Unit 1

1. Write a java program to check if a given integer entered by the user is odd or even.

Ans-

```
import java.util.Scanner;

public class OddOrEven {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int number = input.nextInt();

        if (number % 2 == 0) {
            System.out.println(number + " is even.");
        } else {
            System.out.println(number + " is odd.");
        }

        input.close();
    }
}
```

2. Write a java program that calculates the compound interest, taking the necessary input from the user.

   Ans:-

```
import java.util.Scanner;

public class CompoundInterest {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter the principal amount: ");
        double principal = input.nextDouble();

        System.out.print("Enter the annual interest rate: ");
        double interestRate = input.nextDouble();

        System.out.print("Enter the number of years: ");
        int years = input.nextInt();

        System.out.print("Enter the number of times interest is compounded per
year: ");
        int timesCompounded = input.nextInt();

        double amount = principal * Math.pow(1 + (interestRate /
timesCompounded), timesCompounded * years);
        double interest = amount - principal;

        System.out.printf("The compound interest is %.2f", interest);

        input.close();
```

```
    }
}
```

3. Write a java program to sort an array in ascending order, array elements should be taken as input from the user.

Ans:-

```
import java.util.Scanner;

public class ArraySort {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int size = input.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter the elements of the array:");

        for (int i = 0; i < size; i++) {
            arr[i] = input.nextInt();
        }

        for (int i = 0; i < size; i++) {
            for (int j = i + 1; j < size; j++) {
                if (arr[i] > arr[j]) {
                    int temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }

        System.out.println("The array in ascending order is:");

        for (int i = 0; i < size; i++) {
            System.out.print(arr[i] + " ");
        }

        input.close();
    }
}
```

4. Write a program to give the example of control statements.
   1 If statements. 2 Switch Statements. 3 For loop. 4 While Statements. 5 Do statements

Ans:-

```
import java.util.Scanner;

public class ControlStatementsExample {
```

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    // if statement example
    System.out.print("Enter a number: ");
    int num = input.nextInt();

    if (num % 2 == 0) {
        System.out.println(num + " is even.");
    } else {
        System.out.println(num + " is odd.");
    }

    // switch statement example
    System.out.print("Enter a day number (1-7): ");
    int day = input.nextInt();

    switch (day) {
        case 1:
            System.out.println("Monday");
            break;
        case 2:
            System.out.println("Tuesday");
            break;
        case 3:
            System.out.println("Wednesday");
            break;
        case 4:
            System.out.println("Thursday");
            break;
        case 5:
            System.out.println("Friday");
            break;
        case 6:
            System.out.println("Saturday");
            break;
        case 7:
            System.out.println("Sunday");
            break;
        default:
            System.out.println("Invalid day number.");
    }

    // for loop example
    System.out.print("Enter the number of terms in the sequence: ");
    int n = input.nextInt();
    int a = 0, b = 1;

    System.out.print("The Fibonacci sequence of " + n + " terms is: ");

    for (int i = 1; i <= n; i++) {
        System.out.print(a + " ");

        int sum = a + b;
        a = b;
        b = sum;
    }
```

```
        System.out.println();

        // while loop example
        int i = 1;
        System.out.print("The first 10 natural numbers are: ");

        while (i <= 10) {
            System.out.print(i + " ");
            i++;
        }

        System.out.println();

        // do-while loop example
        int password = 0;

        do {
            System.out.print("Enter the password (1234): ");
            password = input.nextInt();
        } while (password != 1234);

        System.out.println("Access granted!");

        input.close();
    }
}
```

5. Write a program to calculate the following

   1 Find the length of the array. 2 Demonstrate a one-dimensional array. 3 Demonstrate a two-dimensional array. 4 Demonstrate a multidimensional array.

Ans:-

```
public class ArrayExample {
    public static void main(String[] args) {
        // Finding the length of the array
        int[] array = {1, 2, 3, 4, 5};
        System.out.println("Length of the array is: " + array.length);

        // One-dimensional array example
        int[] oneDimArray = new int[5];
        oneDimArray[0] = 1;
        oneDimArray[1] = 2;
        oneDimArray[2] = 3;
        oneDimArray[3] = 4;
        oneDimArray[4] = 5;
        System.out.print("One-dimensional array elements: ");
        for (int i = 0; i < oneDimArray.length; i++) {
            System.out.print(oneDimArray[i] + " ");
        }
        System.out.println();

        // Two-dimensional array example
```

```
        int[][] twoDimArray = new int[3][3];
        twoDimArray[0][0] = 1;
        twoDimArray[0][1] = 2;
        twoDimArray[0][2] = 3;
        twoDimArray[1][0] = 4;
        twoDimArray[1][1] = 5;
        twoDimArray[1][2] = 6;
        twoDimArray[2][0] = 7;
        twoDimArray[2][1] = 8;
        twoDimArray[2][2] = 9;
        System.out.println("Two-dimensional array elements:");
        for (int i = 0; i < twoDimArray.length; i++) {
            for (int j = 0; j < twoDimArray[i].length; j++) {
                System.out.print(twoDimArray[i][j] + " ");
            }
            System.out.println();
        }

        // Multidimensional array example
        int[][][] multiDimArray = {{{1, 2}, {3, 4}}, {{5, 6}, {7, 8}}};
        System.out.println("Multidimensional array elements:");
        for (int i = 0; i < multiDimArray.length; i++) {
            for (int j = 0; j < multiDimArray[i].length; j++) {
                for (int k = 0; k < multiDimArray[i][j].length; k++) {
                    System.out.print(multiDimArray[i][j][k] + " ");
                }
                System.out.println();
            }
            System.out.println();
        }
    }
}
```

6. Write a program to find the following 1. Prime number checking 2. Sum of digit

```
import java.util.Scanner;

public class PrimeNumberChecker {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a number: ");
      int num = sc.nextInt();
      sc.close();

      boolean isPrime = true;
      if(num <= 1) {
         isPrime = false;
      } else {
         for(int i = 2; i <= Math.sqrt(num); i++) {
            if(num % i == 0) {
               isPrime = false;
```

```
                break;
            }
        }
    }

    if(isPrime) {
        System.out.println(num + " is a prime number.");
    } else {
        System.out.println(num + " is not a prime number.");
    }
  }
}
```

```
import java.util.Scanner;

public class SumOfDigits {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a number: ");
    int num = sc.nextInt();
    sc.close();

    int sum = 0;
    while(num != 0) {
      sum += num % 10;
      num /= 10;
    }

    System.out.println("Sum of digits: " + sum);
  }
}
```

7. Write a program to arrange the numbers in ascending order.

Ans:-

```
import java.util.Scanner;

public class AscendingOrderSort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter " + n + " numbers:");
```

```
        for(int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        sc.close();

        // sort the array in ascending order using bubble sort
        for(int i = 0; i < n - 1; i++) {
            for(int j = 0; j < n - i - 1; j++) {
                if(arr[j] > arr[j+1]) {
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }

        // display the sorted array
        System.out.println("Array in ascending order:");
        for(int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

8. Write a program to calculate the roots of Quadratic equations.

```
import java.util.Scanner;

public class QuadraticEquationRoots {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the value of coefficient a: ");
        double a = sc.nextDouble();
        System.out.print("Enter the value of coefficient b: ");
        double b = sc.nextDouble();
        System.out.print("Enter the value of coefficient c: ");
        double c = sc.nextDouble();
        sc.close();

        double discriminant = b * b - 4 * a * c;
        if(discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("The roots are real and different.");
            System.out.println("Root 1: " + root1);
            System.out.println("Root 2: " + root2);
        } else if(discriminant == 0) {
            double root = -b / (2 * a);
            System.out.println("The roots are real and equal.");
            System.out.println("Root: " + root);
        } else {
            double realPart = -b / (2 * a);
            double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
            System.out.println("The roots are complex and different.");
            System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
            System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
```

```
        }
    }
}
```

9. Write a program for calculating Matrix Operations. 1 Addition. 2 Multiplication

Ans:-

```
import java.util.Scanner;

public class MatrixOperations {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows of matrix 1: ");
        int r1 = sc.nextInt();
        System.out.print("Enter the number of columns of matrix 1: ");
        int c1 = sc.nextInt();
        int[][] mat1 = new int[r1][c1];
        System.out.println("Enter the elements of matrix 1:");
        for(int i = 0; i < r1; i++) {
            for(int j = 0; j < c1; j++) {
                mat1[i][j] = sc.nextInt();
            }
        }
        System.out.print("Enter the number of rows of matrix 2: ");
        int r2 = sc.nextInt();
        System.out.print("Enter the number of columns of matrix 2: ");
        int c2 = sc.nextInt();
        int[][] mat2 = new int[r2][c2];
        System.out.println("Enter the elements of matrix 2:");
        for(int i = 0; i < r2; i++) {
            for(int j = 0; j < c2; j++) {
                mat2[i][j] = sc.nextInt();
            }
        }
        sc.close();

        // check if matrices can be added and multiplied
        if(r1 != r2 || c1 != c2) {
            System.out.println("Matrices cannot be added or multiplied.");
            return;
        }

        // add matrices
        int[][] sum = new int[r1][c1];
        for(int i = 0; i < r1; i++) {
            for(int j = 0; j < c1; j++) {
                sum[i][j] = mat1[i][j] + mat2[i][j];
            }
        }
        System.out.println("The sum of the matrices is:");
        for(int i = 0; i < r1; i++) {
            for(int j = 0; j < c1; j++) {
                System.out.print(sum[i][j] + " ");
            }
            System.out.println();
        }
```

```
        // multiply matrices
        int[][] product = new int[r1][c2];
        for(int i = 0; i < r1; i++) {
            for(int j = 0; j < c2; j++) {
                for(int k = 0; k < c1; k++) {
                    product[i][j] += mat1[i][k] * mat2[k][j];
                }
            }
        }
        System.out.println("The product of the matrices is:");
        for(int i = 0; i < r1; i++) {
            for(int j = 0; j < c2; j++) {
                System.out.print(product[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

10. Write a program to check if two strings are equal or not.

Ans:-

```
import java.util.Scanner;

public class StringEquality {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String str1 = sc.nextLine();
        System.out.print("Enter the second string: ");
        String str2 = sc.nextLine();
        sc.close();

        if(str1.equals(str2)) {
            System.out.println("The two strings are equal.");
        } else {
            System.out.println("The two strings are not equal.");
        }
    }
}
```

Unit II

1. Write a program to create interface
   A. In this interface we have two methods meth1 and meth2.
   Implements this interface in another class named MyClass.

Ans:-

```
interface MyInterface {
   void meth1();
   void meth2();
}

class MyClass implements MyInterface {
   public void meth1() {
      System.out.println("Inside meth1 method of MyClass");
   }
   public void meth2() {
      System.out.println("Inside meth2 method of MyClass");
   }
}

public class InterfaceDemo {
   public static void main(String[] args) {
      MyClass obj = new MyClass();
      obj.meth1();
      obj.meth2();
   }
}
```

In this program, we first create an interface named **`MyInterface`** with two methods **`meth1()`** and **`meth2()`**. Then, we create a class named **`MyClass`** that implements this interface. In the implementation of **`MyClass`**, we define the two methods **`meth1()`** and **`meth2()`** that are declared in the **`MyInterface`** interface.

In the **`main()`** method of the **`InterfaceDemo`** class, we create an object of **`MyClass`** and call the **`meth1()`** and **`meth2()`** methods on this object. When we run this program, it will output:

```
Inside meth1 method of MyClass
Inside meth2 method of MyClass
```

This demonstrates how we can define an interface with methods and then implement those methods in a class.

2. Write a program to give examples for multiple inheritance in Java.
   Ans:-
   Java does not support multiple inheritance through classes. However, it does support multiple inheritance through interfaces. Here's an example program that demonstrates multiple inheritance through interfaces:

```
interface Animal {
    void eat();
}

interface Mammal {
    void run();
}

class Platypus implements Animal, Mammal {
    public void eat() {
        System.out.println("Platypus is eating");
    }
    public void run() {
        System.out.println("Platypus is running");
    }
}

public class MultipleInheritanceDemo {
    public static void main(String[] args) {
        Platypus p = new Platypus();
        p.eat();
        p.run();
    }
}
```

In this program, we define two interfaces **`Animal`** and **`Mammal`** with their respective methods **`eat()`** and **`run()`**. We then define a class **`Platypus`** that implements both these interfaces. In the implementation of **`Platypus`**, we define the **`eat()`** and **`run()`** methods.

Finally, in the `main()` method of the `MultipleInheritanceDemo` class, we create an object of `Platypus` and call the `eat()` and `run()` methods on it. Since `Platypus` implements both `Animal` and `Mammal` interfaces, it is able to inherit the methods from both the interfaces.

Note that this is an example of multiple inheritance through interfaces. Java does not allow multiple inheritance through classes because it can lead to the diamond problem where the compiler is unable to resolve conflicts when two or more superclasses have methods with the same signature.

3. Write a program to create an interface named test. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called ToTestInt in this class using the object of the arithmetic class.
   Ans:-
   Sure, here's a Java program that creates an interface named `Test` with a method `square()`, implements this interface in an `Arithmetic` class, and creates a new class named `ToTestInt` that uses the `Arithmetic` class to call the `square()` method:

```
interface Test {
    int square(int n);
}

class Arithmetic implements Test {
    public int square(int n) {
        return n * n;
    }
}

class ToTestInt {
    public static void main(String[] args) {
        Arithmetic obj = new Arithmetic();
        int result = obj.square(5);
        System.out.println("Square of 5 is: " + result);
    }
}
```

In this program, we first create an interface named `Test` with a method `square()` that takes an integer argument and returns an integer result. Then, we create a class named `Arithmetic` that implements the `Test` interface and provides an implementation for the `square()` method.

Finally, we create a new class named **`ToTestInt`** that uses an object of the **`Arithmetic`** class to call the **`square()`** method with the argument **`5`** and print out the result. When we run this program, it will output:

```
Square of 5 is: 25
```

This demonstrates how we can define an interface with a method and then implement that method in a class. We can then use an object of the implementing class to call the method.

4. Create an outer class with a function display, again create another class inside the outer class named inner with a function called display and call the two functions in the main class.

   Ans:-

Sure, here's a Java program that creates an outer class **`OuterClass`** with a function **`display()`**, creates an inner class **`InnerClass`** inside **`OuterClass`** with its own function **`display()`**, and calls both functions in the **`main()`** method:

```java
class OuterClass {
    void display() {
        System.out.println("This is the outer class display method");
    }

    class InnerClass {
        void display() {
            System.out.println("This is the inner class display method");
        }
    }
}

public class OuterInnerDemo {
    public static void main(String[] args) {
        OuterClass outerObj = new OuterClass();
        outerObj.display();

        OuterClass.InnerClass innerObj = outerObj.new InnerClass();
        innerObj.display();
    }
}
```

In this program, we define an outer class named **`OuterClass`** with a function **`display()`** that prints a message to the console. We then define an inner class named **`InnerClass`** inside **`OuterClass`** with its own **`display()`** function that also prints a message to the console.

In the **`main()`** method of the **`OuterInnerDemo`** class, we create an object of the **`OuterClass`** and call its **`display()`** function. We then create an object of the **`InnerClass`** using the **`new`** keyword and the **`outerObj`** object, and call its **`display()`** function. Since the **`InnerClass`** is inside the **`OuterClass`**, we need to create an object of the **`InnerClass`** using an object of the **`OuterClass`**.

When we run this program, it will output:

```
This is the outer class display method
This is the inner class display method
```

This demonstrates how we can create an inner class inside an outer class, and call both the outer and inner class methods in the main class.

5. Write a program to create a class named shape. In this class we have three sub classes circle, triangle and square. Each class has two member functions named draw () and erase (). Create these using polymorphism concepts.

   Ans:- Here's a Java program that creates a **`Shape`** class and three subclasses **`Circle`**, **`Triangle`**, and **`Square`**, and uses polymorphism to call their **`draw()`** and **`erase()`** methods:

```
class Shape {
    void draw() {
        System.out.println("Drawing a shape");
    }

    void erase() {
        System.out.println("Erasing a shape");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle");
    }

    void erase() {
        System.out.println("Erasing a circle");
    }
```

```
}

class Triangle extends Shape {
    void draw() {
        System.out.println("Drawing a triangle");
    }

    void erase() {
        System.out.println("Erasing a triangle");
    }
}

class Square extends Shape {
    void draw() {
        System.out.println("Drawing a square");
    }

    void erase() {
        System.out.println("Erasing a square");
    }
}

public class ShapeDemo {
    public static void main(String[] args) {
        Shape[] shapes = new Shape[3];
        shapes[0] = new Circle();
        shapes[1] = new Triangle();
        shapes[2] = new Square();

        for (Shape shape : shapes) {
            shape.draw();
            shape.erase();
            System.out.println();
        }
    }
}
```

In this program, we define a **`Shape`** class with **`draw()`** and **`erase()`** methods that simply print a message to the console.

We then define three subclasses **`Circle`**, **`Triangle`**, and **`Square`** that extend the **`Shape`** class and override its **`draw()`** and **`erase()`** methods to print messages specific to the shape.

In the **`main()`** method of the **`ShapeDemo`** class, we create an array of **`Shape`** objects and initialize them with objects of **`Circle`**, **`Triangle`**, and **`Square`**. We then loop through the array, calling the **`draw()`** and **`erase()`** methods on each **`Shape`** object using polymorphism.

When we run this program, it will output:

```
Drawing a circle
Erasing a circle

Drawing a triangle
Erasing a triangle

Drawing a square
Erasing a square
```

This demonstrates how we can use polymorphism to call methods on objects of different classes that share the same base class.

6. Create a class named ‘a’ and create a sub class ‘b’. Which extends from class ‘a’. And use these classes in the ‘inherit’ class.

   Ans:-

   Here's a Java program that creates a class **`A`** and a subclass **`B`** which extends **`A`**, and uses these classes in an **`Inherit`** class:

```
class A {
   void display() {
      System.out.println("This is class A");
   }
}

class B extends A {
   void display() {
      System.out.println("This is class B");
   }
}
```

```
public class Inherit {
    public static void main(String[] args) {
        A a = new A();
        B b = new B();

        a.display();
        b.display();
    }
}
```

In this program, we define a class **A** with a method **display()** that prints a message to the console.

We then define a subclass **B** that extends **A** and overrides its **display()** method to print a different message.

In the **main()** method of the **Inherit** class, we create an object of **A** and an object of **B**, and call their **display()** methods. When we call the **display()** method on the **A** object, it prints "This is class A", and when we call it on the **B** object, it prints "This is class B". This demonstrates how subclass **B** inherits the behavior of its superclass **A** and can override its methods with its own implementation.

7. Write a program to give the example for method overriding concepts.
   Ans:-

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

class Dog extends Animal {
    void makeSound() {
        System.out.println("The dog barks");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal animal = new Animal();
        Dog dog = new Dog();
```

```
        animal.makeSound(); // Output: The animal makes a sound
        dog.makeSound();    // Output: The dog barks
    }
}
```

In this program, we define a class **`Animal`** with a method **`makeSound()`** that simply prints "The animal makes a sound" to the console.

We then define a subclass **`Dog`** that extends **`Animal`** and overrides its **`makeSound()`** method with its own implementation that prints "The dog barks" to the console.

In the **`main()`** method of the **`Main`** class, we create an object of **`Animal`** and an object of **`Dog`**, and call their **`makeSound()`** methods. When we call the **`makeSound()`** method on the **`Animal`** object, it prints "The animal makes a sound", and when we call it on the **`Dog`** object, it prints "The dog barks". This demonstrates how the **`Dog`** subclass overrides the behavior of its superclass **`Animal`** with its own implementation of the **`makeSound()`** method.

8. Write a program to give the example for ‘super’ keyword

Ans:-

```
class Animal {
    String name;

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

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

class Dog extends Animal {
    String breed;

    Dog(String name, String breed) {
        super(name);
        this.breed = breed;
    }
```

```
    void makeSound() {
        super.makeSound();
        System.out.println("The dog barks");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog("Fido", "Golden Retriever");
        System.out.println("Name: " + dog.name);
        System.out.println("Breed: " + dog.breed);
        dog.makeSound();
    }
}
```

In this program, we define a class **`Animal`** with a **`name`** field and a method **`makeSound()`** that simply prints "The animal makes a sound" to the console.

We then define a subclass **`Dog`** that extends **`Animal`** and adds a **`breed`** field. The **`Dog`** constructor takes two arguments, **`name`** and **`breed`**, and calls the **`Animal`** constructor with **`super(name)`** to set the **`name`** field of the superclass.

The **`Dog`** class also overrides the **`makeSound()`** method of the **`Animal`** class with its own implementation that first calls the **`makeSound()`** method of the superclass using **`super.makeSound()`**, and then prints "The dog barks" to the console.

In the **`main()`** method of the **`Main`** class, we create a **`Dog`** object with a name of "Fido" and a breed of "Golden Retriever". We then print out the **`name`** and **`breed`** fields of the **`Dog`** object, and call its **`makeSound()`** method. When we call the **`makeSound()`** method, it first prints "The animal makes a sound" (which comes from the **`Animal`** class implementation), and then "The dog barks" (which comes from the **`Dog`** class implementation). This demonstrates how the **`super`** keyword can be used to call a method of the superclass from a subclass, and how it can be used in conjunction with overriding to extend the behavior of the superclass.

9. Write a java program to create a package named myPackage and add two classes in it.
   Ans:-

To create a package named **`myPackage`** and add two classes in it, follow the steps below:

1. Create a directory named **`myPackage`** in your project directory.

2. Inside the **myPackage** directory, create two Java files named **ClassA.java** and **ClassB.java**.
3. In each of the Java files, add the package declaration as **package myPackage;**.
4. Define the classes **ClassA** and **ClassB** with some methods and properties.

Here is an example code for **ClassA** and **ClassB**:

ClassA.java:

```
package myPackage;

public class ClassA {
    public void display() {
        System.out.println("This is ClassA");
    }
}
```

ClassB.java:

```
package myPackage;

public class ClassB {
    public void display() {
        System.out.println("This is ClassB");
    }
}
```

Note: The class names should be the same as the file names. Also, make sure to set the classpath correctly to access the classes in the package.

.

10. Write a program to calculate the roots of Quadratic equations.

Ans:-

To calculate the roots of a quadratic equation in Java, we can use the following code:

```
import java.util.Scanner;

public class QuadraticEquation {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter the values of a, b, and c:");
        double a = input.nextDouble();
        double b = input.nextDouble();
        double c = input.nextDouble();
```

```
        double root1, root2;
        double determinant = b * b - 4 * a * c;

        if (determinant > 0) {
            root1 = (-b + Math.sqrt(determinant)) / (2 * a);
            root2 = (-b - Math.sqrt(determinant)) / (2 * a);

            System.out.println("Roots are real and different.");
            System.out.println("Root1 = " + root1);
            System.out.println("Root2 = " + root2);
        } else if (determinant == 0) {
            root1 = root2 = -b / (2 * a);

            System.out.println("Roots are real and same.");
            System.out.println("Root1 = Root2 = " + root1);
        } else {
            double realPart = -b / (2 * a);
            double imaginaryPart = Math.sqrt(-determinant) / (2 * a);

            System.out.println("Roots are complex and different.");
            System.out.println("Root1 = " + realPart + "+" + imaginaryPart + "i");
            System.out.println("Root2 = " + realPart + "-" + imaginaryPart + "i");
        }
    }
}
```

In this program, we first read the values of a, b, and c from the user using the Scanner class. Then, we calculate the determinant of the quadratic equation using the formula **b^2 - 4ac**. Based on the value of the determinant, we calculate the roots of the equation and print them to the console.

Note: In case the determinant is negative, we are calculating the square root of a negative number. This will result in a complex number. We are using the "i" notation to represent the imaginary part of the number.

Unit III

1. Write a java program to create a chat room application using GUI. Use multithreading.

   Ans:-

   To create a chat room application using GUI and multithreading in Java, we can use the **`Swing`** library for the GUI and the **`Thread`** class for multithreading. Here is an example program:

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

public class ChatRoom extends JFrame implements ActionListener, Runnable {
   private JTextField textField;
   private JTextArea textArea;
   private JButton sendButton;
   private String message;

   public ChatRoom() {
      super("Chat Room");
      textField = new JTextField(30);
      textArea = new JTextArea(10, 30);
      textArea.setEditable(false);
      sendButton = new JButton("Send");
      sendButton.addActionListener(this);

      Container container = getContentPane();
      container.setLayout(new BorderLayout());
      container.add(textField, BorderLayout.NORTH);
      container.add(new JScrollPane(textArea), BorderLayout.CENTER);
      container.add(sendButton, BorderLayout.SOUTH);

      setSize(400, 300);
      setVisible(true);

      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      Thread thread = new Thread(this);
      thread.start();
```

```
    }

    public void run() {
        while (true) {
            try {
                Thread.sleep(1000);
                if (message != null) {
                    textArea.append("You: " + message + "\n");
                    message = null;
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void actionPerformed(ActionEvent event) {
        if (event.getSource() == sendButton) {
            message = textField.getText();
            textField.setText("");
        }
    }

    public static void main(String[] args) {
        ChatRoom chatRoom = new ChatRoom();
    }
}
```

In this program, we create a **`JFrame`** object and add a **`JTextField`** for input, a **`JTextArea`** for output, and a **`JButton`** for sending messages. We also implement the **`ActionListener`** and **`Runnable`** interfaces to handle events and multithreading. The **`run`** method runs in a separate thread and checks for incoming messages, while the **`actionPerformed`** method handles sending messages.

2. Write a java program using multithreading that will illustrate joint operation of child threads.

   Ans:-

   Here's an example program that demonstrates the joint operation of child threads in Java using multithreading:

```
public class JointThreadsExample {
```

```
    public static void main(String[] args) throws InterruptedException {
        Thread t1 = new Thread(() -> {
            for(int i = 0; i < 5; i++) {
                System.out.println("Thread 1 is running...");
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        Thread t2 = new Thread(() -> {
            for(int i = 0; i < 5; i++) {
                System.out.println("Thread 2 is running...");
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        t1.start();
        t2.start();

        t1.join();
        t2.join();

        System.out.println("Both threads have completed execution.");
    }
}
```

In this program, we create two threads (**`t1`** and **`t2`**) using lambdas that simply print a message and sleep for 500 milliseconds. We start both threads and then use the **`join()`** method to wait for both threads to complete execution before printing a final message indicating that both threads have completed. This ensures that the main thread waits for both child threads to finish before exiting.

3. Write a java program that will search a file on disk using GUI. Create two labels to show current file and current folder under search operation. After the search, if a file is found, show the complete path of that file in the message box or show “file not found” message in the message box.

Ans:-

Here's an example Java program that searches a file on disk using GUI and displays the current file and folder during the search:

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

public class FileSearchGUI extends JFrame implements ActionListener {

    private JTextField searchField;
    private JLabel currentFolderLabel, currentFileLabel;
    private JButton searchButton;

    public FileSearchGUI() {
        setTitle("File Search");
        setSize(500, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        searchField = new JTextField();
        currentFolderLabel = new JLabel("Current Folder: ");
        currentFileLabel = new JLabel("Current File: ");
        searchButton = new JButton("Search");
        searchButton.addActionListener(this);

        JPanel panel1 = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panel1.add(searchField);
        panel1.add(searchButton);

        JPanel panel2 = new JPanel(new GridLayout(2, 1));
        panel2.add(currentFolderLabel);
        panel2.add(currentFileLabel);
```

```
        add(panel1, BorderLayout.NORTH);
        add(panel2, BorderLayout.CENTER);
    }

    public void actionPerformed(ActionEvent e) {
        String searchQuery = searchField.getText();
        File rootFolder = new File("/");
        search(rootFolder, searchQuery);
    }

    private void search(File folder, String searchQuery) {
        File[] files = folder.listFiles();
        if (files != null) {
            for (File file : files) {
                if (file.isDirectory()) {
                                currentFolderLabel.setText("Current Folder: " +
file.getAbsolutePath());
                    search(file, searchQuery);
                } else {
                                        currentFileLabel.setText("Current File: " +
file.getAbsolutePath());
                    if (file.getName().equals(searchQuery)) {
                            JOptionPane.showMessageDialog(null, "File found: " +
file.getAbsolutePath());
                        return;
                    }
                }
            }
        }
        JOptionPane.showMessageDialog(null, "File not found.");
    }

    public static void main(String[] args) {
        FileSearchGUI fileSearchGUI = new FileSearchGUI();
        fileSearchGUI.setVisible(true);
    }
}
```

This program creates a simple GUI with a text field for the search query, a label to display the current folder being searched, a label to display the current file being searched, and a button to start the search. When the user clicks the search button, the program searches the root folder recursively for the file with the given name. During the search, the program updates the current folder and file labels to show the current folder and file being searched. When the file is found, the program displays the complete path of the file in a message box. If the file is not found, the program displays a message box with the message "File not found."

4. Write a program for example of try and catch block. In this check whether the given array size is negative or not.

Ans:- Sure, here's an example program demonstrating the use of try and catch blocks to handle negative array size:

```
import java.util.Scanner;

public class ArrayExample {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter the size of the array: ");
    int size = input.nextInt();

    try {
      if (size < 0) {
            throw new NegativeArraySizeException("Array size cannot be
negative!");
      }

      int[] arr = new int[size];
      System.out.println("Array of size " + size + " created successfully!");

    } catch (NegativeArraySizeException ex) {
      System.out.println(ex.getMessage());
    }

    input.close();
  }
}
```

In this program, the user is prompted to enter the size of an array. The program then checks whether the size is negative using a try and catch block. If the size is negative, the program throws a NegativeArraySizeException with a custom error message. If the size is non-negative, the program creates an array of that size and displays a success message.

Note that we're explicitly throwing the exception when the input size is negative. This exception is then caught in the catch block and the message is displayed.

5. Write a program for example of multiple catch statements occurring in a program.

   Ans:- sure, here's an example program that demonstrates the use of multiple catch statements:

```
import java.util.Scanner;

public class MultipleCatchExample {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      try {
         System.out.print("Enter the numerator: ");
         int numerator = input.nextInt();
         System.out.print("Enter the denominator: ");
         int denominator = input.nextInt();
         int result = numerator / denominator;
         System.out.println("Result: " + result);
      } catch (ArithmeticException e) {
         System.out.println("Error: " + e.getMessage());
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      } finally {
         System.out.println("Program execution completed.");
      }
   }
}
```

   In this program, we first prompt the user to enter a numerator and denominator. We then try to divide the numerator by the denominator and print the result. If an **`ArithmeticException`** is thrown (i.e., if the user enters 0 as the denominator), we catch the exception and print an error message. If any other exception is thrown, we catch it and print an error message as well. Finally, we use a **`finally`** block to

print a message indicating that the program has completed execution, regardless of whether any exceptions were thrown.

6. Write a program to illustrate sub class exception precedence over base class.

Ans:- Here is a Java program that demonstrates sub class exception precedence over base class:

```
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            String str = null;
                        System.out.println(str.length()); // this will throw NullPointerException
                        int[] arr = new int[-1]; // this will throw NegativeArraySizeException
        } catch (Exception e) {
            System.out.println("Exception caught: " + e);
        } catch (NullPointerException e) {
            System.out.println("Caught NullPointerException");
        } catch (NegativeArraySizeException e) {
            System.out.println("Caught NegativeArraySizeException");
        }
    }
}
```

In this program, we intentionally cause two different exceptions: **`NullPointerException`** and **`NegativeArraySizeException`**. The first exception is a sub class of **`Exception`** and the second is a sub class of **`RuntimeException`**.

In the **`try`** block, we first create a **`null`** string and try to call the **`length()`** method on it, which will throw a **`NullPointerException`**. Then we create an array of negative size, which will throw a **`NegativeArraySizeException`**.

In the **`catch`** blocks, we catch the exceptions in reverse order of their hierarchy, i.e. first we catch the base class **`Exception`** and then the sub classes **`NullPointerException`** and **`NegativeArraySizeException`**.

When we run the program, the output will be:

```
Caught NullPointerException
```

This demonstrates that the sub class **`NullPointerException`** is caught before the base class **`Exception`**.

7. Write a program to illustrate usage of try/catch with the finally clause.
   Ans:-
   Sure, here's an example program that demonstrates the usage of try/catch with the finally clause:

```
import java.util.Scanner;

public class TryCatchFinallyExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        try {
            int num = scanner.nextInt();
            int result = 100 / num;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero");
        } finally {
            System.out.println("Finally block executed");
            scanner.close();
        }
    }
}
```

In this program, the user is prompted to enter a number. The program attempts to divide 100 by the number entered by the user, and if the number is 0, an **`ArithmeticException`** is thrown. The **`try`** block handles this exception by printing an error message. Regardless of whether an exception is thrown or not, the **`finally`** block is executed and the **`Scanner`** object is closed.

Here's an example of the program in action:

```
Enter a number: 5
Result: 20
Finally block executed

Enter a number: 0
```

Error: Division by zero
Finally block executed

In the first example, the program successfully divides 100 by 5 and prints the result, and the **finally** block is executed and closes the **Scanner**. In the second example, the user enters 0, which results in an **ArithmeticException** being thrown. The **catch** block handles the exception and prints an error message, and the **finally** block is executed and closes the **Scanner**.

8. Write a program to describe usage of throws clause.
   Ans:-

In Java, the **throws** keyword is used to declare that a method may throw a certain type of exception. This means that if a method calls another method which throws an exception, the calling method must handle or re-throw that exception.

Here is an example program that demonstrates the usage of the **throws** clause

```
public class ThrowsExample {

    public static void main(String[] args) {
        try {
            int result = divide(10, 0);
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static int divide(int a, int b) throws ArithmeticException {
        if (b == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return a / b;
    }
}
```

In this program, the **divide()** method takes two integer arguments and returns the result of dividing them. However, if the second argument is zero, it throws an **ArithmeticException** with a message "Division by zero". The method is declared with the **throws ArithmeticException** clause, which means that any method calling **divide()** must either handle the exception or declare it with its own **throws** clause.

In the **`main()`** method, we call **`divide()`** with arguments 10 and 0, which causes an exception to be thrown. Since we have a **`catch`** block that catches **`ArithmeticException`**, the program doesn't crash and prints the message "Exception caught: Division by zero". If we didn't catch the exception, it would propagate up the call stack until it was caught by another **`try-catch`** block or the program terminated.

9. Write a program for creation of user defined exceptions.

   Ans:-

   To create a user-defined exception in Java, we need to create a class that extends the pre-defined Exception class. Here's an example program that demonstrates the creation and usage of a custom exception class:

```
// Custom exception class
class MyException extends Exception {
  public MyException(String message) {
    super(message);
  }
}

// Main program
class Main {
  public static void main(String[] args) {
    int age = 15;

    try {
      if (age < 18) {
        throw new MyException("Age is less than 18");
      } else {
        System.out.println("You are eligible to vote");
      }
    } catch (MyException e) {
      System.out.println("Caught an exception: " + e.getMessage());
    }
  }
}
```

In this program, we have created a custom exception class **`MyException`** that extends the pre-defined **`Exception`** class. The **`MyException`** class has a constructor that takes a string message as an argument and passes it to the superclass constructor using the **`super()`** keyword.

In the main program, we have defined an age variable and checked if the age is less than 18. If the age is less than 18, we throw a **`MyException`** with a message "Age is less than 18". If the age is greater than or equal to 18, we print a message "You are eligible to vote".

We have used a try-catch block to catch the **`MyException`** thrown by the program. In the catch block, we print the exception message using the **`getMessage()`** method of the **`Exception`** class.

10. Write a program to create a file and write data into it using the methods OutputStream class.

Ans:- Here's an example program that creates a file named "output.txt" and writes data into it using the **`OutputStream`** class:

```
import java.io.FileOutputStream;

import java.io.OutputStream;


public class WriteToFileExample {

    public static void main(String[] args) {

        try {

            // create a new file named "output.txt"

            OutputStream os = new FileOutputStream("output.txt");


            // write some data into the file

            String data = "Hello, world!";

            os.write(data.getBytes());


            // close the output stream

            os.close();
```

```
            System.out.println("Data has been written to the file.");
        } catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

When you run this program, it will create a file named "output.txt" in the same directory as the program file and write the string "Hello, world!" to it. If an error occurs, such as if the file cannot be created or written to, the program will catch the exception and print an error message.

Unit IV

1. Write a program to give the example for button control.
   Ans:-

Sure, here's an example Java program that creates a window with a button:

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

public class ButtonExample {
    public static void main(String[] args) {
        // Create a window
        JFrame frame = new JFrame("Button Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);

        // Create a button
        JButton button = new JButton("Click me!");
```

```
        button.setBounds(80, 70, 140, 40);

        // Add an action listener to the button
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(frame, "Button clicked!");
            }
        });

        // Add the button to the window
        frame.add(button);

        // Show the window
        frame.setVisible(true);
    }
}
```

When you run this program, it will display a window with a button labeled "Click me!". When the button is clicked, a message box will appear with the text "Button clicked!".

2. Write a program to give the example for panel control.

Ans:- Here's an example program in Java that demonstrates the use of **`JPanel`** control:

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JLabel;

public class PanelExample {
    public static void main(String[] args) {
        // create a new JFrame window
        JFrame frame = new JFrame("Panel Example");

        // create a new JPanel
        JPanel panel = new JPanel();

        // create a new JButton and add it to the panel
        JButton button = new JButton("Click me!");
        panel.add(button);
```

```
        // create a new JLabel and add it to the panel
        JLabel label = new JLabel("Hello, world!");
        panel.add(label);

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

        // set the size of the frame and make it visible
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

In this program, we create a **`JFrame`** window and a **`JPanel`** control. We add a **`JButton`** and a **`JLabel`** to the panel, and then add the panel to the frame. Finally, we set the size of the frame and make it visible.

3. Write a program that will display check boxes and option buttons they are numbered from1 to. Use a textbox to display the number of those corresponding boxes or buttons checked.

   Ans:-

   Here's an example program that displays check boxes and option buttons and counts the number of boxes or buttons checked:

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

public class CheckboxAndRadioButtonExample extends JFrame implements
ActionListener {
    private JCheckBox[] checkBoxes;
    private JRadioButton[] radioButtons;
    private ButtonGroup radioButtonGroup;
    private JTextField textField;

    public CheckboxAndRadioButtonExample() {
        // Set up the window
        super("Checkbox and Radio Button Example");
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLayout(new BorderLayout());

        // Create the check boxes
        JPanel checkBoxPanel = new JPanel();
        checkBoxPanel.setBorder(BorderFactory.createTitledBorder("Check
Boxes"));
        checkBoxes = new JCheckBox[5];
        for (int i = 0; i < checkBoxes.length; i++) {
            checkBoxes[i] = new JCheckBox("Checkbox " + (i+1));
            checkBoxes[i].addActionListener(this);
            checkBoxPanel.add(checkBoxes[i]);
        }
        add(checkBoxPanel, BorderLayout.NORTH);

        // Create the radio buttons
        JPanel radioButtonPanel = new JPanel();
        radioButtonPanel.setBorder(BorderFactory.createTitledBorder("Radio
Buttons"));
        radioButtons = new JRadioButton[5];
        radioButtonGroup = new ButtonGroup();
        for (int i = 0; i < radioButtons.length; i++) {
            radioButtons[i] = new JRadioButton("Radio Button " + (i+1));
            radioButtons[i].addActionListener(this);
            radioButtonGroup.add(radioButtons[i]);
            radioButtonPanel.add(radioButtons[i]);
        }
        add(radioButtonPanel, BorderLayout.CENTER);

        // Create the text field to display the count
        textField = new JTextField();
        textField.setEditable(false);
        add(textField, BorderLayout.SOUTH);
    }

    public void actionPerformed(ActionEvent e) {
```

```
        // Count the number of check boxes and radio buttons that are selected
        int count = 0;
        for (int i = 0; i < checkBoxes.length; i++) {
            if (checkBoxes[i].isSelected()) {
                count++;
            }
        }
        for (int i = 0; i < radioButtons.length; i++) {
            if (radioButtons[i].isSelected()) {
                count++;
            }
        }
        // Update the text field with the count
        textField.setText(count + " boxes/buttons selected");
    }

    public static void main(String[] args) {
        CheckboxAndRadioButtonExample example = new
CheckboxAndRadioButtonExample();
        example.setVisible(true);
    }
}
```

This program creates a window with five check boxes and five radio buttons. When a box or button is checked, the **`actionPerformed()`** method is called, which counts the number of check boxes and radio buttons that are selected and updates a text field with the count.

4. Write a program to create a simple calculator.

   Ans:-

```
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double num1, num2, result;
        char operator;
        System.out.print("Enter first number: ");
        num1 = scanner.nextDouble();
        System.out.print("Enter second number: ");
```

```
        num2 = scanner.nextDouble();
        System.out.print("Enter an operator (+, -, *, /): ");
        operator = scanner.next().charAt(0);
        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                if (num2 == 0) {
                    System.out.println("Error: division by zero!");
                    return;
                }
                result = num1 / num2;
                break;
            default:
                System.out.println("Error: invalid operator!");
                return;
        }
        System.out.println(num1 + " " + operator + " " + num2 + " = " + result);
    }
}
```

5. Write a program as above with a combo box and list boxes instead.
   Ans:-
   Sure, here's an example Java program that uses a combo box and two list boxes:

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

public class ComboBoxAndListBoxesExample extends JFrame implements
ActionListener {
```

```
private JComboBox<String> comboBox;
private JList<String> leftList, rightList;
private DefaultListModel<String> leftListModel, rightListModel;

public ComboBoxAndListBoxesExample() {
    setTitle("ComboBox and List Boxes Example");
    setSize(400, 300);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Create and add a combo box to the top panel
    JPanel topPanel = new JPanel();
    topPanel.setLayout(new FlowLayout());
    String[] items = {"Item 1", "Item 2", "Item 3", "Item 4"};
    comboBox = new JComboBox<>(items);
    comboBox.addActionListener(this);
    topPanel.add(new JLabel("Select an item: "));
    topPanel.add(comboBox);
    add(topPanel, BorderLayout.NORTH);

    // Create and add two list boxes to the center panel
    JPanel centerPanel = new JPanel();
    centerPanel.setLayout(new GridLayout(1, 2, 10, 0));
    leftListModel = new DefaultListModel<>();
    leftListModel.addElement("Option 1");
    leftListModel.addElement("Option 2");
    leftListModel.addElement("Option 3");
    leftListModel.addElement("Option 4");
    leftListModel.addElement("Option 5");
    leftList = new JList<>(leftListModel);
    JScrollPane leftScrollPane = new JScrollPane(leftList);
    centerPanel.add(leftScrollPane);
    JButton addButton = new JButton("Add >>");
    addButton.addActionListener(this);
    centerPanel.add(addButton);
    rightListModel = new DefaultListModel<>();
    rightList = new JList<>(rightListModel);
```

```
        JScrollPane rightScrollPane = new JScrollPane(rightList);
        centerPanel.add(rightScrollPane);
        add(centerPanel, BorderLayout.CENTER);
    }

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();

        if (source == comboBox) {
            String selectedItem = (String) comboBox.getSelectedItem();
            JOptionPane.showMessageDialog(this, "You selected: " + selectedItem);
        } else if (source instanceof JButton) {
            int[] selectedIndices = leftList.getSelectedIndices();
            for (int i = selectedIndices.length - 1; i >= 0; i--) {
                String selectedValue = leftListModel.getElementAt(selectedIndices[i]);
                leftListModel.removeElementAt(selectedIndices[i]);
                rightListModel.addElement(selectedValue);
            }
        }
    }

    public static void main(String[] args) {
        ComboBoxAndListBoxesExample example = new
ComboBoxAndListBoxesExample();
        example.setVisible(true);
    }

}
```

In this program, we create a JFrame window with a combo box and two list boxes. The combo box allows the user to select an item from a list, and the list boxes allow the user to move items between them. When the user selects an item from the combo box, a message dialog is displayed showing the selected item. When the user clicks the "Add >>" button, the selected items from the left list box are moved to the right list box.

6. Write a program that displays the x and y position of the cursor movement using Mouse.

   Ans:-

   Here's an example Java program that displays the x and y position of the cursor movement using Mouse:

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

public class MousePositionExample extends JFrame implements
MouseMotionListener {

    private JLabel positionLabel;

    public MousePositionExample() {
        setTitle("Mouse Position Example");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a label to display the position
        positionLabel = new JLabel("Mouse position: (0, 0)");
        add(positionLabel, BorderLayout.SOUTH);

        // Add a mouse motion listener to the frame
        addMouseMotionListener(this);
    }

    public void mouseMoved(MouseEvent e) {
        // Update the position label with the current position
        int x = e.getX();
        int y = e.getY();
        positionLabel.setText("Mouse position: (" + x + ", " + y + ")");
    }

    public void mouseDragged(MouseEvent e) {
        // Do nothing
    }

    public static void main(String[] args) {
        MousePositionExample example = new MousePositionExample();
        example.setVisible(true);
```

```
    }

}
```

In this program, we create a JFrame window and add a label to display the position of the mouse cursor. We then add a MouseMotionListener to the frame to detect mouse movement. When the mouse is moved, the mouseMoved method is called, which updates the position label with the current x and y coordinates of the mouse cursor. The mouseDragged method is empty, as we don't need to do anything when the mouse is dragged.

7. Write a program to create a canvas.

   Ans:-

   Here's an example Java program that creates a canvas:

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

public class CanvasExample extends JFrame {

    private Canvas canvas;

    public CanvasExample() {
        setTitle("Canvas Example");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a canvas and add it to the frame
        canvas = new Canvas();
        add(canvas);
    }

    public static void main(String[] args) {
        CanvasExample example = new CanvasExample();
        example.setVisible(true);
    }

}
```

In this program, we create a JFrame window and add a canvas to it. The canvas is created using the Canvas class, which is part of the java.awt package. The canvas is added to the frame using the add method.

Note that the canvas is a blank component, so you won't see anything on it until you draw something on it using graphics. To do that, you can get the graphics object of the canvas and use it to draw lines, shapes, text, and images.

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

public class CanvasExample extends JFrame {

    public CanvasExample() {
        setTitle("Canvas Example");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a canvas and add it to the frame
        MyCanvas canvas = new MyCanvas();
        add(canvas);
    }

    public static void main(String[] args) {
        CanvasExample example = new CanvasExample();
        example.setVisible(true);
    }

}

class MyCanvas extends Canvas {

    public MyCanvas() {
        setBackground(Color.WHITE);
    }

    public void paint(Graphics g) {
        // Draw some shapes on the canvas
        g.setColor(Color.RED);
```

```
        g.fillRect(50, 50, 100, 100);
        g.setColor(Color.BLUE);
        g.fillOval(200, 50, 100, 100);
        g.setColor(Color.GREEN);
        g.drawLine(50, 200, 300, 200);
    }

}
```

In this program, we create a JFrame window and add a canvas to it. The canvas is implemented as a separate class called **`MyCanvas`** that extends the **`Canvas`** class. In the **`MyCanvas`** constructor, we set the background color of the canvas to white. We then override the **`paint`** method to draw some shapes on the canvas using the **`Graphics`** object passed to it as a parameter.

In the main program, we create an instance of the **`CanvasExample`** class and make it visible. When the program is run, a window is displayed containing the canvas with the shapes drawn on it.

8. Write a program that displays the x and y position of the cursor movement using Keyboard.
   Ans:-

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

public class KeyboardPositionExample extends JFrame implements KeyListener {

    private JLabel positionLabel;
    private int x = 0, y = 0;

    public KeyboardPositionExample() {
        setTitle("Keyboard Position Example");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a label to display the position
        positionLabel = new JLabel("Cursor position: (" + x + ", " + y + ")");
```

```
    add(positionLabel, BorderLayout.SOUTH);

    // Add a key listener to the frame
    addKeyListener(this);
    setFocusable(true);
    requestFocusInWindow();
}

public void keyPressed(KeyEvent e) {
    // Move the cursor depending on the arrow key pressed
    switch (e.getKeyCode()) {
        case KeyEvent.VK_LEFT:
            x--;
            break;
        case KeyEvent.VK_RIGHT:
            x++;
            break;
        case KeyEvent.VK_UP:
            y--;
            break;
        case KeyEvent.VK_DOWN:
            y++;
            break;
    }
    // Update the position label with the new position
    positionLabel.setText("Cursor position: (" + x + ", " + y + ")");
}

public void keyReleased(KeyEvent e) {
    // Do nothing
}

public void keyTyped(KeyEvent e) {
    // Do nothing
}

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

```
        KeyboardPositionExample example = new KeyboardPositionExample();
        example.setVisible(true);
    }

}
```

In this program, we create a JFrame window and add a label to display the position of the cursor. We then add a KeyListener to the frame to detect arrow key presses. When an arrow key is pressed, the keyPressed method is called, which updates the x and y coordinates of the cursor depending on which arrow key was pressed. The position label is then updated with the new position.

Note that the frame needs to be focused in order for the arrow key presses to be detected, which is why we call setFocusable and requestFocusInWindow in the constructor.

9. Write a program to create a text box control.
   Ans:-
   Here's an example Java program that creates a text box control using the JTextField class:

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

public class TextBoxExample extends JFrame {

    private JTextField textField;

    public TextBoxExample() {
        setTitle("Text Box Example");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a text field and add it to the frame
        textField = new JTextField(20);
        add(textField, BorderLayout.CENTER);

        // Create a button to print the text in the text field
        JButton button = new JButton("Print");
        button.addActionListener(e -> System.out.println(textField.getText()));
```

```
        add(button, BorderLayout.SOUTH);
    }

    public static void main(String[] args) {
        TextBoxExample example = new TextBoxExample();
        example.setVisible(true);
    }

}
```

In this program, we create a JFrame window and add a JTextField to it. The JTextField is created with a width of 20 columns, which determines its initial size. We also create a JButton to print the text in the text field when clicked.

When the button is clicked, the ActionListener is called, which uses the getText method of the JTextField to retrieve the text entered in the text box. This text is then printed to the console using the println method.

Note that we're using lambda expressions to define the ActionListener for the button, which is a shorthand way of defining a method with a single parameter. If you're using an older version of Java that doesn't support lambda expressions, you can define a separate class that implements the ActionListener interface instead.

```
import javax.swing.*;

public class TextBoxExample extends JFrame {

    public TextBoxExample() {
        setTitle("Text Box Example");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a text box and add it to the frame
        JTextField textField = new JTextField("Enter text here");
        add(textField);
    }

    public static void main(String[] args) {
        TextBoxExample example = new TextBoxExample();
```

```
        example.setVisible(true);
    }

}
```

In this program, we create a JFrame window and add a text box control to it using the **`JTextField`** class. We set the default text of the text box to "Enter text here".

When the program is run, a window is displayed containing the text box control. The user can enter text into the box, and the entered text can be retrieved using the **`getText()`** method of the **`JTextField`** class.

10. Write a program to create an analog clock.

    Ans:-

    Here's an example Java program that creates an analog clock using the **`java.util.Calendar`** and **`java.awt.Graphics`** classes:

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

public class AnalogClockExample extends JPanel {

    private int centerX = 150;
    private int centerY = 150;
    private int radius = 100;
    private Calendar calendar;

    public AnalogClockExample() {
        calendar = Calendar.getInstance();
        Timer timer = new Timer(1000, e -> {
            calendar.setTimeInMillis(System.currentTimeMillis());
            repaint();
        });
        timer.start();
    }

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

```
        // Draw clock face
        g.setColor(Color.WHITE);
        g.fillOval(centerX - radius, centerY - radius, radius * 2, radius * 2);
        g.setColor(Color.BLACK);
        g.drawOval(centerX - radius, centerY - radius, radius * 2, radius * 2);
        // Draw hour markers
        for (int i = 1; i <= 12; i++) {
            double angle = Math.PI / 6 * i;
            int x = (int) (centerX + (radius - 10) * Math.sin(angle));
            int y = (int) (centerY - (radius - 10) * Math.cos(angle));
            g.drawString(Integer.toString(i), x, y);
        }
        // Draw hour hand
        int hour = calendar.get(Calendar.HOUR);
        double hourAngle = Math.PI / 6 * hour + Math.PI / 360 *
calendar.get(Calendar.MINUTE);
        int hourX = (int) (centerX + (radius - 50) * Math.sin(hourAngle));
        int hourY = (int) (centerY - (radius - 50) * Math.cos(hourAngle));
        g.setColor(Color.BLUE);
        g.drawLine(centerX, centerY, hourX, hourY);
        // Draw minute hand
        int minute = calendar.get(Calendar.MINUTE);
        double minuteAngle = Math.PI / 30 * minute;
        int minuteX = (int) (centerX + (radius - 30) * Math.sin(minuteAngle));
        int minuteY = (int) (centerY - (radius - 30) * Math.cos(minuteAngle));
        g.setColor(Color.RED);
        g.drawLine(centerX, centerY, minuteX, minuteY);
        // Draw second hand
        int second = calendar.get(Calendar.SECOND);
        double secondAngle = Math.PI / 30 * second;
        int secondX = (int) (centerX + (radius - 10) * Math.sin(secondAngle));
        int secondY = (int) (centerY - (radius - 10) * Math.cos(secondAngle));
        g.setColor(Color.GREEN);
        g.drawLine(centerX, centerY, secondX, secondY);
    }

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

```
        JFrame frame = new JFrame("Analog Clock Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 300);
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.add(new AnalogClockExample());
        frame.setVisible(true);
    }

}
```

In this program, we create a **`JPanel`** that draws the clock face and hands using the **`Graphics`** class. We use a **`Timer`** to update the time displayed on the clock every second.

To draw the clock face, we use the **`fillOval`** and **`drawOval`** methods to draw a circle with the specified radius. We then draw the hour markers around the perimeter of the circle using the **`drawString`** method

11. Write a program for Temperature convertor.

    Ans:-

    Here's an example Java program that converts temperature between Celsius and Fahrenheit:

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

public class TemperatureConverter extends JFrame {

    private JLabel inputLabel, outputLabel;
    private JTextField inputField, outputField;
    private JButton celsiusButton, fahrenheitButton;

    public TemperatureConverter() {
        setTitle("Temperature Converter");
        setSize(300, 150);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
inputLabel = new JLabel("Enter temperature:");
outputLabel = new JLabel("Converted temperature:");
inputField = new JTextField(10);
outputField = new JTextField(10);
outputField.setEditable(false);
celsiusButton = new JButton("Celsius");
fahrenheitButton = new JButton("Fahrenheit");

JPanel panel = new JPanel();
panel.setLayout(new GridLayout(3, 2));
panel.add(inputLabel);
panel.add(inputField);
panel.add(outputLabel);
panel.add(outputField);
panel.add(celsiusButton);
panel.add(fahrenheitButton);
add(panel);

celsiusButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      String input = inputField.getText().trim();
      if (!input.isEmpty()) {
         try {
            double fahrenheit = Double.parseDouble(input);
            double celsius = (fahrenheit - 32) * 5 / 9;
            outputField.setText(String.format("%.2f", celsius));
         } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(TemperatureConverter.this,
                  "Invalid input. Please enter a number.",
                  "Error", JOptionPane.ERROR_MESSAGE);
         }
      }
   }
});

fahrenheitButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
```

```
                String input = inputField.getText().trim();
                if (!input.isEmpty()) {
                    try {
                        double celsius = Double.parseDouble(input);
                        double fahrenheit = celsius * 9 / 5 + 32;
                        outputField.setText(String.format("%.2f", fahrenheit));
                    } catch (NumberFormatException ex) {
                        JOptionPane.showMessageDialog(TemperatureConverter.this,
                                "Invalid input. Please enter a number.",
                                "Error", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });
    }

    public static void main(String[] args) {
        TemperatureConverter converter = new TemperatureConverter();
        converter.setVisible(true);
    }

}
```

In this program, we create a JFrame window with two text fields for input and output, and two buttons for selecting the conversion direction (Celsius to Fahrenheit or Fahrenheit to Celsius).

When the user enters a temperature in either Celsius or Fahrenheit and clicks the corresponding button, the program converts the temperature to the other unit of measurement and displays the result in the output text field.

To convert Celsius to Fahrenheit, we use the formula `F = (C * 9/5) + 32`, where F is the temperature in Fahrenheit and C is the temperature in Celsius.

To convert Fahrenheit to Celsius, we use the formula `C = (F - 32) * 5/9`, where C is the temperature in Celsius and F is the temperature in Fahrenheit.

We use exception handling to catch any invalid input entered by the user and display an error message in a dialog box.

12. Write a program to calculate

Unit V

1. Write an Applet program that automatically displays the text with Font Style, Font type .

   Ans:-

   Here's an example Java applet program that displays text with a specified font style and font type:

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

public class FontApplet extends Applet {
   public void paint(Graphics g) {
      Font font = new Font("Times New Roman", Font.BOLD + Font.ITALIC, 24);
      g.setFont(font);
      g.drawString("Hello, world!", 50, 50);
   }
}
```

In this program, we create an applet that overrides the **`paint`** method to draw the text "Hello, world!" with a specified font.

We create a **`Font`** object with the font type "Times New Roman", a combination of bold and italic font styles, and a font size of 24. We then set this font for the graphics context by calling the **`setFont`** method on the **`Graphics`** object. Finally, we draw the text "Hello, world!" at the position (50, 50) by calling the **`drawString`** method on the **`Graphics`** object.

When you run this program as an applet, it will automatically display the text with the specified font style and font type.

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

public class FontApplet extends Applet {
```

```
    public void init() {
        // Set the font style and font type
        Font font = new Font("Arial", Font.BOLD | Font.ITALIC, 24);
        setFont(font);
    }

    public void paint(Graphics g) {
        // Draw the text with the custom font
        g.drawString("Welcome to FontApplet", 50, 50);
    }
}
```

In this program, we create an Applet and override its **`init()`** and **`paint()`** methods.

In the **`init()`** method, we set the font style and font type using the **`Font`** class. We create a new **`Font`** object with the font family name "Arial", a bold and italic font style, and a font size of 24. We then call the **`setFont()`** method of the Applet to set the font for the entire Applet.

In the **`paint()`** method, we draw the text "Welcome to FontApplet" on the Applet using the **`drawString()`** method of the **`Graphics`** class. The custom font that we set in the **`init()`** method will be used to display the text.

When you run this program as an Applet in a web browser, it will display the text "Welcome to FontApplet" with the custom font style and font type that we set.

2. Write a program that has a menubar and also a quit option and if the user clicks the quiet option the applet should quit.
   Ans:-
   Here's an example Java program that creates a simple window with a menubar that includes a "Quit" option. When the user clicks the "Quit" option, the program will exit:

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

public class MenuBarExample extends JFrame implements ActionListener {

    public MenuBarExample() {
        // Create the menubar and menu items
```

```
        JMenuBar menubar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenuItem quitMenuItem = new JMenuItem("Quit");
        quitMenuItem.addActionListener(this);

        // Add the menu items to the menubar
        fileMenu.add(quitMenuItem);
        menubar.add(fileMenu);

        // Set the menubar for the window
        setJMenuBar(menubar);

        // Set the properties for the window
        setTitle("MenuBar Example");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {
        // If the Quit menu item is clicked, exit the program
        if (e.getActionCommand().equals("Quit")) {
            System.exit(0);
        }
    }

    public static void main(String[] args) {
        // Create and display the window
        MenuBarExample window = new MenuBarExample();
        window.setVisible(true);
    }
}
```

In this program, we create a window using the **`JFrame`** class, and override the **`actionPerformed()`** method of the **`ActionListener`** interface.

In the constructor of the **`MenuBarExample`** class, we create a **`JMenuBar`** object and add a **`JMenu`** object called "File" to it. We then create a **`JMenuItem`** object called "Quit" and add it to the

"File" menu. We also set the **`actionListener`** of the "Quit" menu item to be the current object (**`this`**). Finally, we set the menubar for the window using the **`setJMenuBar()`** method.

In the **`actionPerformed()`** method, we check if the "Quit" menu item was clicked by comparing the action command of the event to the string "Quit". If it was clicked, we exit the program using the **`System.exit()`** method.

In the **`main()`** method, we create an instance of the **`MenuBarExample`** class and make it visible.

3. Write an Applet program that automatically displays the text with Font Style, Font type Using getParameter Method.
   Ans:-
   Here's an example Java Applet program that displays text with a custom font style and font type, using the **`getParameter()`** method to get the font information from the HTML code that embeds the Applet:

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

public class FontApplet extends Applet {

    public void init() {
        // Get the font style and font type from the HTML code
        String fontStyle = getParameter("fontStyle");
        String fontType = getParameter("fontType");

        // Set the font based on the parameters
        Font font = new Font(fontType, Font.PLAIN, 24);
        if (fontStyle.equals("bold")) {
            font = new Font(fontType, Font.BOLD, 24);
        } else if (fontStyle.equals("italic")) {
            font = new Font(fontType, Font.ITALIC, 24);
        } else if (fontStyle.equals("bolditalic")) {
            font = new Font(fontType, Font.BOLD | Font.ITALIC, 24);
        }
        setFont(font);
    }

    public void paint(Graphics g) {
        // Draw the text with the custom font
```

```
        g.drawString("Welcome to FontApplet", 50, 50);
    }
}
```

In this program, we create an Applet and override its **`init()`** and **`paint()`** methods.

In the **`init()`** method, we get the font style and font type from the HTML code using the **`getParameter()`** method. We then set the font based on the values of the parameters. We create a new **`Font`** object with the font family name specified by the **`fontType`** parameter, a font size of 24, and a font style that depends on the **`fontStyle`** parameter. We use the **`setFont()`** method of the Applet to set the font for the entire Applet.

In the **`paint()`** method, we draw the text "Welcome to FontApplet" on the Applet using the **`drawString()`** method of the **`Graphics`** class. The custom font that we set in the **`init()`** method will be used to display the text.

When you embed this Applet in an HTML page, you can pass in the font style and font type parameters in the **`<applet>`** tag, like this:

```
<applet code="FontApplet.class" width="300" height="100">
    <param name="fontStyle" value="bolditalic">
    <param name="fontType" value="Arial">
</applet>
```

This will display the text "Welcome to FontApplet" with a bold italic font style and the font family "Arial".

Here's an example Java Applet program that displays text with a custom font style and font type using the **`getParameter()`** method to retrieve the font information from the HTML page:

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

public class FontApplet extends Applet {

    private Font font;

    public void init() {
        // Get the font style and font type from the HTML page
        String fontStyle = getParameter("fontStyle");
        String fontType = getParameter("fontType");
```

```
        // Set the font style and font type
        font = new Font(fontType, Font.BOLD | Font.ITALIC, 24);
        setFont(font);
    }

    public void paint(Graphics g) {
        // Draw the text with the custom font
        g.drawString("Welcome to FontApplet", 50, 50);
    }
}
```

In this program, we create an Applet and override its **`init()`** and **`paint()`** methods.

In the **`init()`** method, we use the **`getParameter()`** method to retrieve the font style and font type from the HTML page. The **`getParameter()`** method is used to retrieve the values of parameters passed to the Applet from the HTML page.

We then create a new **`Font`** object with the font family name specified in **`fontType`**, a bold and italic font style, and a font size of 24. We set this font for the entire Applet using the **`setFont()`** method.

In the **`paint()`** method, we draw the text "Welcome to FontApplet" on the Applet using the **`drawString()`** method of the **`Graphics`** class. The custom font that we set in the **`init()`** method will be used to display the text.

When you run this program as an Applet in a web browser with an HTML page that specifies the **`fontStyle`** and **`fontType`** parameters, it will display the text "Welcome to FontApplet" with the custom font style and font type that we set. For example, the HTML code might look like this:

To run an Applet program in Eclipse, you can follow these steps:

1. Open Eclipse and create a new Java project by selecting File > New > Java Project from the menu.
2. Give your project a name and click Finish.
3. Create a new Java class by right-clicking on the src folder in the Package Explorer and selecting New > Class.
4. Give your class a name and make sure the box next to "public static void main(String[] args)" is unchecked.
5. Copy the code for the Applet program into your class.
6. To run the program as an Applet, you need to create an HTML file that references your Applet class. Right-click on the project folder in the Package Explorer and select New > File. Name the file "index.html" and click Finish.
7. Copy the following code into your HTML file:

```
<!DOCTYPE html>

<html>

<head>

        <title>My Applet</title>

</head>

<body>

        <applet code="MyApplet.class" width="300" height="200">

        </applet>

</body>

</html>
```

Replace "MyApplet" with the name of your Applet class.

8. Save your HTML file and right-click on it in the Package Explorer. Select Run As > Java Applet.

Your Applet should now run in the Applet viewer within Eclipse.

4. Write a program that displays the menu bar and when You click the options it has to display a dialog box stating which option has been clicked.
   Ans:-

   Here's an example Java program that displays a menu bar and shows a dialog box when a menu item is clicked:

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

public class MenuDemo extends JFrame implements ActionListener {

   private JMenuBar menuBar;
   private JMenu fileMenu;
```

```
private JMenuItem openItem;
private JMenuItem saveItem;
private JMenuItem exitItem;

public MenuDemo() {
    // Create the menu bar
    menuBar = new JMenuBar();

    // Create the file menu
    fileMenu = new JMenu("File");

    // Create the menu items
    openItem = new JMenuItem("Open");
    saveItem = new JMenuItem("Save");
    exitItem = new JMenuItem("Exit");

    // Add the menu items to the file menu
    fileMenu.add(openItem);
    fileMenu.add(saveItem);
    fileMenu.addSeparator();
    fileMenu.add(exitItem);

    // Add the file menu to the menu bar
    menuBar.add(fileMenu);

    // Add action listeners to the menu items
    openItem.addActionListener(this);
    saveItem.addActionListener(this);
    exitItem.addActionListener(this);

    // Set the menu bar for the JFrame
    setJMenuBar(menuBar);

    // Set the JFrame properties
    setTitle("Menu Demo");
    setSize(300, 200);
    setLocationRelativeTo(null);
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {
        // Show a dialog box indicating which menu item was clicked
        if (e.getSource() == openItem) {
            JOptionPane.showMessageDialog(this, "Open clicked");
        } else if (e.getSource() == saveItem) {
            JOptionPane.showMessageDialog(this, "Save clicked");
        } else if (e.getSource() == exitItem) {
            JOptionPane.showMessageDialog(this, "Exit clicked");
            System.exit(0);
        }
    }

    public static void main(String[] args) {
        MenuDemo menuDemo = new MenuDemo();
        menuDemo.setVisible(true);
    }
}
```

In this program, we create a **`JFrame`** with a menu bar that contains a file menu. The file menu has three menu items: Open, Save, and Exit.

We add action listeners to the menu items to detect when they are clicked. When a menu item is clicked, we show a dialog box using the **`JOptionPane.showMessageDialog()`** method to indicate which menu item was clicked.

To run this program, compile and run the **`MenuDemo`** class. When the program starts, a window with a menu bar containing a file menu will be displayed. Clicking any of the menu items will display a dialog box indicating which item was clicked.

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

public class MenuBarDemo extends JFrame implements ActionListener {

    public MenuBarDemo() {
        // Set the title of the window
```

```
    super("Menu Bar Demo");

    // Create the menu bar
    JMenuBar menuBar = new JMenuBar();

    // Create the file menu
    JMenu fileMenu = new JMenu("File");

    // Create the options for the file menu
    JMenuItem newMenuItem = new JMenuItem("New");
    JMenuItem openMenuItem = new JMenuItem("Open");
    JMenuItem saveMenuItem = new JMenuItem("Save");
    JMenuItem quitMenuItem = new JMenuItem("Quit");

    // Add the options to the file menu
    fileMenu.add(newMenuItem);
    fileMenu.add(openMenuItem);
    fileMenu.add(saveMenuItem);
    fileMenu.addSeparator();
    fileMenu.add(quitMenuItem);

    // Add the file menu to the menu bar
    menuBar.add(fileMenu);

    // Set the menu bar for the window
    setJMenuBar(menuBar);

    // Add action listeners to the menu items
    newMenuItem.addActionListener(this);
    openMenuItem.addActionListener(this);
    saveMenuItem.addActionListener(this);
    quitMenuItem.addActionListener(this);

    // Set the size of the window and show it
    setSize(300, 200);
    setVisible(true);
}
```

```
    public void actionPerformed(ActionEvent e) {
        // Show a dialog box when a menu item is clicked
        JOptionPane.showMessageDialog(this, "You clicked " +
e.getActionCommand());

        // Quit the program if the Quit option is clicked
        if (e.getActionCommand().equals("Quit")) {
            System.exit(0);
        }
    }

    public static void main(String[] args) {
        // Create a new instance of the menu bar demo
        new MenuBarDemo();
    }

}
```

5. Write a Java program compute factorial value using Applet.

   Ans:-

   Here's an example Java program that computes the factorial of a number using an Applet:

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

public class FactorialApplet extends Applet implements ActionListener {

    private TextField inputField;
    private Label outputLabel;
    private Button calculateButton;

    public void init() {
        // Create the user interface
        inputField = new TextField(10);
        outputLabel = new Label("Factorial of the number will appear here");
        calculateButton = new Button("Calculate");
```

```
        add(new Label("Enter a number: "));
        add(inputField);
        add(calculateButton);
        add(outputLabel);

        // Add action listener to the calculate button
        calculateButton.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        // Get the input number
        int inputNumber = Integer.parseInt(inputField.getText());

        // Calculate the factorial of the number
        int factorial = 1;
        for (int i = 1; i <= inputNumber; i++) {
            factorial *= i;
        }

        // Display the factorial in the output label
        outputLabel.setText("Factorial of " + inputNumber + " is " + factorial);
    }
}
```

In this program, we create a simple user interface consisting of a text field, a button, and a label. When the user clicks the button, the program calculates the factorial of the input number and displays the result in the label.

To run this program, you can create an HTML file that references the Applet class:

```
<!DOCTYPE html>
<html>
<head>
        <title>Factorial Applet</title>
</head>
<body>
        <applet code="FactorialApplet.class" width="300" height="200">
        </applet>
```

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

Save this HTML file and the Java file in the same directory, and then open the HTML file in a web browser. The Applet should be displayed in the web page, and you can enter a number and click the "Calculate" button to compute its factorial.

6. Write a program for passing parameters using Applet.
   Ans:-
   Here's an example Java program that passes parameters to an Applet:

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

public class ParameterApplet extends Applet {

    private String message;

    public void init() {
        // Get the value of the "message" parameter
        message = getParameter("message");

        // Create a label with the message text
        Label messageLabel = new Label(message);

        // Add the label to the Applet's user interface
        add(messageLabel);
    }
}
```

In this program, we create an Applet that displays a message passed in as a parameter. We use the **`getParameter`** method to get the value of the "message" parameter, and then create a **`Label`** with that text. Finally, we add the label to the Applet's user interface.

To pass parameters to this Applet, we need to include them in the HTML code that references the Applet. Here's an example HTML file that passes a "message" parameter:

```
<!DOCTYPE html>
<html>
<head>
```

```
        <title>Parameter Applet</title>
</head>
<body>
        <applet code="ParameterApplet.class" width="300" height="200">
                <param name="message" value="Hello, world!">
        </applet>
</body>
</html>
```

In this HTML file, we include a **`param`** tag inside the **`applet`** tag, with the **`name`** attribute set to "message" and the **`value`** attribute set to "Hello, world!". When the Applet is loaded in the web page, it will get the value of the "message" parameter and display it in a label.

7. Write a program to create an Applet life cycle.

   Ans:-

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

public class AppletLifeCycle extends Applet {

   public void init() {
      // Called when the Applet is initialized
      System.out.println("Initializing Applet...");
   }

   public void start() {
      // Called when the Applet is started
      System.out.println("Starting Applet...");
   }

   public void paint(Graphics g) {
      // Called when the Applet needs to be painted
      System.out.println("Painting Applet...");
      g.drawString("Hello, world!", 50, 50);
   }

   public void stop() {
      // Called when the Applet is stopped
      System.out.println("Stopping Applet...");
```

```
    }

    public void destroy() {
        // Called when the Applet is destroyed
        System.out.println("Destroying Applet...");
    }
}
```

In this program, we create an Applet and override several of its methods to demonstrate its life cycle. The **`init`** method is called when the Applet is initialized, the **`start`** method is called when the Applet is started, the **`paint`** method is called when the Applet needs to be painted, the **`stop`** method is called when the Applet is stopped, and the **`destroy`** method is called when the Applet is destroyed.

When the Applet is loaded in a web page, its **`init`** method is called first. This is where we can initialize any variables or resources needed by the Applet. Then, the **`start`** method is called to start the Applet. The **`paint`** method is called whenever the Applet needs to be painted, such as when it is first displayed or when it is resized. In this method, we can draw graphics or display text on the Applet's user interface. The **`stop`** method is called when the Applet is stopped, such as when the user navigates away from the web page or closes the browser. Finally, the **`destroy`** method is called when the Applet is destroyed, such as when the web page is closed or the browser is shut down. In this method, we can clean up any resources used by the Applet.

By printing messages to the console in each of these methods, we can see the order in which they are called and better understand the life cycle of an Applet.

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

public class LifeCycleApplet extends Applet {

    // Initialize the Applet
    public void init() {
        System.out.println("Initializing Applet...");
    }

    // Start the Applet
    public void start() {
        System.out.println("Starting Applet...");
    }

    // Paint the Applet
    public void paint(Graphics g) {
```

```
        System.out.println("Painting Applet...");
        g.drawString("Hello, world!", 50, 50);
    }

    // Stop the Applet
    public void stop() {
        System.out.println("Stopping Applet...");
    }

    // Destroy the Applet
    public void destroy() {
        System.out.println("Destroying Applet...");
    }

}
```

In this program, we define an Applet with the following methods:

- **`init()`**: This method is called when the Applet is initialized. It is typically used to perform any necessary setup.
- **`start()`**: This method is called when the Applet is started, either initially or after being stopped. It is typically used to start any animations or other processes that should run while the Applet is active.
- **`paint()`**: This method is called when the Applet needs to be repainted. It is typically used to draw the Applet's user interface.
- **`stop()`**: This method is called when the Applet is stopped, either by the user or by the system. It is typically used to stop any animations or other processes that are running.
- **`destroy()`**: This method is called when the Applet is about to be destroyed. It is typically used to perform any necessary cleanup.

When you run this Applet, the **`init()`** method will be called first, followed by the **`start()`** method. The **`paint()`** method will be called next, to draw the "Hello, world!" message on the screen. If you resize or minimize the window, the **`stop()`** method will be called, and if you close the window or navigate away from the page, the **`destroy()`** method will be called.

8. Write a program for card Layout control.
   Ans:-
   Here is an example program that demonstrates how to use a CardLayout control in Java:

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

```
import javax.swing.*;

public class CardLayoutExample extends JFrame implements ActionListener {

    private CardLayout cardLayout;
    private JPanel cardPanel;
    private JButton button1, button2, button3;

    public CardLayoutExample() {
        setTitle("Card Layout Example");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a new CardLayout and JPanel to hold the cards
        cardLayout = new CardLayout();
        cardPanel = new JPanel(cardLayout);

        // Create three buttons to switch between the cards
        button1 = new JButton("Card 1");
        button1.addActionListener(this);
        button2 = new JButton("Card 2");
        button2.addActionListener(this);
        button3 = new JButton("Card 3");
        button3.addActionListener(this);

        // Create three cards to display in the CardLayout
        JPanel card1 = new JPanel();
        card1.add(new JLabel("This is card 1."));
        JPanel card2 = new JPanel();
        card2.add(new JLabel("This is card 2."));
        JPanel card3 = new JPanel();
        card3.add(new JLabel("This is card 3."));

        // Add the cards to the cardPanel
        cardPanel.add(card1, "Card 1");
        cardPanel.add(card2, "Card 2");
        cardPanel.add(card3, "Card 3");
```

```
        // Add the buttons and cardPanel to the JFrame
        getContentPane().setLayout(new BorderLayout());
        getContentPane().add(cardPanel, BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel(new FlowLayout());
        buttonPanel.add(button1);
        buttonPanel.add(button2);
        buttonPanel.add(button3);
        getContentPane().add(buttonPanel, BorderLayout.SOUTH);
    }

    // Switch to the card associated with the clicked button
    public void actionPerformed(ActionEvent e) {
        String card = e.getActionCommand();
        cardLayout.show(cardPanel, card);
    }

    public static void main(String[] args) {
        CardLayoutExample example = new CardLayoutExample();
        example.setVisible(true);
    }
}
```

In this program, we create a JFrame with a CardLayout and a JPanel to hold the cards. We also create three buttons to switch between the cards. We then create three JPanels to display as the cards in the CardLayout.

We add the cards to the cardPanel, and add the buttons and cardPanel to the JFrame. We also add an ActionListener to each of the buttons to switch to the appropriate card when clicked.

When you run this program, you will see the first card displayed. Clicking any of the three buttons will switch to the corresponding card.

9. Write a program using Java swings to create a simple calculator.
   Ans:-

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

```
public class Calculator extends JFrame implements ActionListener {

    private JPanel panel;
    private JTextField textField;
    private JButton[] buttons;
    private String[] buttonLabels = {"7", "8", "9", "/", "4", "5", "6", "*", "1", "2",
"3", "-", "0", ".", "=", "+"};
    private double operand1, operand2, result;
    private char operator;

    public Calculator() {
        setTitle("Calculator");
        setSize(300, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        panel = new JPanel(new GridLayout(5, 4));
        textField = new JTextField(20);
        textField.setEditable(false);
        add(textField, BorderLayout.NORTH);

        buttons = new JButton[buttonLabels.length];
        for (int i = 0; i < buttonLabels.length; i++) {
            buttons[i] = new JButton(buttonLabels[i]);
            buttons[i].addActionListener(this);
            panel.add(buttons[i]);
        }

        add(panel, BorderLayout.CENTER);
    }

    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        switch (command) {
            case "0": case "1": case "2": case "3": case "4":
            case "5": case "6": case "7": case "8": case "9":
                textField.setText(textField.getText() + command);
```

```
            break;
        case "+": case "-": case "*": case "/":
            operator = command.charAt(0);
            operand1 = Double.parseDouble(textField.getText());
            textField.setText("");
            break;
        case "=":
            operand2 = Double.parseDouble(textField.getText());
            switch (operator) {
                case '+':
                    result = operand1 + operand2;
                    break;
                case '-':
                    result = operand1 - operand2;
                    break;
                case '*':
                    result = operand1 * operand2;
                    break;
                case '/':
                    result = operand1 / operand2;
                    break;
            }
            textField.setText(String.valueOf(result));
            break;
        case ".":
            if (textField.getText().indexOf('.') == -1) {
                textField.setText(textField.getText() + ".");
            }
            break;
        default:
            break;
    }
}

public static void main(String[] args) {
    Calculator calculator = new Calculator();
    calculator.setVisible(true);
```

```
    }

}
```

In this program, we create a JFrame with a GridLayout and a JTextField to display the input and output of the calculator. We also create an array of JButtons for the digits and operators, and add them to the panel. We add an ActionListener to each button to handle the user input.

The actionPerformed method parses the user input and performs the appropriate arithmetic operation. The result is displayed in the JTextField.

When you run this program, you will see a simple calculator with buttons for digits and arithmetic operations. You can use the calculator to perform basic arithmetic operations.

10. Write a program using Java swing to accept personal information of students and store them into a database.

Ans:- Here's an example Java Swing program that allows the user to input personal information for students and store them into a MySQL database:

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

public class StudentInfo extends JFrame implements ActionListener {

    private JLabel nameLabel, emailLabel, phoneLabel;
    private JTextField nameField, emailField, phoneField;
    private JButton saveButton;
    private Connection conn;

    public StudentInfo() {
        setTitle("Student Information");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        nameLabel = new JLabel("Name:");
        emailLabel = new JLabel("Email:");
        phoneLabel = new JLabel("Phone:");
```

```
        nameField = new JTextField(20);
        emailField = new JTextField(20);
        phoneField = new JTextField(20);

        saveButton = new JButton("Save");
        saveButton.addActionListener(this);

        JPanel panel = new JPanel(new GridLayout(4, 2));
        panel.add(nameLabel);
        panel.add(nameField);
        panel.add(emailLabel);
        panel.add(emailField);
        panel.add(phoneLabel);
        panel.add(phoneField);
        panel.add(saveButton);

        add(panel, BorderLayout.CENTER);

        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost/student",
"root", "");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }

    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String email = emailField.getText();
        String phone = phoneField.getText();

        try {
            PreparedStatement stmt = conn.prepareStatement("INSERT INTO students
(name, email, phone) VALUES (?, ?, ?)");
            stmt.setString(1, name);
            stmt.setString(2, email);
```

```
            stmt.setString(3, phone);
            int result = stmt.executeUpdate();
            if (result == 1) {
                JOptionPane.showMessageDialog(this, "Student information saved
successfully.");
                nameField.setText("");
                emailField.setText("");
                phoneField.setText("");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    public static void main(String[] args) {
        StudentInfo studentInfo = new StudentInfo();
        studentInfo.setVisible(true);
    }

}
```

In this program, we create a JFrame with a GridLayout and JTextFields to input the name, email, and phone number of the student. We also create a JButton to save the student information and add an ActionListener to the button to handle the database insertion.

The actionPerformed method retrieves the values of the JTextFields and inserts them into a MySQL database using a PreparedStatement. If the insertion is successful, a message dialog is displayed to inform the user.

To run this program, you will need to have MySQL installed and running on your system, and create a database named "student" with a table named "students" that has columns for "name", "email", and "phone". You will also need to add the MySQL connector JAR file to your project's classpath.