Experiment Number : 8 **Date: 13.2.2026**

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?

   java.util is a built-in Java package that provides utility classes and the **Collection Framework**.

   It contains collections like:

- **List** → ArrayList, LinkedList
- **Set** → HashSet, TreeSet
- **Queue** → PriorityQueue
- **Map** → HashMap, TreeMap

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

   Built-in Packages (Predefined Packages)

   These are provided by Java.

   Examples:

- java.lang
- java.util
- java.io
- java.sql

User-defined Packages

These are created by the programmer using the package keyword.

Example:

package mypackage;

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: 14

```
[Running] cd "c:\Users\msand\OneDrive\Desktop\
Random number: 14

[Done] exited with code=0 in 1.097 seconds
```

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!

```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\" && javac Main.java && java Main
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]

```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\" && javac Main.java && java Main
[3, 5, 7]

[Done] exited with code=0 in 0.998 seconds
```

POST LAB EXERCISE

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

   If two classes with the same name are imported from different packages,
   Java will give a compilation error (ambiguity error).
   Example:
   import java.util.Date;
   import java.sql.Date; // Error: Date is ambiguous
   Solution:
   Use the **fully qualified name**:

java.util.Date d1 = new java.util.Date();
java.sql.Date d2 = new java.sql.Date(0);

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

Purpose of using packages in Java

Packages are used to:

- Organize classes
- Avoid name conflicts
- Provide access protection
- Improve code reusability
- Make large projects manageable

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:

C. java.awt

java.awt (Abstract Window Toolkit) is used to create GUI components like:

- Window
- Button
- Label
- Frame

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: 13.2.2026**

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?

   Import a single class:

   import java.util.Scanner;

   Import multiple classes (entire package):

   import java.util.*;

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

   java.lang

   It contains commonly used classes like:

   String

   System

   Math

   Object

   No need to import java.lang 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

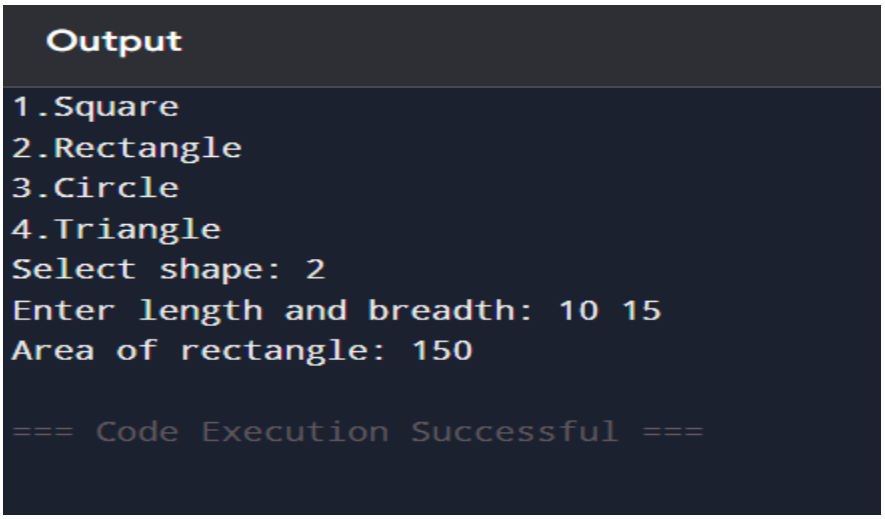
1. Square
2. Rectangle
3. Circle
4. Triangle

   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

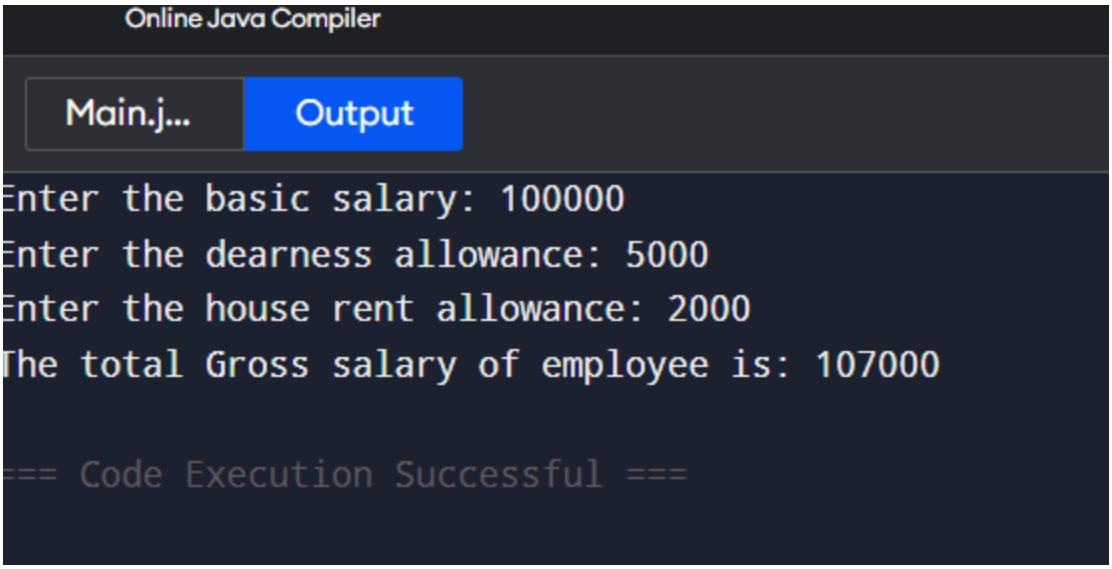
java manager

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

Feature	**java.lang**	**java.util**
Import	Imported automatically	Must be imported manually
Purpose	Core language classes	Utility & collection classes
Examples	String, System, Math, Object	Scanner, ArrayList, HashMap, Date
Usage	Basic Java functionality	Data structures & utilities

2. List some of the subpackages of java.util

Some Subpackages of java.util

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

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: 21.2.2026**

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?

   public – Accessible from anywhere (inside and outside the package).

   protected – Accessible within the same package and in subclasses (even in different packages).
   default (package-private) – Accessible only within the same package. *(No keyword is used.)*
   private – Accessible only within the same class.

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

   To control access to class members (variables, methods, constructors), Java allows:

   - public
   - protected
   - default (package-private)
   - private

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

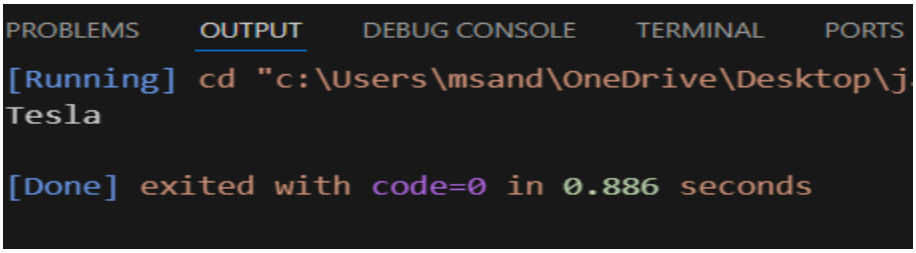
Fig: Comparison table of Access Modifiers in Java

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



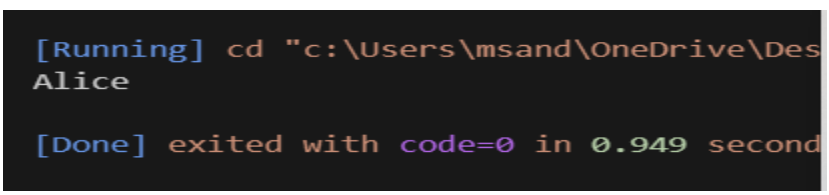
//Private Access Modifier

```
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
```

```
[Running] cd "c:\Users\msand\OneDrive\Desktop\
Access via subclass method: 100
0

[Done] exited with code=0 in 1.036 seconds
```

//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

```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\
15

[Done] exited with code=0 in 0.851 seconds
```

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.

Default access modifier means **no modifier is specified**.
It is accessible **only within the same package** and **NOT accessible from a different package.**
Example:

```
// Same package → Works
class DefaultClass {
  void display() {
    System.out.println("Hello");
  }
}
```

2. Which access modifier provides the highest level of access?
   public provides the highest level of access.
   Access order:
   public > protected > default > private

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: 21.2.2025**

Implementation of multiple inheritance in Java using Interface

Aim:

Write a Java program to implement multiple inheritance using Interface.

PRE LAB EXERCISE

QUESTIONS

3. Why Java does not support multiple inheritance using classes like that of C?
   Java does not support multiple inheritance using classes to avoid the **Diamond Problem** (ambiguity when two parent classes have the same method).
   It reduces complexity and makes the program simpler and more secure.
4. 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

   Correct Answer: B. To define a contract of methods a class must implement

   An interface defines methods that a class must implement, ensuring abstraction and multiple inheritance support.

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

```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\" && javac Main.java && jav
Addition : 3
Subtraction : 1

[Done] exited with code=0 in 0.937 seconds
```

POST LAB EXERCISE

1. Can a functional interface extend another interface?
   Y**es**, a functional interface can extend another interface
   But it must still have **only one abstract method** (including inherited ones).
   Otherwise, it will not be 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

Correct Answer: C. Default and static methods

Java 8 introduced:

- **Default methods**
- **Static methods**

to allow method implementation inside interfaces.

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: 22.2.2026**

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 in Java to **handle runtime errors** so that the program does not terminate abruptly and runs normally.
2. What is the purpose of using try-catch blocks in exception handling?
   The try block contains code that may cause an exception.
   The catch block handles the exception and prevents the program from crashing.

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.");
        } }
}
```

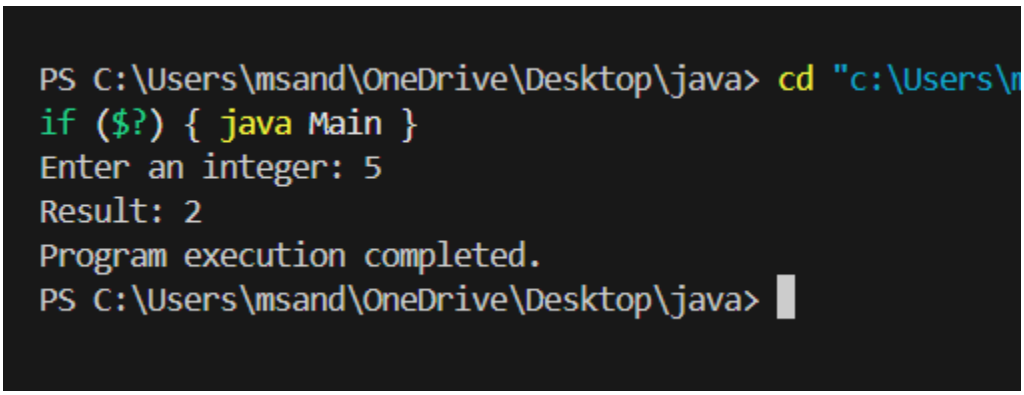
Outputs

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.

```
Enter an integer: 0
Cannot divide by zero!
Program execution completed.
PS C:\Users\msand\OneDrive\Desktop\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\msand\OneDrive\Desktop\java>
```

POST LAB EXERCISE

1. What is the purpose of the Scanner object in the Java program?
   The **Scanner** object is used to **read input from the user** (keyboard input).
   In this program, it reads an integer using nextInt().

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

1. **InputMismatchException**
   - Thrown if the user enters **non-integer input** (like text instead of a number).
2. **ArithmeticException**
   - Thrown if the user enters **0**, because dividing by zero is not allowed.

   These exceptions are handled using catch blocks to prevent program crash.

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: 23.2.2026**

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 the Exception or RuntimeException class to handle specific application errors.
2. What are the two types of custom exceptions in Java?

   Checked Custom Exception
   Extends Exception

   Must be handled using try-catch or declared using throws

   Unchecked Custom Exception
   Extends Runtime Exception
   No need to handle it compulsorily

IN LAB EXERCISE

Objective

To understand and implement Checked and Unchecked custom exceptions.

Source Code

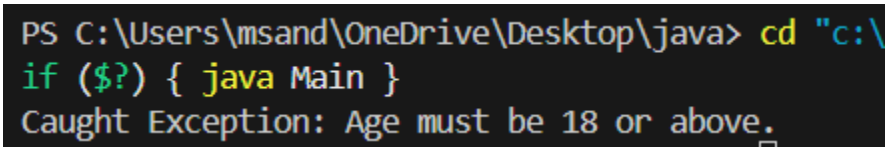
Ex 1: // Custom Checked Exception

```
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.



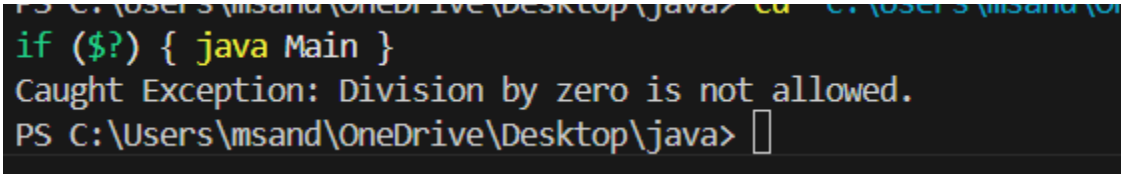
Ex 2: // Custom Unchecked Exception

```
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

Correct Answer:
C. Extend Exception or RuntimeException

Custom exceptions are created by extending:

- **Exception** → for checked exceptions
- **RuntimeException** → for unchecked exceptions

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

Unchecked Exceptions (Runtime Exceptions)
Used for programming errors.
Examples:

- Division by zero
- Null pointer access
- Array index out of bounds
- Invalid user input

ASSESSMENT

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