

# **RAJARAJESWARI COLLEGE OF ENGINEERING**

Affiliated to Visvesvaraya Technological University- Belagavi

#14 Ramohalli Cross, Kumbalgodu, Mysore Road, Bengaluru-560074



**Master of Computer Applications (MCA)**

**Programming in Java**

**Laboratory Manual**

**22MCA12**

**II- Semester**

(Prepared by Dr. Annu Sharma, Associate Prof. RRCE)

## **JAVA LAB Manual**

### **Introduction to Java:-**

- Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.
- Java was developed by Sun Microsystem in the year 1995. James Gosling is known as the father of Java. Before Java, its name was Oak.

### **History of Java:-**

In 1990, James Gosling was given a task of creating projects to control consumer electronics. Gosling, however, quickly found that C++ was not suitable for this project. They faced problems due to program bugs like memory leak, dangling pointer, multiple inheritance and platform dependent.

Gosling developed simplified computer language to avoid all the problems faced in C++.

Gosling kept the basic syntax and object oriented features of the C++ language to designing a new language.

He completed and named "OAK" in 1991 at Sun Microsystems.

Later Sun Microsystems was discovered that the name "OAK" was already claimed, they changed the name to "JAVA" in 1995.

### **Eclipse IDE(Integrated Development Environment):-**

Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. It is the second-most-popular IDE for Java development.

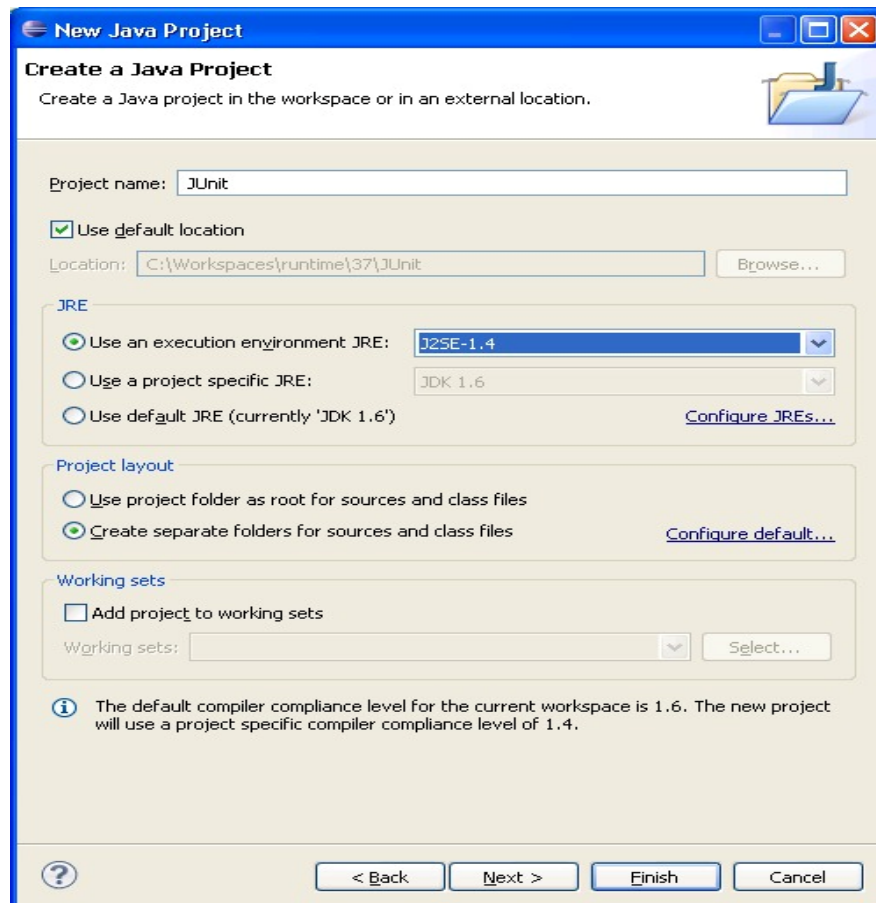
The Eclipse SDK is free and open-source software. It was one of the first IDEs to run under GNU Class path and it runs without problems under IcedTea.

### **Steps to create new Java project in Eclipse IDE:-**

Creating the project Inside Eclipse select the menu -> File -> New -> Project.... to open the New Project wizard

Select Java Project then click Next

- Enter the Project Name
- Select the Java Runtime Environment (JRE) or leave it at the default Select the Project Layout which determines whether there would be a separate folder for the source codes and class files. The recommended option is to create separate folders for sources and class files.



- Then click Finish.

### **Viewing the Newly Created Project:-**

The package explorer shows the newly created Java project. The icon that represents a Project is decorated with a J to show that it is a Java Project. The folder icon is decorated to show that it is a java source folder.



## Loop Costruct

**For Loop:-**for loop is a control flow statement for specifying iteration. Specifically, a for loop functions by running a section of code repeatedly until a certain condition has been satisfied.

**Syntax:-**

```
for(initialization; condition; increment/decrement){  
    //statement or code to be executed  
}
```

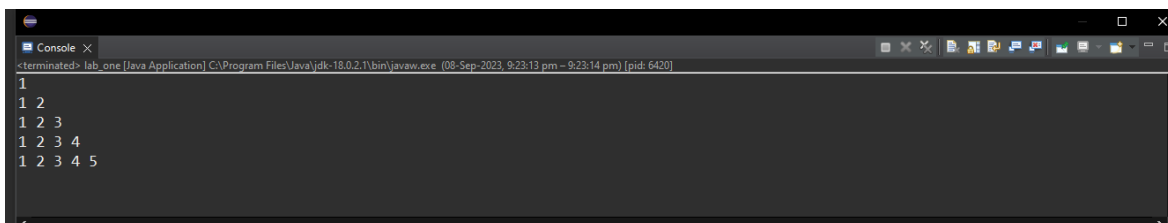
**Example:- Program1**

**1. Write a Java program to print the following triangle of numbers**

```
1  
12  
123  
1234  
12345
```

```
public class Prog1 {  
    public static void main(String[] args)  
    {  
        int rows = 5;  
        for (int i = 1; i <= rows; ++i) {  
            for (int j = 1; j <= i; ++j) {  
                System.out.print(j + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

**OUTPUT :-**



**While Loop:** The while loop loops through a block of code as long as a specified condition is true.**Syntax:-**

```
while (condition) { // code block to be executed }
```

**Example:-Program 2**

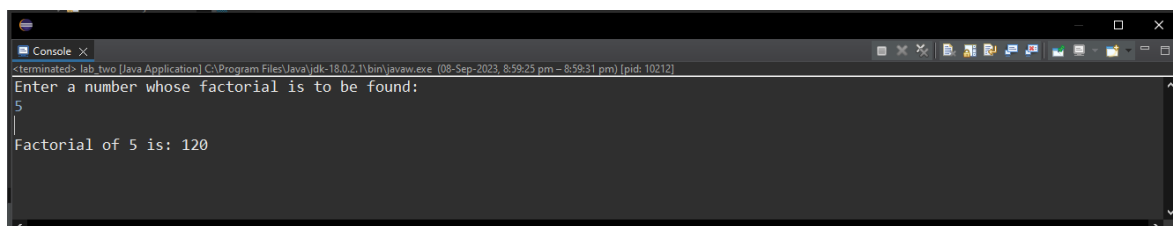
**2. Write a Java program to list the factorial of the numbers 1 to 10. To calculate the factorial value, use while loop.**

Example:-

```
import java.util.Scanner;

public class FactorialUsingWhileLoop {
    public static void main(String[] args) {
        //declaring and intializing variables
        int fact = 1;
        int i = 1;
        //creating object of Scanner class
        Scanner sc = new Scanner(System.in);
        //accepting a number from the user
        System.out.println("Enter a number whose factorial is to be found: ");
        int num = sc.nextInt();
        //counting the factorial using while loop
        while( i <= num ){
            fact = fact * i;
            i++;    //increment i by 1
        }
        //printing the result
        System.out.println("\nFactorial of " + num + " is: " + fact);
    }
}
```

**OUTPUT :-**

A screenshot of a Java console application window. The title bar shows the application name and path. The console output displays the prompt "Enter a number whose factorial is to be found:", the user input "5", and the result "Factorial of 5 is: 120".

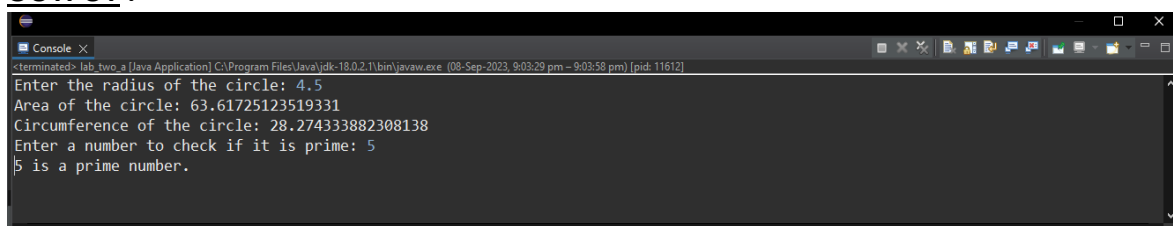
```
<terminated> lab_two [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (08-Sep-2023, 8:59:25 pm - 8:59:31 pm) [pid: 10212]
Enter a number whose factorial is to be found:
5
Factorial of 5 is: 120
```

### 3. Write a Java program

- i. To find the area and circumference of the circle by accepting the radius from the user.
- ii. To accept a number and find whether the number is Prime or not

```
import java.util.Scanner;
public class CircleAndPrime {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Calculate area and circumference of a circle
        System.out.print("Enter the radius of the circle: ");
        double radius = scanner.nextDouble();
        double area = Math.PI * radius * radius;
        double circumference = 2 * Math.PI * radius;
        System.out.println("Area of the circle: " + area);
        System.out.println("Circumference of the circle: " + circumference);
        // Check if a number is prime
        System.out.print("Enter a number to check if it is prime: ");
        int number = scanner.nextInt();
        boolean isPrime = true;
        if (number <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= Math.sqrt(number); i++) {
                if (number % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }
        if (isPrime) {
            System.out.println(number + " is a prime number.");
        } else {
            System.out.println(number + " is not a prime number.");
        }
        scanner.close();
    }
}
```

#### **OUTPUT :-**

A screenshot of a Java console application window. The title bar reads "Console X". The main text area shows the following output: "Enter the radius of the circle: 4.5", "Area of the circle: 63.61725123519331", "Circumference of the circle: 28.274333882308138", "Enter a number to check if it is prime: 5", and "5 is a prime number.". The window's status bar at the bottom indicates the path "C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe" and the PID "11612".

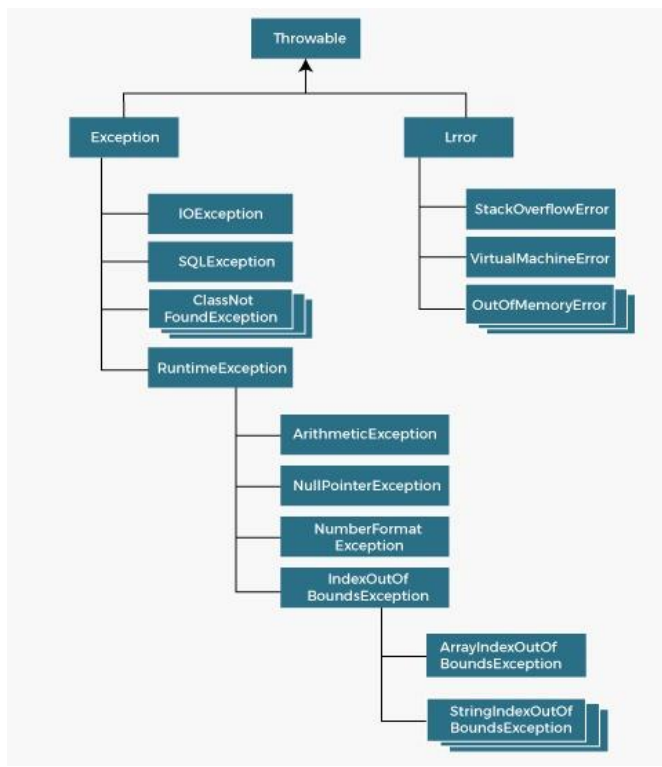
```
<terminated> lab_two.s [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (08-Sep-2023, 9:03:29 pm - 9:03:58 pm) [pid: 11612]
Enter the radius of the circle: 4.5
Area of the circle: 63.61725123519331
Circumference of the circle: 28.274333882308138
Enter a number to check if it is prime: 5
5 is a prime number.
```

## Exception:-

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

### Hierarchy of Java Exception classes:-

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



### Types of Java Exceptions:-

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

**1) Checked Exception:** The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

**2) Unchecked Exception:** The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

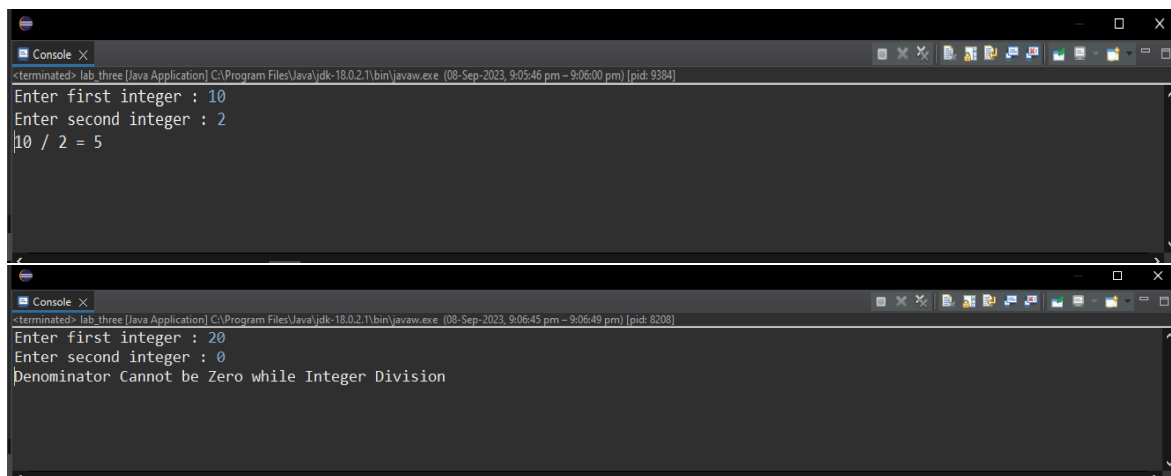
**3) Error:** Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

#### Example:- Program 4

#### **4. Write a Java program to demonstrate a division by zero exception**

```
import java.util.Scanner;
public class DivideByZeroException
{
    public static void main(String[] args)
    {
        int x,y;
        //Create scanner object to obtain input from keyboard
        Scanner input=new Scanner(System.in);
        try
        {
            System.out.print("Enter first integer : ");
            x=input.nextInt(); //Read first integer
            System.out.print("Enter second integer : ");
            y=input.nextInt(); //Read second integer
            System.out.println(x + " / " + y + " = " + (x/y));
        }
        catch(ArithmeticException e)
        {
            System.out.println("Denominator Cannot be Zero while Integer Division");
        }
    }
}
```

#### **OUTPUT :-**

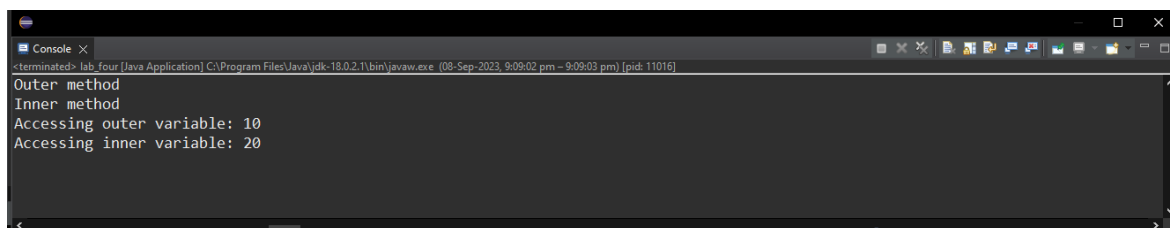




**5. Write a Java program to implement Inner class and demonstrate its Access protection.**

```
public class OuterClass {
    private int outerVariable = 10;
    public void outerMethod() {
        System.out.println("Outer method");
        InnerClass inner = new InnerClass();
        inner.innerMethod();
    }
    // Inner class
    public class InnerClass {
        private int innerVariable = 20;
        public void innerMethod() {
            System.out.println("Inner method");
            System.out.println("Accessing outer variable: " + outerVariable);
            System.out.println("Accessing inner variable: " + innerVariable);
        }
    }
    public static void main(String[] args) {
        OuterClass outer = new OuterClass();
        outer.outerMethod();
    }
}
```

**OUTPUT :-**

A screenshot of a Java console window. The window title is "Console" and it shows the output of a Java application. The output consists of four lines: "Outer method", "Inner method", "Accessing outer variable: 10", and "Accessing inner variable: 20". The console window is dark-themed and has standard Windows window controls at the top. The path shown in the title bar is "C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe".

```
<terminated> lab_four [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (08-Sep-2023, 9:09:02 pm - 9:09:03 pm) [pid: 11016]
Outer method
Inner method
Accessing outer variable: 10
Accessing inner variable: 20
```

**Constructor:** A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes

- Note that the constructor name must match the class name, and it cannot have a return type (like void).
- the constructor is called when the object is created.
- All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you. However, then you are not able to set initial values for object attributes.

### **Constructor overloading in Java:**

The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

### **Method Overloading in Java:**

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

### **Example:-Program 6**

## 6. Write a Java program to demonstrate Constructor Overloading and Method Overloading

### Example:

```
public class OverloadingDemo {
    // Constructor Overloading
    public OverloadingDemo() {
        System.out.println("Default constructor");
    }
    public OverloadingDemo(int number) {
        System.out.println("Parameterized constructor with an integer: " +
number);
    }
    public OverloadingDemo(String text) {
        System.out.println("Parameterized constructor with a string: " +
text);
    }

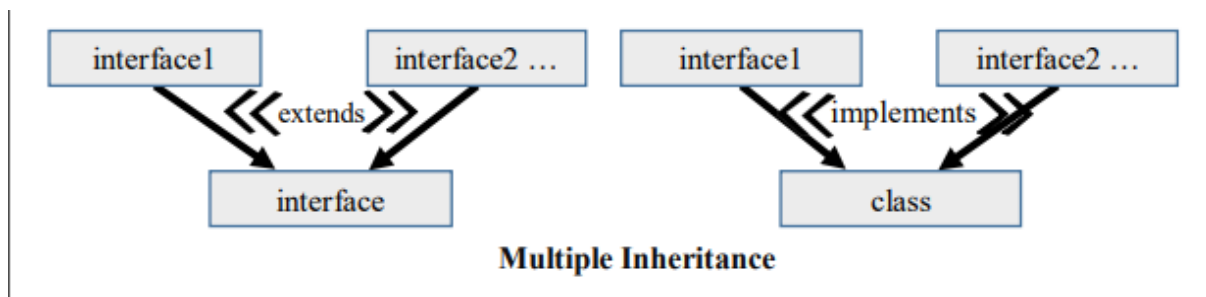
    // Method Overloading
    public void printNumber(int number) {
        System.out.println("Printing an integer: " + number);
    }
    public void printNumber(double number) {
        System.out.println("Printing a double: " + number);
    }
    public void printNumber(String text) {
        System.out.println("Printing a string: " + text);
    }
    public static void main(String[] args) {
        OverloadingDemo demo1 = new OverloadingDemo();
        OverloadingDemo demo2 = new OverloadingDemo(10);
        OverloadingDemo demo3 = new OverloadingDemo("Hello");
        demo1.printNumber(5);
        demo1.printNumber(3.14);
        demo1.printNumber("Java");
    }
}
```

## OUTPUT :-

```
Console X
<terminated> lab_five [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (08-Sep-2023, 9:14:20 pm - 9:14:21 pm) [pid: 3996]
Default constructor
Parameterized constructor with an integer: 10
Parameterized constructor with a string: Hello
Printing an integer: 5
Printing a double: 3.14
Printing a string: Java
```

## Interface:

- Interface is a type where methods are by default abstract and variables are by default static and final.
- We can create reference variables of interface type.
- It is impossible to create object of an interface.
- The class can inherit the properties of an Interface only by using Implements keyword.
- A class can implements any number of interfaces.
- A class can extends ONLY one Super class but it can implements any number of interfaces at the same time.
- A class can extends and implements at the same time.



Example: Program 7

**7. Write a JAVA program to demonstrate Inheritance. Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle.**

Example:

```
import java.io.*;
interface area
{
    float compute(float x, float y);
}

class rectangle
{
    public float compute(float x, float y)
    {
        return (x*y);
    }
}

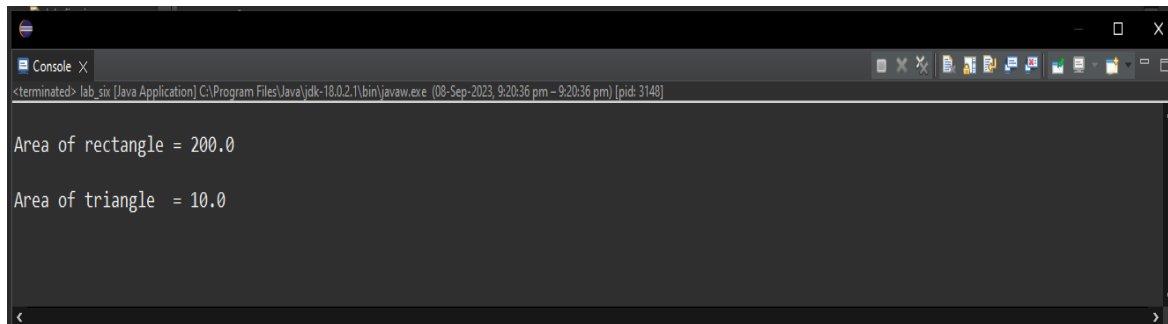
class triangle
{
    public float compute(float x, float y)
    {
        return (x*y/2);
    }
}

class result extends rectangle implements area
{
    public float compute(float x, float y)
    {
        return (x*y);
    }
}

class result1 extends triangle implements area
{
    public float compute(float x, float y)
    {
        return (x*y/2);
    }
}
```

```
class InterfaceMain
{
    public static void main(String args[])
    {
        result rect = new result();
        result1 tri = new result1();
        area a;
        a = rect;
        System.out.println("\nArea of rectangle = " + a.compute(10,20));
        a = tri;
        System.out.println("\nArea of triangle = " +a.compute(10,2));
    }
}
```

**OUTPUT :-**

A screenshot of a Java console window. The title bar shows 'Console X' and standard window controls. The status bar at the bottom indicates the application is terminated. The console output shows two lines: 'Area of rectangle = 200.0' and 'Area of triangle = 10.0'.

```
<terminated> lab_six [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (08-Sep-2023, 9:20:36 pm - 9:20:36 pm) [pid: 3148]

Area of rectangle = 200.0

Area of triangle = 10.0
```

**8. Write a Java applet program, which handles keyboard event.**

- Firstly create Keyboard.java file.
- Next create Keyboard.html file.

**Keyboard.java file.**

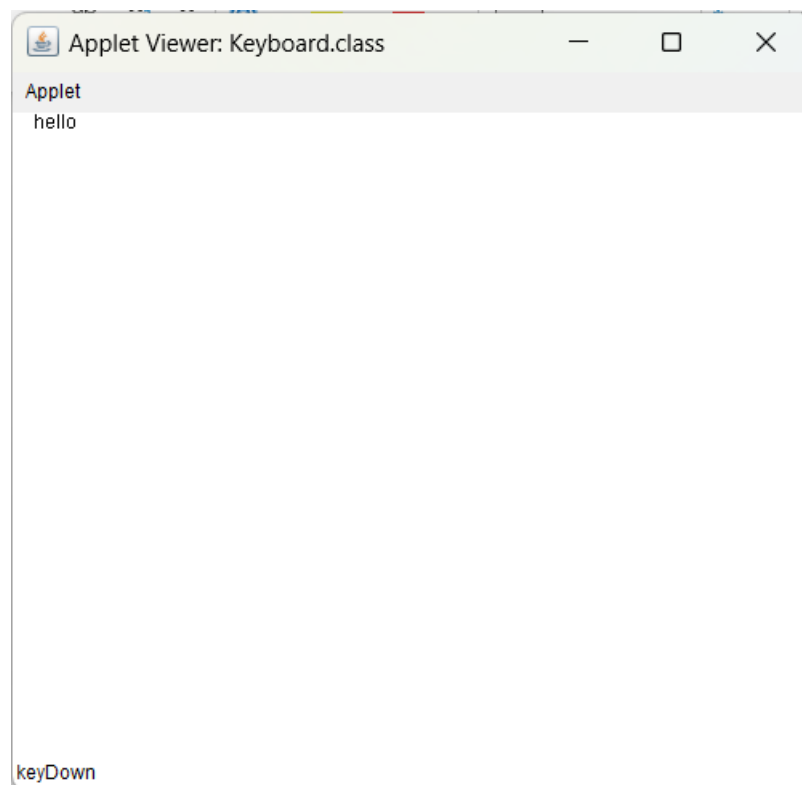
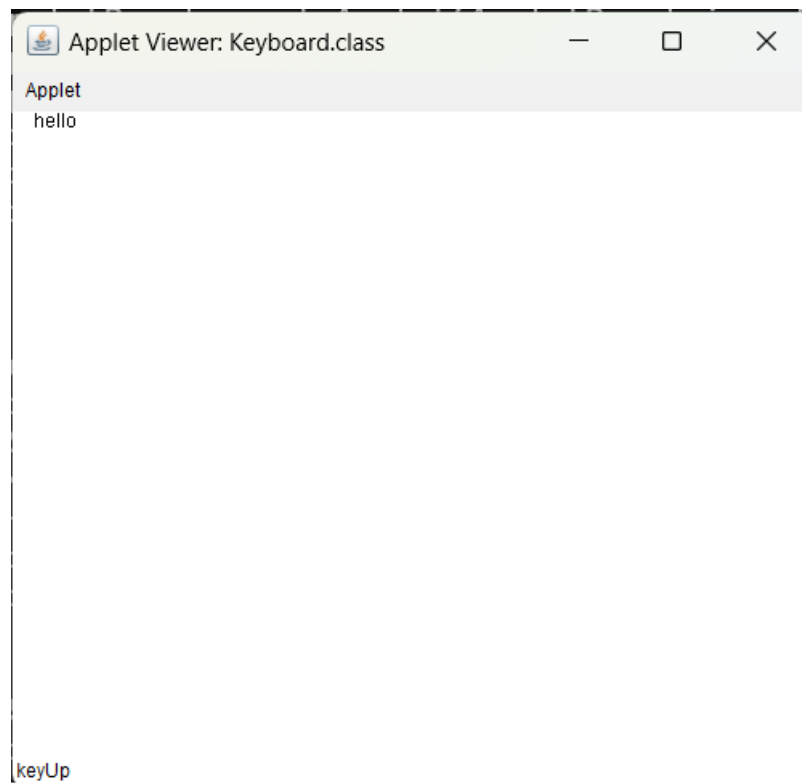
```
import java.awt.*;
import java.applet.*;
public class Keyboard extends Applet{
    String msg=" ";
    public boolean keyDown (Event obj,int key)
    {
        msg=msg+(char) key;
        repaint();
        showStatus("keyDown");
        return true;
    }
    public boolean keyUp (Event obj,int key)
    {
        showStatus("keyUp");
        return true;
    }
    public void paint (Graphics g){
        g.drawString(msg,10,10);
    }
}
```

**Keyboard.html file.**

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

**OUTPUT :-**

```
PS C:\Users\Prasad\OneDrive\Desktop> javac Keyboard.java
Note: Keyboard.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
PS C:\Users\Prasad\OneDrive\Desktop> appletviewer Keyboard.html
```





## Demonstration Experiments (For CIE) if any

9. Write a Java Program to create a window when we press

- M or m the window displays Good Morning
- A or a the window displays Good After Noon
- E ore the window displays Good Evening
- N or n the window displays Good Night

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

public class GreetingsWindow extends JFrame implements KeyListener {

    public GreetingsWindow() {
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setTitle("Greetings Window");
        setVisible(true);
        addKeyListener(this);
    }

    @Override
    public void keyPressed(KeyEvent e) {
        char keyChar = Character.toLowerCase(e.getKeyChar());
        String greeting = "";
        switch (keyChar) {
            case 'm':
                greeting = "Good Morning";
                break;
            case 'a':
                greeting = "Good Afternoon";
                break;
            case 'e':
                greeting = "Good Evening";
```

```

        break;
    case 'n':
        greeting = "Good Night";
        break;
    }
    if (!greeting.isEmpty()) {
        JOptionPane.showMessageDialog(this, greeting);
    }
}

@Override
public void keyTyped(KeyEvent e) {
    // Not used
}

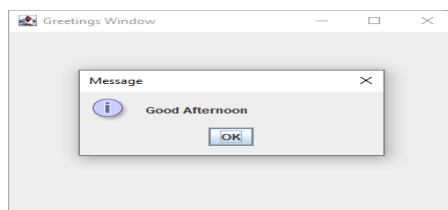
@Override
public void keyReleased(KeyEvent e) {
    // Not used
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(GreetingsWindow::new);
}
}

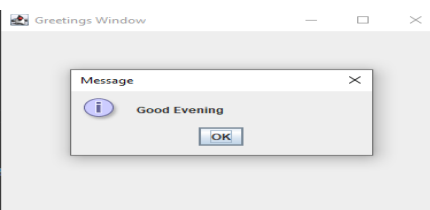
```

### **OUTPUT :-**

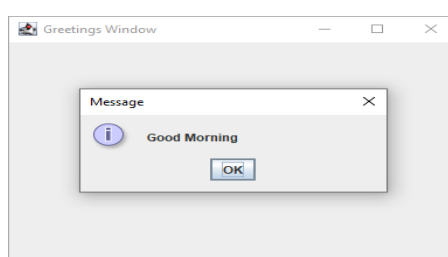
Press – a



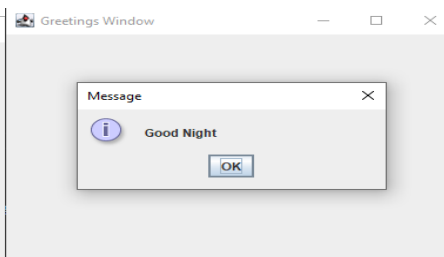
Press - e



Press – m



Press - n



**10. Write a Java program to implement a Queue using user defined Exception Handling (also make use of throw, throws).**

- a. Complete the following:
- b. Create a package named shape.
- c. Create some classes in the package representing some common shapes like Square, Triangle, and Circle.
- d. Import and compile these classes in other program

**Program:**

```
class QueueEmptyException extends Exception {  
    public QueueEmptyException(String message) {  
        super(message);  
    }  
}
```

```
class QueueFullException extends Exception {  
    public QueueFullException(String message) {  
        super(message);  
    }  
}
```

```
class Queue {  
    private int[] elements;  
    private int front;  
    private int rear;  
    private int size;  
  
    public Queue(int capacity) {  
        elements = new int[capacity];  
        front = 0;  
        rear = -1;  
        size = 0;  
    }  
  
    public boolean isEmpty() {  
        return size == 0;  
    }  
}
```

```

public boolean isFull() {
    return size == elements.length;
}

public void enqueue(int item) throws QueueFullException {
    if (isFull()) {
        throw new QueueFullException("Queue is full");
    }
    rear = (rear + 1) % elements.length;
    elements[rear] = item;
    size++;
}

public int dequeue() throws QueueEmptyException {
    if (isEmpty()) {
        throw new QueueEmptyException("Queue is empty");
    }
    int item = elements[front];
    front = (front + 1) % elements.length;
    size--;
    return item;
}
}

public class Main {
    public static void main(String[] args) {
        Queue queue = new Queue(5);
        try {
            queue.enqueue(10);
            queue.enqueue(20);
            queue.enqueue(30);
            System.out.println("Dequeued item: " + queue.dequeue());
            System.out.println("Dequeued item: " + queue.dequeue());
            System.out.println("Dequeued item: " + queue.dequeue());
            System.out.println("Dequeued item: " + queue.dequeue());
        } catch (QueueFullException | QueueEmptyException e) {

```

```
        System.out.println("Exception occurred: " + e.getMessage());
    }
}
}
```

**Save as Main.java**

**Output: -**

```
PS C:\Users\Prasad\OneDrive\Desktop> javac Main.java
PS C:\Users\Prasad\OneDrive\Desktop> java Main
Dequeued item: 10
Dequeued item: 20
Dequeued item: 30
Exception occurred: Queue is empty
PS C:\Users\Prasad\OneDrive\Desktop> |
```

**B. To create a package named "shape," you need to follow these steps:**

- **Create a new folder on your file system with the name "shape."**
- **Inside the "shape" folder, create a new Java file named "Square.java" with the following content:**

**Save as :- Square.java**

```
package shape;
public class Square {
    private double side;
    public Square(double side) {
        this.side = side;
    }
    public double getArea() {
        return side * side;
    }
}
```

**Save as :- Triangle.java**

```
package shape;
public class Triangle {
    private double base;
    private double height;
```

```
public Triangle(double base, double height) {  
    this.base = base;  
    this.height = height;  
}  
public double getArea() {  
    return 0.5 * base * height;  
}  
}
```

Save as :- Circle.java

```
package shape;  
  
public class Circle {  
    private double radius;  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
    public double getArea() {  
        return Math.PI * radius * radius;  
    }  
}
```

**D. To import and compile these classes in another program, follow these steps:**

- **Create a new Java file in the same directory as the "shape" folder. Name it "ShapeMain.java" and use the following code:**

```
import shape.Square;  
import shape.Triangle;  
import shape.Circle;  
  
public class ShapeMain {  
    public static void main(String[] args) {  
        Square square = new Square(5.0);  
        Triangle triangle = new Triangle(4.0, 3.0);  
        Circle circle = new Circle(2.5);  
    }  
}
```

```
        System.out.println("Area of square: " + square.getArea());
        System.out.println("Area of triangle: " + triangle.getArea());
        System.out.println("Area of circle: " + circle.getArea());
    }
}
```

**Output: -**

```
C:\Users\Prasad\OneDrive\Desktop\shape>javac Square.java
C:\Users\Prasad\OneDrive\Desktop\shape>javac Circle.java
C:\Users\Prasad\OneDrive\Desktop\shape>javac Triangle.java

PS C:\Users\Prasad\OneDrive\Desktop> javac ShapeMain.java
PS C:\Users\Prasad\OneDrive\Desktop> java ShapeMain.java
Area of square: 25.0
Area of triangle: 6.0
Area of circle: 19.634954084936208
PS C:\Users\Prasad\OneDrive\Desktop> |
```