanner extends Applet implements Runnable {
  private String message = "Welcome to Java Programming!";
  private int xPos = 100; // Initial position of the text
  private int yPos = 50; // Y-coordinate of the text
  private Thread thread;

  @Override
  public void init() {
    thread = new Thread(this);
    thread.start();
  }

  @Override
  public void run() {
    while (true) {
      xPos -= 2; // Move the text to the left

      // If the text moves completely off screen, reset its position
      if (xPos < -getFontMetrics(getFont()).stringWidth(message)) {
        xPos = getWidth();
      }

      repaint(); // Redraw the applet

      try {
        Thread.sleep(50); // Control the speed of movement
      } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
      }
    }
  }
```

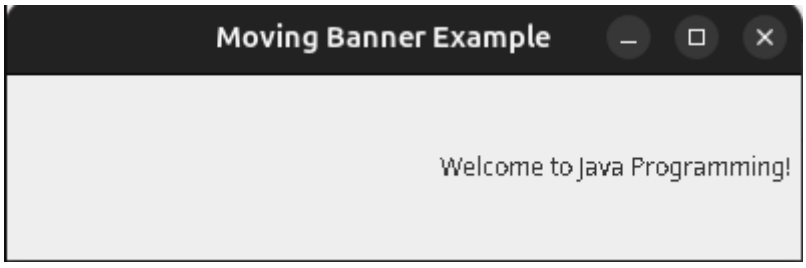
```
    @Override
    public void paint(Graphics g) {
      g.drawString(message, xPos, yPos);
    }

    @Override
    public void stop() {
      thread = null; // Stop the thread when applet is stopped
    }
  }
```

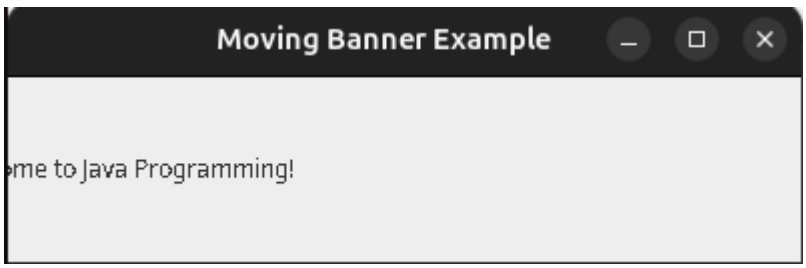
2. q17_MovingBanner.html

```
<html>
  <body>
    <applet code="q17_MovingBanner.class" width="400" height="150"></applet>
  </body>
</html>
```

Output ::

```
Code$ javac  q17_MovingBanner.java
Code$ java q17_MovingBanner
```





Experiment - 18

AIM :: WAP to call a frame using applet and awt.

Code ::

1. q18_AppletWithFrame.java

```
import java.applet.Applet;
import java.awt.*;

@SuppressWarnings("removal")
public class q18_AppletWithFrame extends Applet {

    @Override
    public void init() {
        // Create a Frame
        Frame myFrame = new Frame("Applet with Frame");

        // Create a Label in the Frame
        Label label = new Label("Hello, I am inside a Frame!");

        // Set layout and add label
        myFrame.setLayout(new FlowLayout());
        myFrame.add(label);

        // Set frame size and make it visible
        myFrame.setSize(300, 200);
        myFrame.setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        g.drawString("This is an Applet!", 50, 50);
    }
}
```

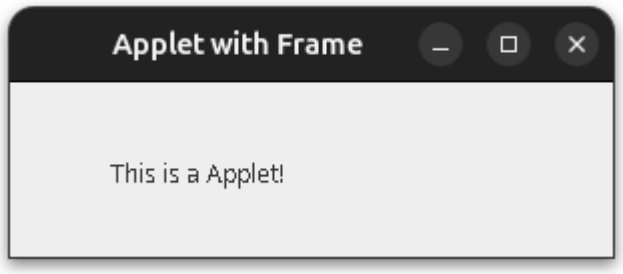
2. q18_AppletWithFrame.html

```
<html>
    <body>
        <applet code="q18_AppletWithFrame.class" width="400" height="250"></applet>
```

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

Output ::

```
Code$ javac  q18_AppletWithFrame.java
Code$ java q18_AppletWithFrame
```



Experiment - 19

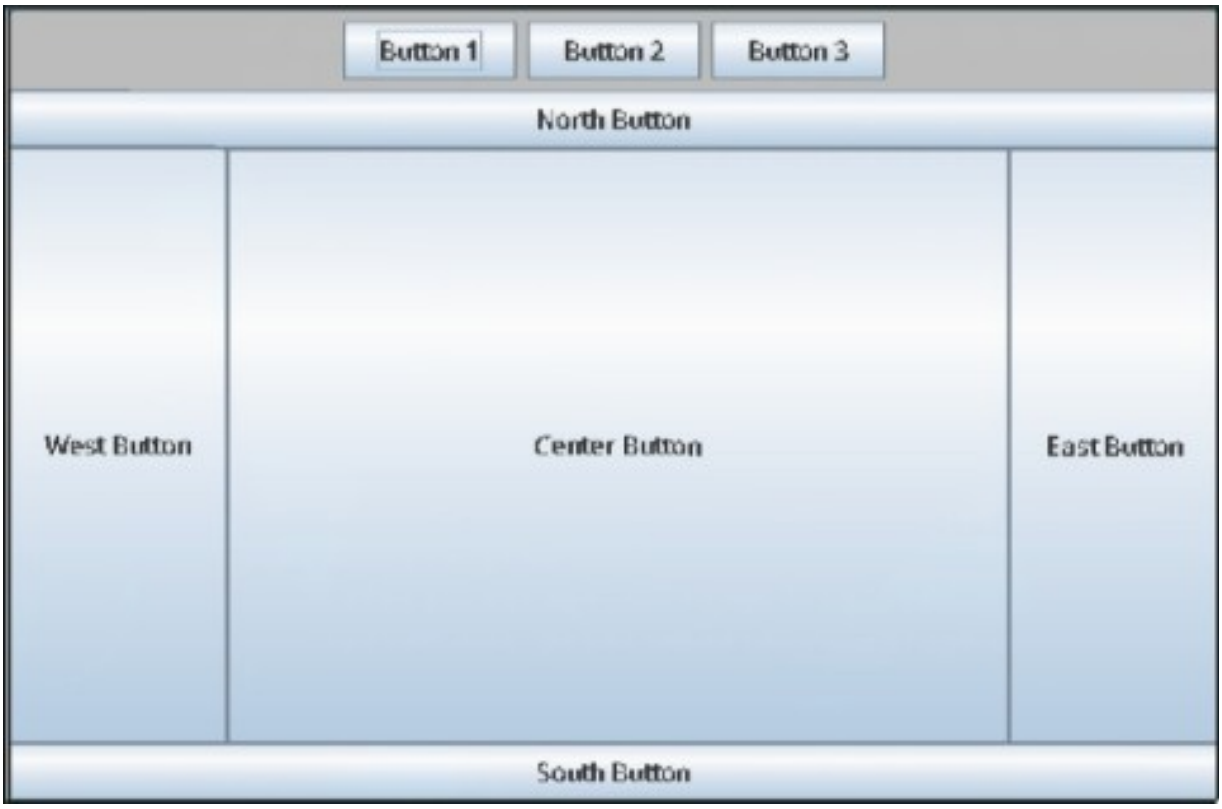
AIM :: Write a Java Program to showcase the use of Layout Manager

Code ::

LayoutManagersExample.java

```
import java.awt.*;
import javax.swing.*;

public class LayoutManagersExample {
    public static void main(String[] args) {
        // Create a new JFrame (main window)
        JFrame frame = new JFrame("Layout Manager Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 200); // Set the size of the window

        // Create a panel with FlowLayout
        JPanel flowPanel = new JPanel(new FlowLayout());
        flowPanel.setBackground(Color.LIGHT_GRAY); // Set background color for the panel
        flowPanel.add(new JButton("Button 1"));
        flowPanel.add(new JButton("Button 2"));
        flowPanel.add(new JButton("Button 3"));

        // Create a panel with BorderLayout
        JPanel borderPanel = new JPanel(new BorderLayout());
        borderPanel.setBackground(Color.CYAN);
        borderPanel.add(new JButton("North Button"), BorderLayout.NORTH);
        borderPanel.add(new JButton("South Button"), BorderLayout.SOUTH);
        borderPanel.add(new JButton("East Button"), BorderLayout.EAST);
        borderPanel.add(new JButton("West Button"), BorderLayout.WEST);
        borderPanel.add(new JButton("Center Button"), BorderLayout.CENTER);

        // Use BorderLayout for the main frame
        frame.setLayout(new BorderLayout());

        // Add FlowLayout panel at the top
        frame.add(flowPanel, BorderLayout.NORTH);

        // Add BorderLayout panel in the center
        frame.add(borderPanel, BorderLayout.CENTER);

        // Make the frame visible
        frame.setVisible(true);
    }
}
```

Output ::


Button 1
Button 2
Button 3
North Button
West Button
Center Button
East Button
South Button

Experiment - 20

AIM :: Write a Java Program to represent a Human Skull

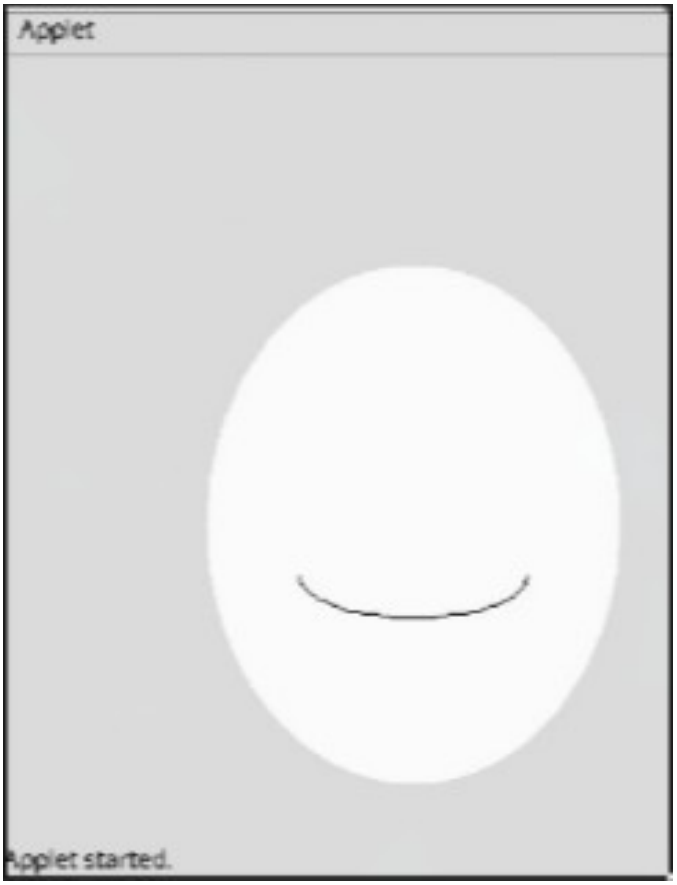
Code ::

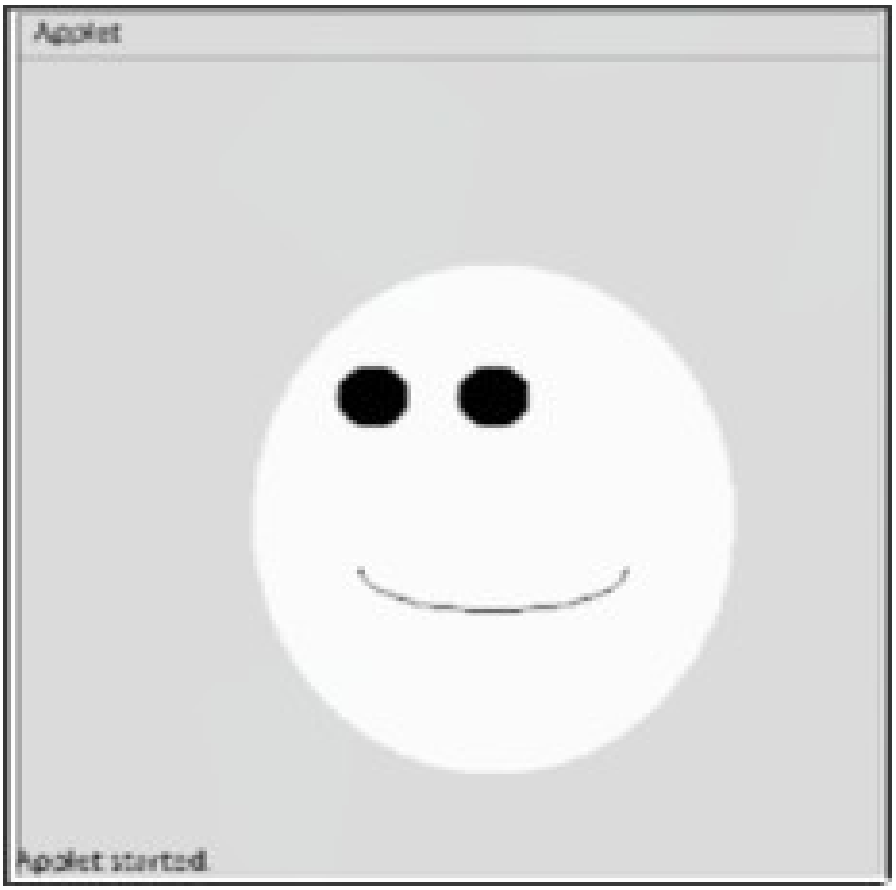
HumanSkullDrawing.java

```
import javax.swing.*;
import java.awt.*;

public class HumanSkullDrawing extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Cast Graphics to Graphics2D for better control over drawing
        Graphics2D g2d = (Graphics2D) g;

        // Anti-aliasing for smoother curves
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        // Draw skull outline (main head shape)
        g2d.setColor(new Color(255, 255, 255)); // Skull color: white
        g2d.fillOval(100, 50, 200, 250); // Oval for the skull

        // Draw eyes (two black circles)
        g2d.setColor(Color.BLACK);
        g2d.fillOval(145, 120, 35, 50); // Left eye
        g2d.fillOval(220, 120, 35, 50); // Right eye

        // Draw nose (triangle-like shape)
        int[] xPoints = {190, 210, 200}; // X coordinates for the triangle
        int[] yPoints = {180, 180, 220}; // Y coordinates for the triangle
        g2d.fillPolygon(xPoints, yPoints, 3); // Nose is a simple triangle

        // Draw mouth (curved shape)
        g2d.setColor(Color.BLACK);
        g2d.drawArc(150, 220, 100, 50, 0, -180); // Mouth using arc

        // Draw teeth (simplified)
        g2d.setColor(Color.WHITE);
        g2d.fillRect(160, 230, 20, 30); // Left tooth
        g2d.fillRect(180, 230, 20, 30); // Second tooth
        g2d.fillRect(200, 230, 20, 30); // Third tooth
        g2d.fillRect(220, 230, 20, 30); // Fourth tooth
        g2d.fillRect(240, 230, 20, 30); // Fifth tooth

        // Draw cracks on the skull (optional for effect)
        g2d.setColor(Color.GRAY);
        g2d.drawLine(160, 80, 220, 110); // Example crack
        g2d.drawLine(230, 100, 170, 150); // Another crack
    }

    public static void main(String[] args) {
        // Create a frame to display the panel
        JFrame frame = new JFrame("Human Skull Drawing");
        HumanSkullDrawing skullPanel = new HumanSkullDrawing();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400); // Set the frame size
        frame.add(skullPanel);   // Add the drawing panel to the frame
```

```
        frame.setVisible(true);  // Make the frame visible
    }
            }
```

Output ::





Experiment - 21

AIM :: WAP to implement a student java bean and access it using another java file.

Code ::

1. Student.java

```
import java.io.Serializable;

public class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id; private int age;
    private String name;

    public Student() {}
    public Student(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public int getAge() { return age; }
    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println("Age must be positive.");
        }
    }

    @Override
    public String toString() {
        return "Student [ID=" + id + ", Name=" + name + ", Age=" + age + "]";
    }
}
```

2. StudentTest.java

```
public class StudentTest {
  public static void main(String[] args) {
     // Creating a Student object using no-arg constructor
     Student student = new Student();
     student.setId(101);
     student.setName("Alice");
     student.setAge(20);

     // Displaying student details
     System.out.println("Student ID: " + student.getId());
     System.out.println("Student Name: " + student.getName());
     System.out.println("Student Age: " + student.getAge());
     System.out.println("The Student 1 Is : " + student);

     // Creating another Student object using parameterized constructor
     Student student2 = new Student(102, "Bob", 22);
     System.out.println("The Student 2 Is : " + student2);
  }
}
```

Output ::

```
Code$ javac Student.java StudentTest.java
Code$ java StudentTest
Student ID: 101
Student Name: Alice
Student Age: 20
The Student 1 Is : Student [ID=101, Name=Alice, Age=20]
The Student 2 Is : Student [ID=102, Name=Bob, Age=22]
```

Experiment - 22

AIM :: WAP to implement an employee java bean and access it using another java file.

Code ::

1. Employee.java

```
import java.io.Serializable;

public class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    private int empId;
    private double salary;
    private String empName;

    public Employee() {}

    public int getEmpId() { return empId; }
    public void setEmpId(int empId) { this.empId = empId; }

    public String getEmpName() { return empName; }
    public void setEmpName(String empName) { this.empName = empName; }

    public double getSalary() { return salary; }
    public void setSalary(double salary) {
        if (salary > 0) {
            this.salary = salary;
        } else {
            System.out.println("Salary must be positive.");
        }
    }

    @Override
    public String toString() {
        return "Employee [ID=" + empId + ", Name=" + empName + ", Salary=" + salary + "]";
    }
}
```

2. EmployeeTest.java

```
public class EmployeeTest {
```

```
    public static void main(String[] args) {
        // Creating an Employee Bean using the no-arg constructor
        Employee emp = new Employee();

        // Setting properties using setter methods with random values
        emp.setEmpId(2034);
        emp.setEmpName("Lena Rivers");
        emp.setSalary(78560.50);

        // Getting properties using getter methods
        System.out.println("Employee ID: " + emp.getEmpId());
        System.out.println("Employee Name: " + emp.getEmpName());
        System.out.println("Employee Salary: $" + emp.getSalary());
        System.out.println("The Employee Details Are : " + emp);
    }
}
```

Output ::

```
Code$ javac Employee.java EmployeeTest.java
Code$ java EmployeeTest
Employee ID: 2034
Employee Name: Lena Rivers
Employee Salary: $78560.5
The Employee Details Are : Employee [ID=2034, Name=Lena Rivers, Salary=78560.5]
```

Experiment - 23

AIM :: Write a servlet using Web-Servlet to handle GET request and display HTML response

Code ::

HelloServlet.java

```
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request , HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE html><html lang='en'><head>");

        out.println("<meta charset='UTF-8'><meta name='viewport' content='width=device-
width, initial-scale=1.0'>");
        out.println("<title>Document</title></head><body>");
        out.println("<h2>Hello, World! Welcome to Servlet Programming.</h2>");
        out.println("<p>Created By : Amit Singhal , 11614802722</p></body></html>");
    }
}
```

pom.xml (Dependency Section Snippet)

```
<dependency>
    <groupId>jakarta.servlet</groupId>
    <artifactId>jakarta.servlet-api</artifactId>
    <version>5.0.0</version>
    <scope>provided</scope>
</dependency>
```

Output ::

Experiment - 24

AIM :: Write a Java Program to create a Chat Server

Code ::

1. ChatServer.java

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer {
    // List to hold all connected client handlers
    private static final List<ClientHandler> clients = new ArrayList<>();

    public static void main(String[] args) {
        int port = 12345; // The server port
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Chat Server started on port " + port);
            // Continuously accept client connections
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected: " +
clientSocket.getInetAddress());
                // Create a new client handler to manage the client's messages
                ClientHandler clientHandler = new ClientHandler(clientSocket);
                clients.add(clientHandler);
                // Start a new thread for the client
                new Thread(clientHandler).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Method to broadcast messages to all clients
    public static void broadcastMessage(String message, ClientHandler sender) {
        for (ClientHandler client : clients) {
            // Don't send the message to the sender
            if (client != sender) {
                client.sendMessage(message);
            }
        }
    }

    // Inner class to handle each client's communication
    private static class ClientHandler implements Runnable {
        private Socket socket;
        private PrintWriter out;
        private BufferedReader in;
        private String clientName;

        public ClientHandler(Socket socket) {
            this.socket = socket;
            try {
                // Setup input and output streams
                in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                out = new PrintWriter(socket.getOutputStream(), true);
                // Get the client's name
                out.println("Enter your name: ");
```

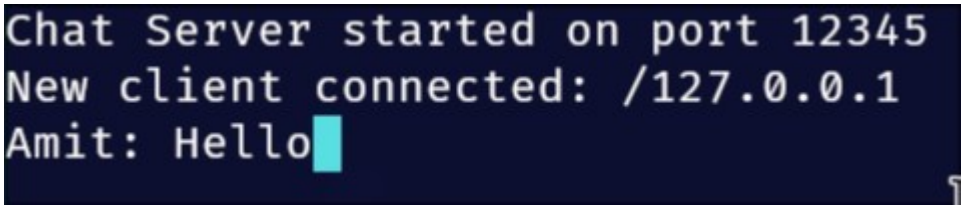
```
                clientName = in.readLine();
                out.println("Welcome to the chat, " + clientName + "!");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        // Method to send message to the client
        public void sendMessage(String message) {
            out.println(message);
        }

        @Override
        public void run() {
            String message;
            try {
                while ((message = in.readLine()) != null) {
                    if (message.equalsIgnoreCase("exit")) {
                        break;
                    }
                    // Broadcast the message to all clients
                    System.out.println(clientName + ": " + message);
                    ChatServer.broadcastMessage(clientName + ": " + message, this);
                }
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    // Remove the client from the list of active clients
                    clients.remove(this);
                    socket.close();
                    System.out.println(clientName + " has disconnected.");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

2. ChatClient.java

```
import java.io.*;
import java.net.*;

public class ChatClient {
    private static PrintWriter out;
    private static BufferedReader in;
    private static BufferedReader userInput;

    public static void main(String[] args) {
        String serverAddress = "localhost"; // The server address (localhost for local
machine)
        int port = 12345; // The server port

        try (Socket socket = new Socket(serverAddress, port)) {
            // Setup input and output streams
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);
            userInput = new BufferedReader(new InputStreamReader(System.in));

            // Read welcome message from the server
            String serverMessage = in.readLine();
            System.out.println(serverMessage);

            // Get the client's name from the user
```

```
            String name = userInput.readLine();
            out.println(name);

            // Listen for server messages and print them to the console
            new Thread(new ServerListener()).start();

            // Send messages to the server
            String message;
            while (true) {
                message = userInput.readLine();
                out.println(message);
                if (message.equalsIgnoreCase("exit")) {
                    break;
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Inner class to listen for messages from the server
    private static class ServerListener implements Runnable {
        @Override
        public void run() {
            String message;
            try {
                while ((message = in.readLine()) != null) {
                    System.out.println(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output ::



Experiment - 25

AIM :: Write a Java Program to Showcase the use of Listeners

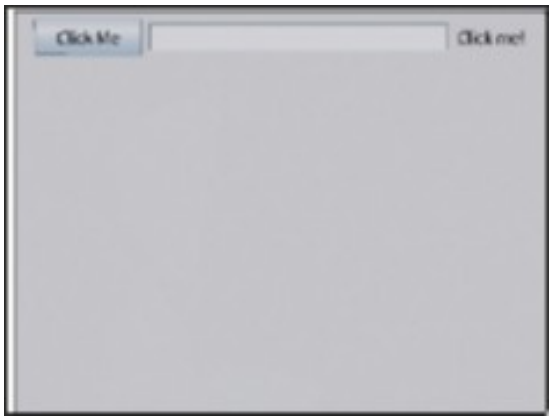
Code ::

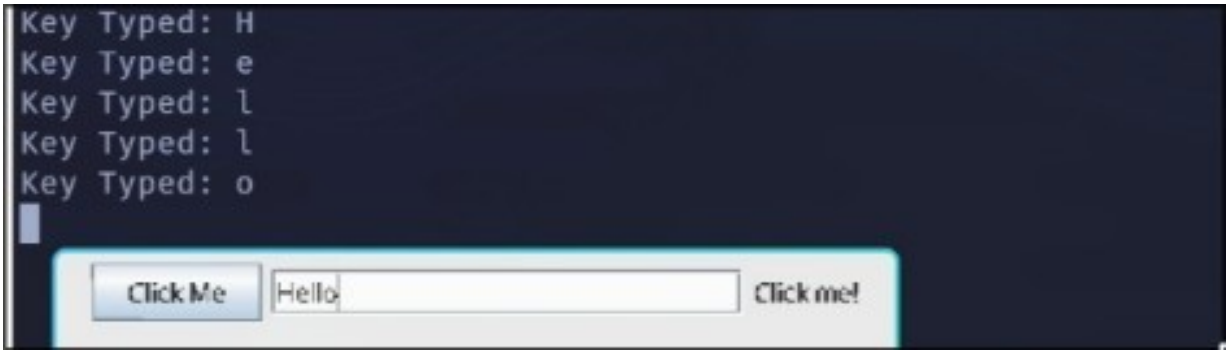
1. SwingListenersExample.java

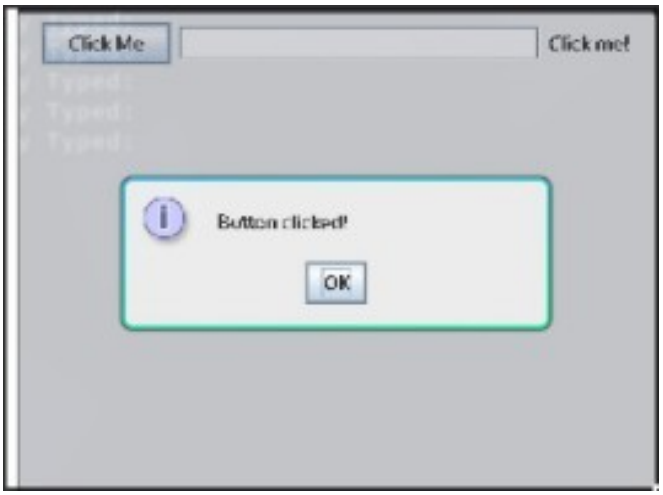
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingListenersExample {
    public static void main(String[] args) {
        // Create a frame for the GUI
        JFrame frame = new JFrame("Swing Listeners Example");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a panel to hold components
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout());

        // Create a button and add an ActionListener
        JButton button = new JButton("Click Me");
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(frame, "Button clicked!");
            }
        });

        // Create a text field and add a KeyListener
        JTextField textField = new JTextField(20);
        textField.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent e) {
                // Display the key that is typed
                char c = e.getKeyChar();
                System.out.println("Key Typed: " + c);
            }
        });

        // Create a label and add a MouseListener
        JLabel label = new JLabel("Click me!");
        label.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                JOptionPane.showMessageDialog(frame, "Label clicked!");
            }
        });

        // Add the components to the panel
        panel.add(button);
        panel.add(textField);
        panel.add(label);

        // Add the panel to the frame
        frame.add(panel);

        // Set the frame to be visible
```

```
            frame.setVisible(true);
        }
    }
```

Output ::







Experiment - 26

AIM :: Write a simple Swing Program in java.

Code ::

1. SimpleSwingProgram.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SimpleSwingProgram {
    public static void main(String[] args) {
        // Create a new JFrame (main window)
        JFrame frame = new JFrame("Simple Swing Program");

        // Set the layout for the frame
        frame.setLayout(new FlowLayout());

        // Create a label
        JLabel label = new JLabel("Enter your name:");

        // Create a text field for user input
        JTextField textField = new JTextField(20);

        // Create a button that will trigger an action
        JButton button = new JButton("Submit");

        // ActionListener for the button
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = textField.getText();
                label.setText("Hello, " + name + "!");
            }
        });

        // Add the components to the frame
        frame.add(label);
        frame.add(textField);
        frame.add(button);

        // Set frame properties
        frame.setSize(300, 150); // Set the size of the window
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close the application
when the window is closed
        frame.setVisible(true);
    }
}
```

Output ::

Experiment - 27

AIM :: Write a Java Program to showcase the use of JSP

Code ::

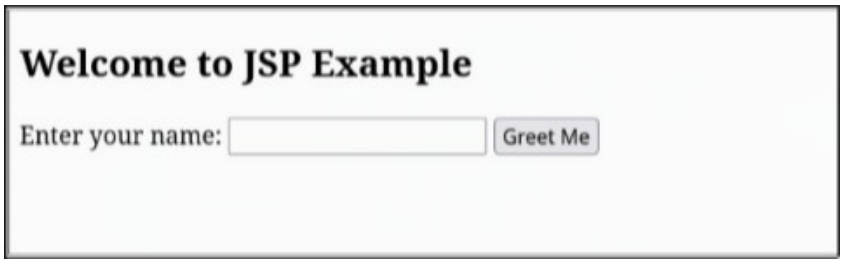
1. index.html

```
<html>
<head>
    <title>Simple JSP Example</title>
</head>
<body>
    <h2>Welcome to JSP Example</h2>
    <form action="greet.jsp" method="post">
        Enter your name: <input type="text" name="username" />
        <input type="submit" value="Greet Me" />
    </form>
</body>
</html>
```

2. greet.jsp

```
<!DOCTYPE html>
<html>
<head>
    <title>Greeting Page</title>
</head>
<body>
    <h2>Hello, <%= request.getParameter("username") %>!</h2>
    <p>Welcome to the JSP World!</p>
    <a href="index.jsp">Go Back</a>
</body>
</html>
```

Output ::



Experiment - 28

AIM :: Write a Java Program to showcase the use of a Servlet

Code ::

1. HelloWorldServlet.java

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

public class HelloWorldServlet extends HttpServlet {
    // Overriding the doGet method to handle HTTP GET requests
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Setting the response content type
        response.setContentType("text/html");

        // Writing the response message
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Hello, World!</h1>");
        out.println("</body></html>");
    }
}
```

2. web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">

    <servlet>
        <servlet-name>HelloWorldServlet</servlet-name>
        <servlet-class>HelloWorldServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWorldServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

Output ::