Practical No 3

Aim: Write a Java program to implement Ineheritance and

exception handling

Resources Required:

- · Java Development Kit (JDK)
- · Text Editor (e.g., Notepad++) or IDE (e.g., Eclipse, IntelliJ IDEA, NetBeans)
- · Command-line terminal or Java compiler

Theory:

1. Inheritance in Java:

Inheritance is one of the core principles of Object-Oriented Programming (OOP). It allows a class (known as a **subclass** or **child class**) to acquire the properties and behaviors (fields and methods) of another class (called a **superclass** or **parent class**).
Inheritance promotes **code reusability**, **method overriding**, and helps in establishing a natural hierarchy between classes.

Types of Inheritance in Java:

- **Single Inheritance**
  ➢ One class inherits from one superclass.
  Example: `class B extends A {}`
- **Multilevel Inheritance**
  ➢ A class inherits from a derived class, forming a chain.
  Example: `class C extends B extends A {}`
- **Hierarchical Inheritance**
  ➢ Multiple classes inherit from a single parent class.
  Example:
  class A {}
  class B extends A {}
  class C extends A {}
  Java does not support multiple inheritance using classes (to avoid ambiguity), but it is supported using **interfaces.**

2. Exception Handling in Java:

Exception Handling is a mechanism to handle runtime errors, ensuring the normal flow of the application. An **exception** is an unwanted or unexpected event that occurs during program execution, such as dividing by zero or accessing an invalid array index.
Java provides robust exception-handling tools to catch and respond to these errors.

Types of Exceptions:

- **Checked Exceptions**
  ➢ Checked at compile time (e.g., `IOException`, `SQLException`).
- **Unchecked Exceptions**
  ➢ Occur at runtime (e.g., `ArithmeticException`, `NullPointerException`, `ArrayIndexOutOfBoundsException`).
- **Errors**
  ➢ Serious problems that applications should not try to catch (e.g., `OutOfMemoryError`)

✅ Exception Handling Keywords in Java

Keyword	Purpose
`try`	Defines a block of code to be tested for exceptions.
`catch`	Catches and handles the exception thrown in the `try` block.
`finally`	A block that always executes, whether an exception is thrown or not. Typically used for cleanup code.
`throw`	Used to explicitly throw an exception.
`throws`	Declares exceptions that a method might throw.

1. Source Code: Inheritance

```
// Superclass
class Vehicle {
    String brand;

    Vehicle(String brand) {
        this.brand = brand;
    }

    void showBrand() {
        System.out.println("Vehicle Brand: " + brand);
    }
}

// Subclass
class Car extends Vehicle {
    String model;

    Car(String brand, String model) {
        super(brand);  // Call constructor of superclass
        this.model = model;
    }
```

```
    void showModel() {
        System.out.println("Car Model: " + model);
    }
}

public class InheritanceDemo {
    public static void main(String[] args) {
        Car myCar = new Car("Toyota", "Corolla");
        myCar.showBrand();  // Inherited method
        myCar.showModel();  // Own method
    }
}
```

2. Source Code: Exception Handling

```
public class ExceptionHandlingDemo {
    public static void main(String[] args) {
        try {
            int a = 10;
            int b = 0;
            int result = a / b;  // This will throw ArithmeticException
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: Division by zero is not allowed.");
        } finally {
            System.out.println("Finally block executed.");
        }

        System.out.println("Program continues after exception handling.");
    }
}
```

Output:

Conclusion:

This Java program successfully demonstrates the implementation of inheritance and exception handling.