Experiment Number : 8 **Date:12-02-2026**

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 and what collection framework does it contain?

   Answer: java.util package:
   A built-in Java package that provides utility classes and the Collection Framework to store and manage groups of objects (like List, Set, Map, ArrayList, HashMap, etc.).

2. What are the two types of packages in Java?

   Answer: Two types of packages in Java:

1. Built-in packages
2. User-defined packages

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.

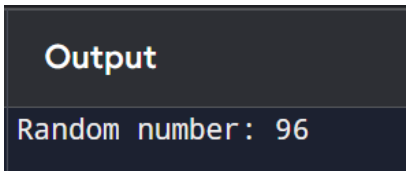
Built-in Packages comprise a large number of classes that are part of the Java API. Some of the commonly used built-in packages are:

- 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



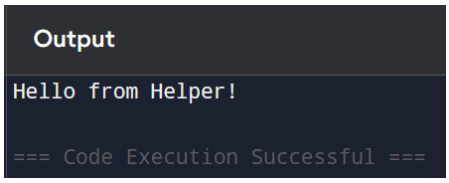
User-defined Packages are the packages that are defined by the user.

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:



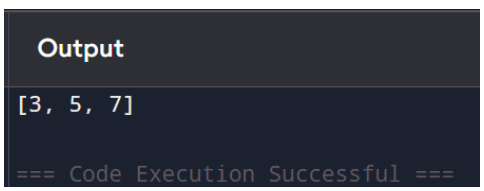
//Importing all classes from a package.

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. What will happen if two classes in different packages have the same name and are imported in a Java file?

   Answer: It causes **ambiguity error**. You must use the **fully qualified class name** for one of them.

2. What is the purpose of using packages in Java?

   Answer: To organize classes, avoid name conflicts, and support code reuse & security.

3. Which built-in Java package would you use if you want to create a GUI window and display a message?

   A. java.util

   B. java.sql

   C. java.awt

   D. java.net

   Answer: java.awt

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:12-02-2026**

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. How to import a single class and multiple classes from a package in Java?

   Answer:

   Single Class: import packageName.ClassName;

   Multiple Class: import packageName.*;

2. Which package is always imported by default in every Java class?

   Answer: java.lang

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

```
Output
1. Square
2. Rectangle
3. Circle
4. Triangle
Select the shape: 2
Enter length: 10
Enter breadth: 20
Area of Rectangle = 200.0

=== Code Execution Successful ===
```

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

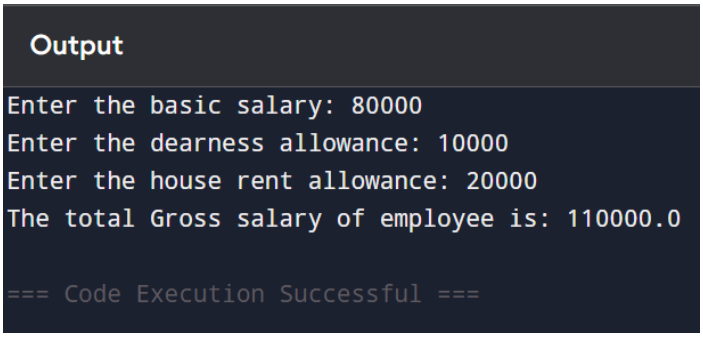
java manager

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. Find the key differences between java.util and java.lang packages.

Answer:

java.lang	**java.util**
Automatically imported	Must be imported manually
Contains core classes (String, Math, System, etc.)	Contains utility & collection classes
Used for basic language support	Used for data structures, Scanner, Date, Collections

2. List some of the subpackages of java.util

Answer:

- java.util.concurrent
- java.util.regex
- java.util.stream
- java.util.function

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: 25-02-2026**

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?

   Answer:

   The four access modifiers in Java are:

1. public – Accessible from anywhere.
2. private – Accessible only within the same class.
3. protected – Accessible within the same package and by subclasses (even in different packages).
4. default (no modifier) – Accessible only within the same package.

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

   Answer: All four access modifiers can be used to control access to class members (variables, methods, constructors):

- public
- private
- protected
- default (no modifier)
  These modifiers help implement encapsulation by restricting or allowing access to class members.

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

```
Tesla
PS C:\Users\Rohitha B\Downloads\VS java>
```

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

Output

```
Rohitha
PS C:\Users\Rohitha B\Downloads\VS java>
```

//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

```
Access via subclass method: 100
0
PS C:\Users\Rohitha B\Downloads\VS java>
```

//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

```
15
PS C:\Users\Rohitha B\Downloads\VS java>
```

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.

Answer:

In Java, the default access modifier (no modifier) allows access only within the same package. It cannot be accessed from a different package.

DefaultClass.java:

```
package package1;

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

TestSamePackage.java:

```
package package1;

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

TestDifferentPackage.java:

```
package package2;

import package1.DefaultClass;   // Error
```

```
public class TestDifferentPackage {
    public static void main(String[] args) {
        DefaultClass obj = new DefaultClass();  // Not Accessible
        obj.display();
    }
}
```

This will give a **compile-time error** because a default class is **not accessible outside its package**.

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

Answer:

The public access modifier provides the highest level of access.
A public class or member can be accessed from anywhere (same class, same package, different package, and even outside the project if imported).

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: 25-02-2026**

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?

Answer: Java does not support multiple inheritance using classes to avoid the Diamond Problem (ambiguity problem).
In C++, a class can inherit from two parent classes. If both parent classes contain the same method, the child class becomes confused about which method to use. This creates ambiguity and complexity.
To avoid this problem and keep the language simple and secure, Java:

- Does not allow multiple inheritance through classes
- Allows multiple inheritance through interfaces (because interfaces do not create ambiguity in the same way)

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

D. To allow creation of objects

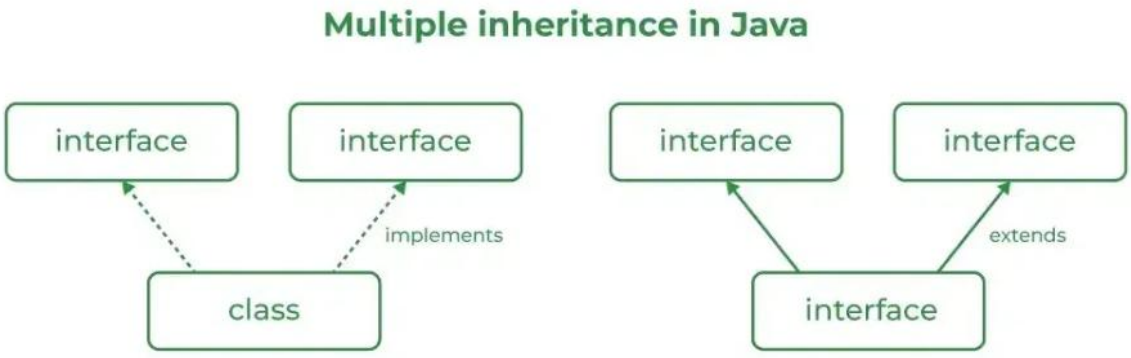
D. To improve runtime performance

Answer: 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

```
Addition : 3
Substraction : 1
PS C:\Users\Rohitha B\Downloads\VS java>
```

POST LAB EXERCISE

1. Can a functional interface extend another interface?

Answer: Yes, a functional interface in Java can extend another interface. However, it must still have only one abstract method in total (including inherited abstract methods).
If extending another interface results in more than one abstract method, then it will not be considered a functional interface.

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

A. Constructors

B. Private methods

C. Default and static methods

D. Final classes

Answer: Default and static methods

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: 25-02-2026**

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?

   Answer: Exception handling in Java is a mechanism used to handle runtime errors so that the program does not terminate abruptly.
   It allows a program to detect, catch, and handle errors using keywords like:

- try
- catch
- finally
- throw
- throws

This ensures smooth execution of the program even when unexpected errors occur.

2. What is the purpose of using try-catch blocks in exception handling?

Answer:

The purpose of using try-catch blocks is to:

- Place risky code inside the try block
- Catch and handle exceptions inside the catch block
- Prevent the program from crashing
- Provide an alternative flow when an error occurs

It helps in maintaining program stability and improving error management.

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.

```
Enter an integer: 5
Result: 2
Program execution completed.
PS C:\Users\Rohitha B\Downloads\VS java>
```

Output 2 (Valid Input - Zero):

Enter an integer: 0

Cannot divide by zero!

Program execution completed.

```
Enter an integer: 0
Cannot divide by zero!
Program execution completed.
PS C:\Users\Rohitha B\Downloads\VS java>
```

Output 3 (Invalid Input - Non-integer):

Enter an integer: abc

Invalid input! Please enter an integer.

Program execution completed.

```
Enter an integer: abc
Invalid input! Please enter an integer.
Program execution completed.
PS C:\Users\Rohitha B\Downloads\VS java>
```

POST LAB EXERCISE

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

   Answer: In Java, the Scanner object is used to take input from the user.
   It reads different types of data such as:

- int → nextInt()
- double → nextDouble()
- String → nextLine()

  It is commonly used with System.in to accept input from the keyboard**.**

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

   Answer: The expected exceptions depend on the operations inside the try block.
   Common exceptions include:

1. ArithmeticException – If a number is divided by zero.
2. InputMismatchException – If the user enters data of the wrong type (e.g., entering text instead of an integer using Scanner).
3. NumberFormatException – If a string is incorrectly converted into a number.
4. ArrayIndexOutOfBoundsException – If an invalid array index is accessed.
   These exceptions occur due to invalid input or incorrect operations performed during program execution.

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: 25-02-2026**

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?

   Answer: In Java, a custom exception is a user-defined exception created by extending the Exception class or RuntimeException class.
   It is used to handle application-specific errors that are not covered by built-in exceptions.

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

   Answer: There are two types of custom exceptions:

Checked Custom Exception

- Extends Exception
- Must be handled using try-catch or declared using throws

Unchecked Custom Exception

- Extends RuntimeException
- Handling is optional (not checked at compile time)

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.

```
Caught Exception: Age must be 18 or above.
PS C:\Users\Rohitha B\Downloads\VS java>
```

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.

```
Caught Exception: Division by zero is not allowed.
PS C:\Users\Rohitha B\Downloads\VS java>
```

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

    Answer: Extend Exception or RuntimeException

2. List some use cases for the checked and unchecked exceptions.

Answer:

Checked Exceptions:

- File handling
- Database connection
- Network errors
  (Recoverable errors, must be handled)

Unchecked Exceptions:

- Divide by zero
- Null pointer access
- Wrong array index
  (Programming errors, optional handling)

ASSESSMENT

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