Experiment Number: 8 **Date: 24.02.2026**

Implementation of Packages in Java

Aim:

Write a Java program to implement built-in, user-defined packages and accessing all classes in a package.

PRE LAB EXERCISE

QUESTIONS

1. **What is java.util package?** java.util is a core Java package that provides utility classes and the **Java Collection Framework (JCF)** for storing and managing data.

Collection Framework includes:

- **Interfaces:** List, Set, Queue, Map
- **Classes:** ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap, PriorityQueue, Vector, Stack

It is mainly used to store and manipulate groups of objects.

2. **Types of packages in Java:**

1. **Built-in packages** – Predefined (e.g., java.lang, java.util, java.io).
2. **User-defined packages** – Created by the programmer using the package keyword.

IN LAB EXERCISE

Objective

To understand and implement the concepts of built-in, user-defined packages and accessing all classes in a package in Java.

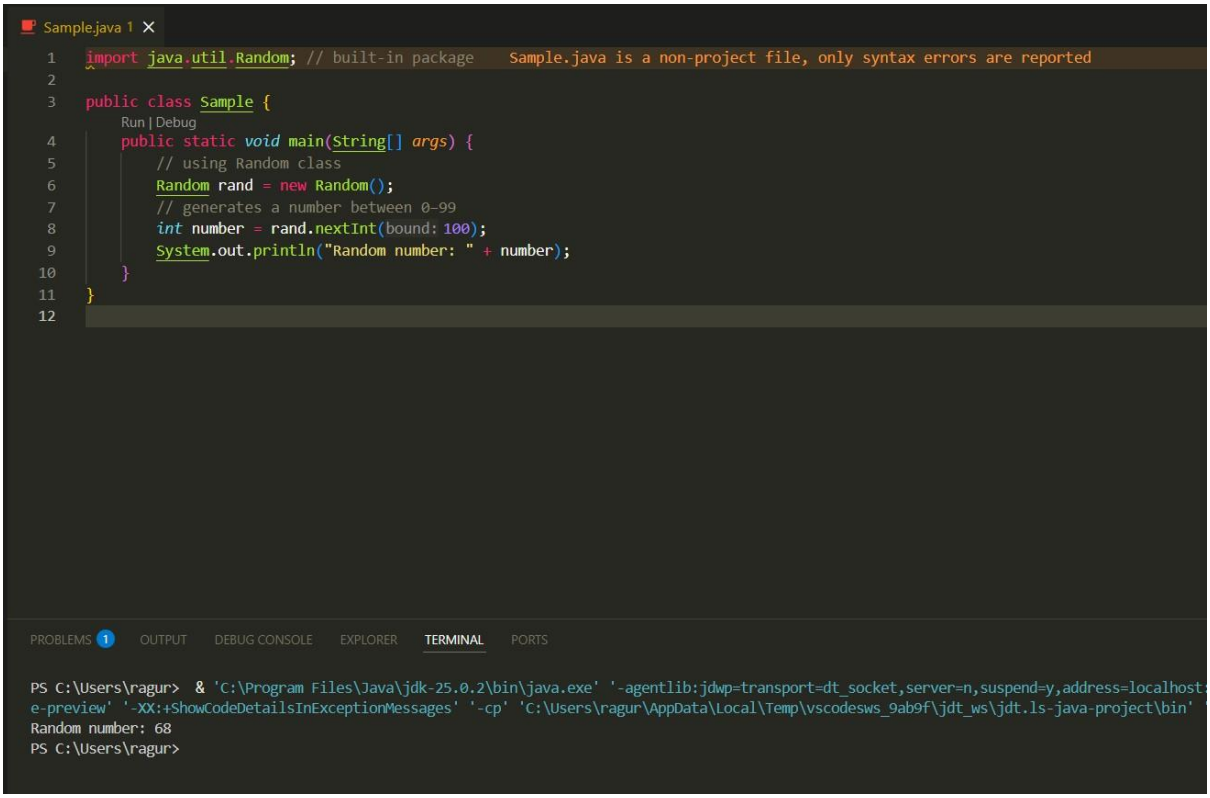
Built-in Packages comprise a large number of classes that are part of the Java API. Some of the commonly used built-in packages are:

- java.lang: Contains language support classes(e.g, classes that define primitive data types, math operations). This package is automatically imported.
- java.io: Contains classes for supporting input/output operations.
- java.util: Contains utility classes that implement data structures such as Linked Lists and Dictionaries, as well as support for date and time operations.
- java.applet: Contains classes for creating Applets.
- java.awt: Contains classes for implementing the components for graphical user interfaces (like buttons, menus, etc). **Source Code**

```
import java.util.Random;   // built-in package public

class Sample{       public static void main(String[]

args) {         // using Random class

      Random rand = new Random();          //

generates a number between 0–99        int

number = rand.nextInt(100);

      System.out.println("Random number: " + number);

   }

}
```

Output

Random number: 49



User-defined Packages are the packages that are defined by the user. **Source code** package com.myapp; public class Helper { public static void show() {

System.out.println("Hello from Helper!");

}

}

==To use this in another class== import com.myapp.Helper; public class Test {

public static void main(String[] args) {

Helper.show();

}

}

Output:

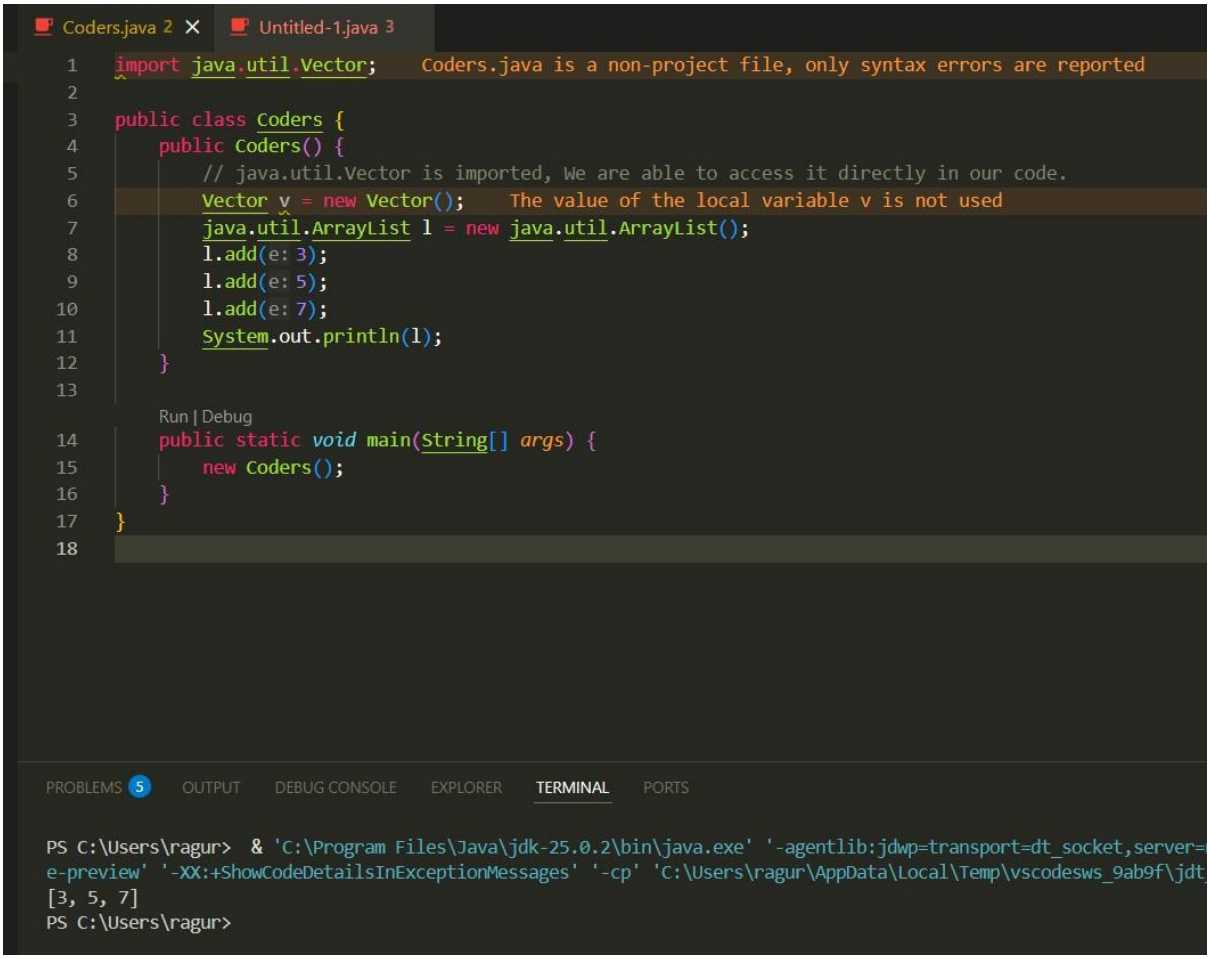
Hello from Helper!

//Importing all classes from a package. Source code

```
import java.util.Vector; public
class Coders {      public Coders()
{
          // java.util.Vector is imported, We are able to access it directly in our code.
Vector v = new Vector();        java.util.ArrayList l = new java.util.ArrayList();        l.add(3);
      l.add(5);
      l.add(7);
      System.out.println(l);
   }
   public static void main(String[] args) {       new
Coders();
   }
}
```

Output

[3,5,7]



POST LAB EXERCISE

1. Same class name from different packages?

It causes a compile-time ambiguity error.

Use fully qualified names (e.g., package1.Sample) to avoid confusion.

2. Purpose of packages in Java:

- Organize classes
- Avoid naming conflicts
- Improve readability and maintenance
- Provide access control • Support code reusability

3. Package for GUI window?

Correct answer: **C. java.awt**

Used to create GUI components like windows and buttons.

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		

Experiment Number: 9 **Date: 24.02.2026**

Implementation of a Java Program to import packages using different methods

Aim:

Write a Java program to import packages using different methods for different use cases.

PRE LAB EXERCISE

QUESTIONS

1. **Importing in Java:**

- **Single class:** import packageName.ClassName; Example: import java.util.Scanner;
- **All classes:** import packageName.*; Example: import java.util.*;

2. **Default imported package:** java.lang

It includes classes like String, System, Math, Integer, and Object. No need to import it explicitly.

IN LAB EXERCISE

Objective

To understand and implement the Java packages using different methods and import them.

Problem

Define a package named ‘useFul’ with a class names ‘UseMe’ having following methods:

1) area()- To calculate the area of given shape.

2) salary()- To calculate the salary given basic Salary,da,hRA.

3) percentage()-To calculate the percentage given total marks and marks obtained.

4) Develop a program named ‘Package Use’ to import the above package ‘useFul’ and use the method area().

5) Develop a program named ‘manager’

Source Code

```
//Package Creation:
package useFull; import
java.util.*;  public class
UseMe
{
Scanner obj=new Scanner(System.in);  public static
void area()
      {
  class method{     void
aos(int a)
  {
     System.out.print("\nArea of square with length "+a+" is "+(a*a));
   }
   void aor(int a,int b)
  {
     System.out.print("\nArea of reactangle with dimensions "+a+" & "+b+" is "+(a*b));
   }
   void aoc(int r)
  {
    double a=3.14*r*r;
   }
           System.out.print("\nArea of circle with radius "+r+" is "+a);
        }
void aot(int a,int b)
{
   float ar=(a*b)/2;
   System.out.print("\nArea of triangle with dimensions "+a+" &"+b+" is "+ar);
 }}
```

```
Scanner obj=new Scanner(System.in); method
m=new method();
System.out.print("\n1.Square\n2.Rectangle\n3.Circle\n4.Triangle\nSelect the shape\n"); int
ch=obj.nextInt();
UseMe u=new UseMe();  switch(ch)
{
case 1:System.out.print("\nEnter the length of side of square : "); int
s=obj.nextInt();m.aos(s); break; case 2:System.out.print("\nEnter the
dimensions of rectangle : "); int l=obj.nextInt(); int b=obj.nextInt();
m.aor(l,b); break; case 3:System.out.print("\nEnter the radius of circle : ");
int r=obj.nextInt(); m.aoc(r); break; case 4:System.out.print("\nEnter the
dimensions of triangle : "); int ba=obj.nextInt(); int w=obj.nextInt();
m.aot(ba,w); break; } }

public void salary()
  {
int ba,da,hra;
System.out.print("\nEnter the basic salary : "); ba=obj.nextInt();
System.out.print("\nEnter the dearness allowance :"); da=obj.nextInt();
System.out.print("\nEnter the house rent allowance : "); hra=obj.nextInt();
System.out.print("\nThe total Gross salary of employee is : "+(ba+da+hra));
   }
public void percentage()
   {
int n,sum=0;  float p;
System.out.print("\nEnter the total number of subjects : ");
n=obj.nextInt(); int m[]=new int[n];
System.out.print("\nEnter the marks of "+n+" subjects : ");  for(int
i=0;i<n;i++)
```

```
{
    m[n]=obj.nextInt();
}
for(int i=0;i<n;i++)
{
     sum=sum+m[i];
}
p=sum/n;
{
System.out.print("\nPercentahe of student : "+p);
}
     }
}
//Package Implementation-1:
import useFull.UseMe; class
packageUse
{
public static void main(String args[])
{
UseMe o=new UseMe();o.area();
}
}
```

Output javac

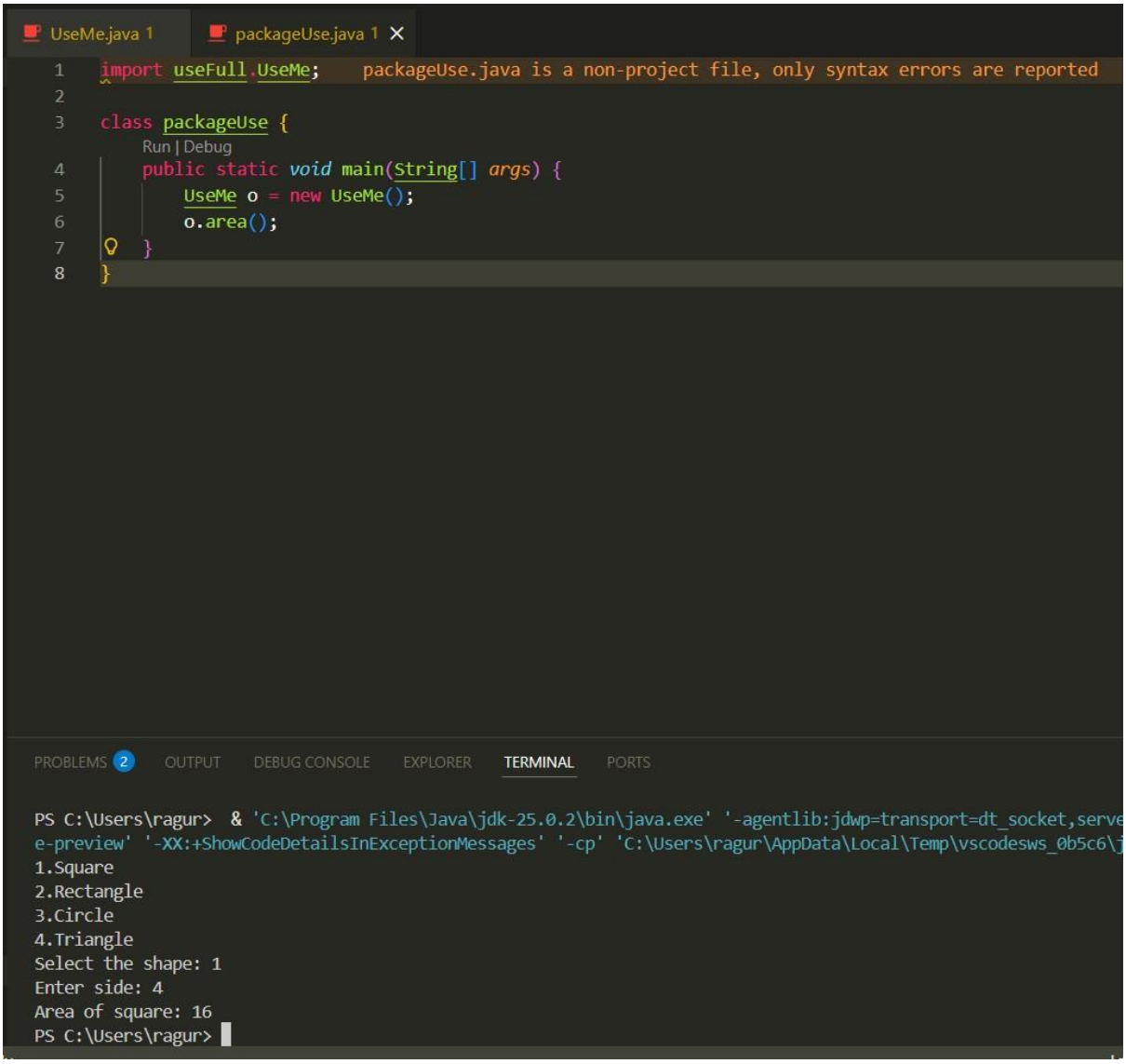
packageUse.java java

packageUse

1. Square
2. Rectangle

3. Circle
4. Triangle

Select the shape



2

Enter the dimensions of the rectangle: 10 15 Area of the

rectangle with dimensions 10&15 is 150 //Package

Implementation-2:

```
import useFull.UseMe; class
manager
{
   public static void main(String args[])
  {
    UseMe obj=new UseMe();obj.salary();
   }
}
```

Output javac

manager.java java

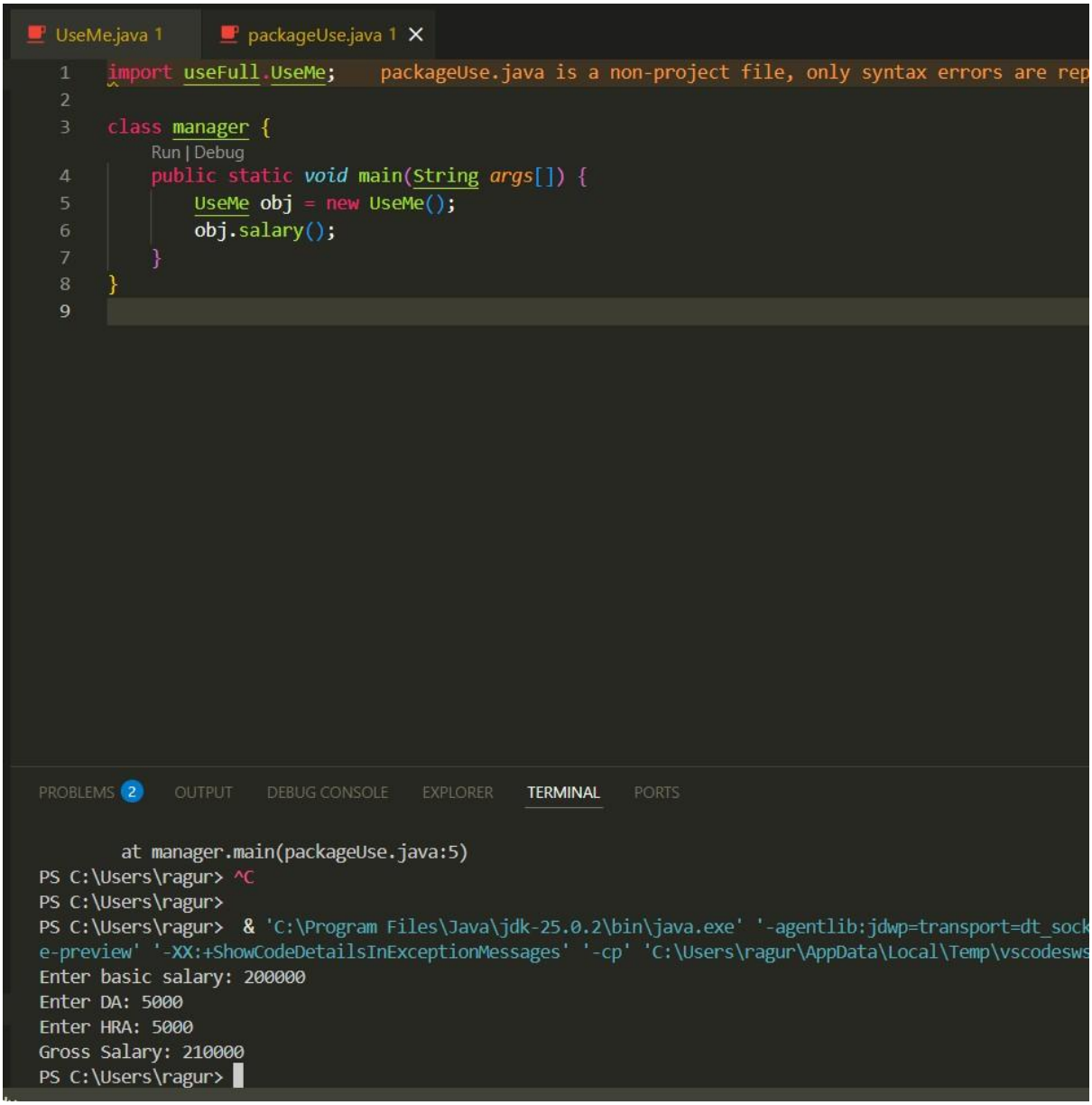
manager

Enter the basic salary: 100000

Enter the dearness allowance: 5000

Enter the house rent allowance: 2000

The total Gross salary of employee is: 107000



POST LAB EXERCISE

1. Difference between java.lang and java.util:

- **java.lang** → Automatically imported; provides core classes like String, System, Math, Integer.
- **java.util** → Must be imported; provides utility classes like Scanner, Random, ArrayList, HashMap, Date.

---

2. Subpackages of java.util:

- java.util.concurrent
- java.util.function
- java.util.logging
- java.util.stream
- java.util.regex
- java.util.zip
- java.util.jar

They support concurrency, streams, logging, regex, compression, etc.

ASSESSMENT

Description	**Max Marks**	**Marks Awarded**
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		

Experiment Number : 10 **Date: 02/03/2026**

Implementation of Access Modifiers in Java

Aim:

Write a Java program to implement different access modifiers.

PRE LAB EXERCISE QUESTIONS

1. What are the 4 access modifiers available in Java?

   - Default
   - Private
   - Protected
   - Public

2. Which access modifiers can be used in Java to control access to class members?

   - Private
   - Protected
   - Public
   - Default

IN LAB EXERCISE

Objective

To demonstrate different access modifiers such as Default, Private, Protected and Public using Java programs.

	Default	Private	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same Package Subclass	Yes	No	Yes	Yes
Same Package Non-Subclass	Yes	No	Yes	Yes
Different Package Subclass	No	No	Yes	Yes
Different Package Non-Subclass	No	No	No	Yes

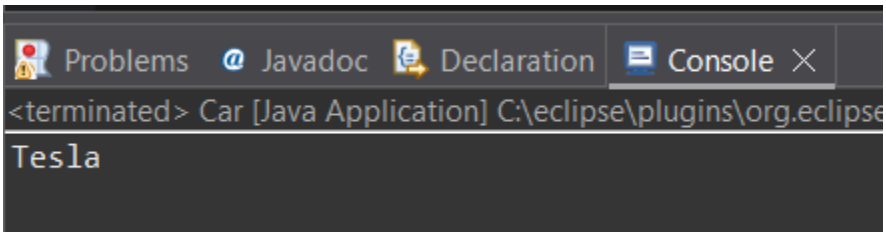
Fig: Comparison table of Access Modifiers in Java

Source Code

```
// Default Access modifier
class Car {
    String model; // default access
}
public class Main {
        public static void main(String[] args){
         Car c = new Car();
         c.model = "Tesla"; // accessible within the same package
         System.out.println(c.model);
    }
}
```

Output:

Tesla

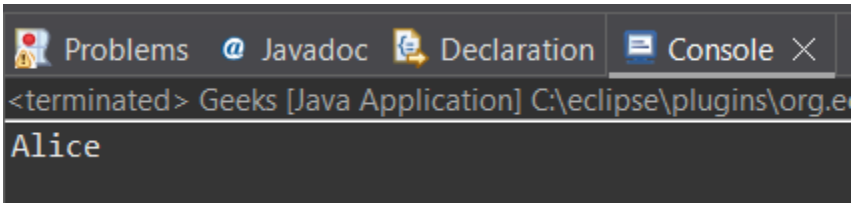


//Private Access Modifier

```
class Person {
    // private variable
    private String name;
    public void setName(String name)  {
     this.name = name; // accessible within class
    }
    public String getName() {
            return name;
   }
}
public class Geeks {
    public static void main(String[] args)
    {
       Person p = new Person();
        p.setName("Alice");

        // System.out.println(p.name);
        // Error: 'name'
        // has private access
        System.out.println(p.getName());
    }
}
```

Output

Alice

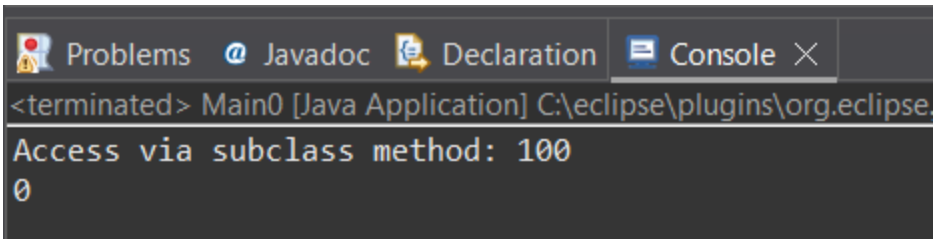


//Protected Access Modifier

```
class Vehicle {
   protected int speed; // protected member
}
class Bike extends Vehicle {
   void setSpeed(int s)
   {
      speed = s; // accessible in subclass
   }
   int getSpeed()
   {
      return speed; // accessible in subclass
   }
}
public class Main {
   public static void main(String[] args){
      Bike b = new Bike();
      b.setSpeed(100);
      System.out.println("Access via subclass method: "+ b.getSpeed());
      Vehicle v = new Vehicle();
      System.out.println(v.speed);
   }
}
```

Output

```
Access via subclass method: 100
0
```

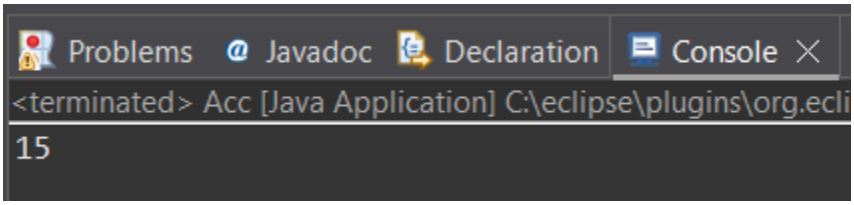


//Public Access Modifier

```
class MathUtils {
        public static int add(int a, int b) {
        return a + b;
    }
}
public class Main {
        public static void main(String[] args) {
        System.out.println(MathUtils.add(5, 10)); // accessible anywhere
    }
}
```

Output

15



POST LAB EXERCISE

1. Write a Java program to implement Default access modifier which has a default class within the same package and a default class from a different package.

```
class DefaultClass {
    void display() {
        System.out.println("Default access within same package");
```

```
    }
}

public class TestDefault {
    public static void main(String[] args) {
        DefaultClass obj = new DefaultClass();
        obj.display();
    }
}
```

2. Which access modifier provides the highest level of access?

   Public access modifier provides the highest level of access.

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		

Experiment Number :11 **Date: 02/03/2026**

Implementation of multiple inheritance in Java using Interface

Aim:

Write a Java program to implement multiple inheritance using Interface.

PRE LAB EXERCISE

QUESTIONS

1. Why Java does not support multiple inheritance using classes like that of C?

   Because it can cause **ambiguity problem (Diamond problem)** and complexity.

2. What is the primary purpose of an interface in Java?

A. To store data using instance variables
B. To define a contract of methods a class must implement
C. To allow creation of objects
D. To improve runtime performance

To define a contract of methods a class must implement

IN LAB EXERCISE

Objective

To demonstrate how an interface in Java defines constants and abstract methods, which are implemented by a class.

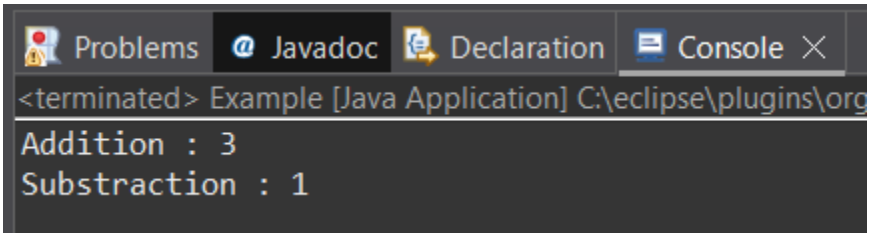
Source Code

```
import java.io.*;
  // Add interface
interface Add{
    int add(int a,int b);
  }
// Sub interface
interface Sub{
        int sub(int a,int b);
  }
// Calculator class implementing Add and Sub
class Cal implements Add , Sub
  {
// Method to add two numbers
        public int add(int a,int b){
        return a+b;
    }
        // Method to sub two numbers
```

```
        public int sub(int a,int b){
        return a-b;
    }
}
class Example{
    // Main Method
    public static void main (String[] args){

        // instance of Cal class
        Cal x = new Cal();
        System.out.println("Addition : " + x.add(2,1));
        System.out.println("Substraction : " + x.sub(2,1));
    }
}
```

Outputs

Addition : 3
Subtraction : 1



POST LAB EXERCISE

3. Can a functional interface extend another interface?

   Yes, but only if it still has **one abstract method**.

4. Which feature was introduced in interfaces starting from Java 8?

   A. Constructors

   B. Private methods

C. Default and static methods

D. Final classes

Default and static methods feature was introduced in interfaces starting from Java 8.

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		

Experiment Number : 12 **Date: 02/03/2026**

Implementation of exception handling in Java

Aim:

Write a java program to implement exception handling.

PRE LAB EXERCISE QUESTIONS

1. What is exception handling in Java?
   Exception handling is a mechanism to handle runtime errors so that the program does not terminate abnormally.

2. What is the purpose of using try-catch blocks in exception handling?
   To catch and handle exceptions that occur inside the try block and prevent program crash.

IN LAB EXERCISE

Objective

To understand and implement exception handling through try-catch and finally blocks.

Source Code

```
import java.util.InputMismatchException;

import java.util.Scanner;
```

```
public class ExceptionHandlingExample {
 public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
         try {
        System.out.print("Enter an integer: ");
        int num = scanner.nextInt();
         int result = 10 / num;
        System.out.println("Result: " + result);
        } catch (InputMismatchException e) {
        System.out.println("Invalid input! Please enter an integer.");
        } catch (ArithmeticException e) {
        System.out.println("Cannot divide by zero!");
        } finally {
        scanner.close();
        System.out.println("Program execution completed.");
        } }
}
```

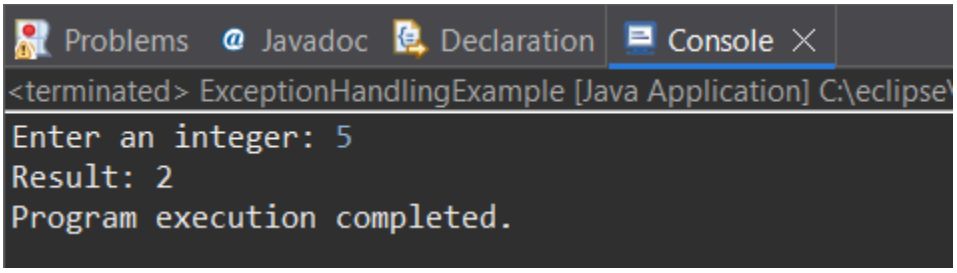
Outputs

Output 1 (Valid Input - Non-zero Integer):

Enter an integer: 5

Result: 2

Program execution completed.

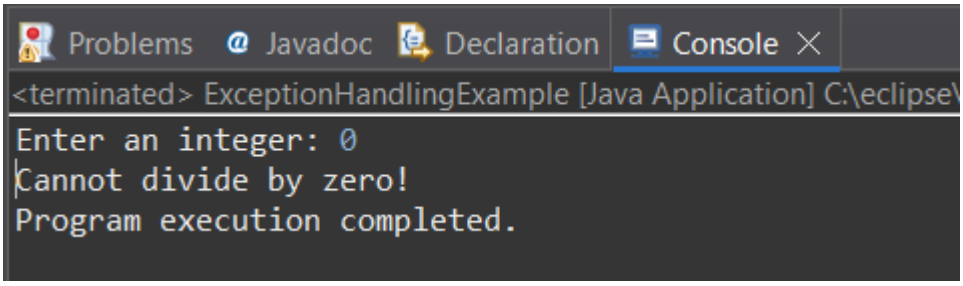


Output 2 (Valid Input - Zero):

Enter an integer: 0

Cannot divide by zero!

Program execution completed.

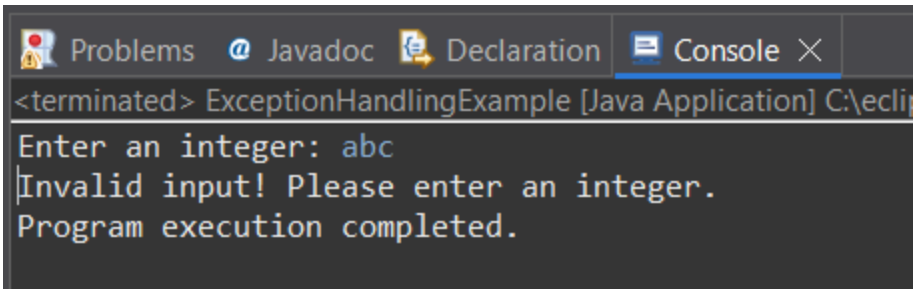


Output 3 (Invalid Input - Non-integer):

Enter an integer: abc

Invalid input! Please enter an integer.

Program execution completed.



POST LAB EXERCISE

1. What is the purpose of the Scanner object in the Java program?

   It is used to take input from the user.

2. What exceptions are expected to be thrown in the code within the try block? Why?

- InputMismatchException – if user enters non-integer value
- ArithmeticException – if user enters zero (division by zero)

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		

Experiment Number : 13 **Date: 02/03/2026**

Implementation of User-defined Exception in Java

Aim:

Write a java program to implement User-defined Exception.

PRE LAB EXERCISE QUESTIONS

1. What is Java custom exception?

   A custom exception is a user-defined exception created by extending Exception or RuntimeException classes.

2. What are the two types of custom exceptions in Java?

   - Checked Exception
   - Unchecked Exception

IN LAB EXERCISE

Objective

To understand and implement Checked and Unchecked custom exceptions.

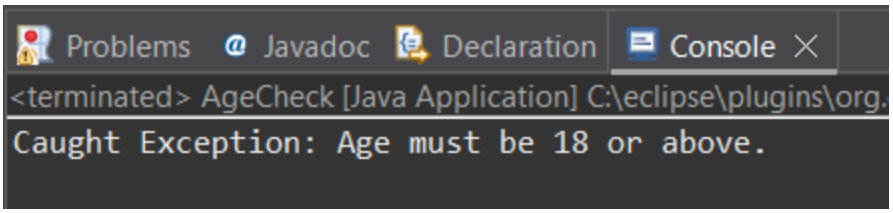
Source Code

Ex 1: // Custom Checked Exception

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String m) {
        super(m);
    }
}
// Using the Custom Exception
public class AgeCheck{
    public static void validate(int age)
      throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age must be 18 or above.");
        }
        System.out.println("Valid age: " + age);
    }
public static void main(String[] args) {
  try {
            validate(12);
        } catch (InvalidAgeException e) {
            System.out.println("Caught Exception: " + e.getMessage());
        }
    }
}
```

Output 1

Caught Exception: Age must be 18 or above.

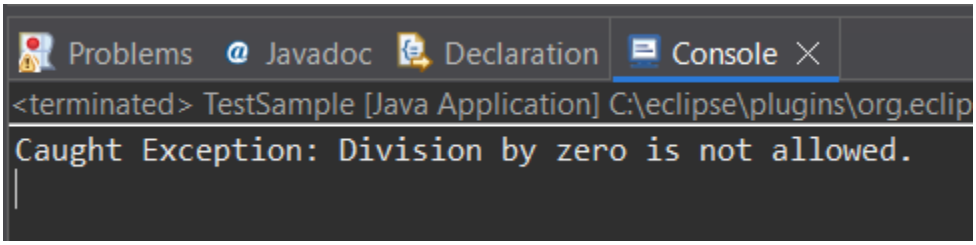


Ex 2: // Custom Unchecked Exception

```
class DivideByZeroException extends RuntimeException {
    public DivideByZeroException(String m) {
        super(m);
    }
}
// Using the Custom Exception
public class TestSample {
    public static void divide(int a, int b) {
        if (b == 0) {
            throw new DivideByZeroException("Division by zero is not allowed.");
        }
        System.out.println("Result: " + (a / b));
    }
public static void main(String[] args) {
  try {
            divide(10, 0);
        } catch (DivideByZeroException e) {
            System.out.println("Caught Exception: " + e.getMessage());
        }
    }
}
```

Output 2

Caught Exception: Division by zero is not allowed.



POST LAB EXERCISE

1. What is required to create a custom exception in Java?
   A. Extend the Object class
   B. Extend the Throwable class
   C. Extend Exception or RuntimeException
   D. Implement the Serializable interface

   Extend Exception or RuntimeException is required to create a custom exception in Java.

2. List some use cases for the checked and unchecked exceptions.
   Checked Exceptions
   - File handling errors
   - Database connection errors
   - Validation errors

   Unchecked Exceptions
   - Arithmetic errors
   - Null pointer errors
   - Logical programming errors

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		