*Labs:*

1. Classes & Objects

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

public class Student {

String name;

int rollNo;

public static void main(String[] args) {

Student s = new Student();

s.name = "Alice";

s.rollNo = 101;

System.out.println(s.name + " - " + s.rollNo);

}

}

2. Constructors & Overloading

class Box {

int width, height;

// Constructor overloading

Box() { width = height = 0; } // Default

Box(int w, int h) { width = w; height = h; } // Parameterized

// Method overloading

void print() { System.out.println("Box: " + width + "x" + height); }

void print(String label) { System.out.println(label + ": " + width + "x" + height); }

}

public class Main {

public static void main(String[] args) {

Box b1 = new Box(); // Default constructor

Box b2 = new Box(10, 20); // Parameterized

b1.print(); // Box: 0x0

b2.print("Custom Box"); // Custom Box: 10x20

}

}

3. Inheritance Types

Single Inheritance

class Animal { void eat() { System.out.println("Eating..."); } }

class Dog extends Animal { void bark() { System.out.println("Barking..."); } }

public class Main {

public static void main(String[] args) {

Dog d = new Dog();

d.eat(); // Eating...

d.bark(); // Barking...

}

}

Multilevel Inheritance

class Animal { void eat() { System.out.println("Eating..."); } }

class Dog extends Animal { void bark() { System.out.println("Barking..."); } }

class Puppy extends Dog { void weep() { System.out.println("Weeping..."); } }

public class Main {

public static void main(String[] args) {

Puppy p = new Puppy();

p.eat(); // Eating...

p.bark(); // Barking...

p.weep(); // Weeping...

}

}

Hierarchical Inheritance

class Animal { void eat() { System.out.println("Eating..."); } }

class Dog extends Animal { void bark() { System.out.println("Barking..."); } }

class Cat extends Animal { void meow() { System.out.println("Meowing..."); } }

public class Main {

public static void main(String[] args) {

Dog d = new Dog();

Cat c = new Cat();

d.eat(); // Eating...

d.bark(); // Barking...

c.meow(); // Meowing...

}

}

4. Multiple Interfaces (Multiple Inheritance)

interface Drawable { void draw(); }

interface Printable { void print(); }

class Circle implements Drawable, Printable {

public void draw() { System.out.println("Drawing Circle"); }

public void print() { System.out.println("Printing Circle"); }

}

public class Main {

public static void main(String[] args) {

Circle c = new Circle();

c.draw(); // Drawing Circle

c.print(); // Printing Circle

}

}

5. Multi-Threading

class MyThread extends Thread {

public void run() {

for (int i = 0; i < 3; i++) {

System.out.println("Thread running...");

try { Thread.sleep(500); } catch (Exception e) {}

}

}

}

public class Main {

public static void main(String[] args) {

MyThread t = new MyThread();

t.start(); // Thread starts execution

}

}

6. Packages

// File: mypackage/Calculator.java

package mypackage;

public class Calculator {

public int add(int a, int b) { return a + b; }

}

// File: Main.java

import mypackage.Calculator;

public class Main {

public static void main(String[] args) {

Calculator calc = new Calculator();

System.out.println(calc.add(5, 3)); // 8

}

}

7. Exception Handling

1. Checked Exceptions (must be handled)

a) IOException

import java.io.\*;

public class IOExceptionExample {

public static void main(String[] args) {

try {

FileReader fr = new FileReader("test.txt"); // File might not exist

} catch (IOException e) {

System.out.println("IOException caught: " + e);

}

}

}

b) InterruptedException

public class InterruptedExceptionExample {

public static void main(String[] args) {

try {

Thread.sleep(1000); // Might be interrupted

System.out.println("Woke up after 1 second");

} catch (InterruptedException e) {

System.out.println("InterruptedException caught: " + e);

}

}

}

c) SQLException (Needs JDBC, so simplified here)

import java.sql.\*;

public class SQLExceptionExample {

public static void main(String[] args) {

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/test", "root", "pass");

} catch (SQLException e) {

System.out.println("SQLException caught: " + e);

}

}

}

2. Unchecked Exceptions (runtime errors, not forced to handle)

a) ArithmeticException

public class ArithmeticExceptionExample {

public static void main(String[] args) {

int a = 10 / 0; // Division by zero

}

}

b) NullPointerException

public class NullPointerExceptionExample {

public static void main(String[] args) {

String str = null;

System.out.println(str.length()); // Null object

}

}

c) ArrayIndexOutOfBoundsException

public class ArrayIndexOutOfBoundsExample {

public static void main(String[] args) {

int[] arr = {1, 2, 3};

System.out.println(arr[5]); // Index out of range

}

}

d) NumberFormatException

public class NumberFormatExceptionExample {

public static void main(String[] args) {

String s = "abc";

int num = Integer.parseInt(s); // Invalid number format

}

}

3. Error Example (not an Exception, can't be caught usually)

a) StackOverflowError

public class StackOverflowErrorExample {

public static void recursiveCall() {

recursiveCall(); // Infinite recursion

}

public static void main(String[] args) {

recursiveCall(); // Will throw StackOverflowError

}

}

8. Generics

class Box<T> {

T content;

void set(T t) { content = t; }

T get() { return content; }

}

public class Main {

public static void main(String[] args) {

Box<String> sBox = new Box<>();

sBox.set("Hello Generics!");

System.out.println(sBox.get()); // Hello Generics!

}

}

9. Collection Framework

import java.util.\*;

public class Main {

public static void main(String[] args) {

// ArrayList

ArrayList<String> list = new ArrayList<>();

list.add("Java");

list.add("Python");

System.out.println("ArrayList: " + list); // [Java, Python]

// HashSet

HashSet<Integer> set = new HashSet<>();

set.add(10);

set.add(20);

set.add(10); // Duplicate ignored

System.out.println("HashSet: " + set); // [20, 10]

}

}

10. File Handling

1. Create a File

import java.io.File;

public class CreateFile {

public static void main(String[] args) {

try {

File file = new File("myfile.txt");

if (file.createNewFile()) {

System.out.println("File created: " + file.getName());

} else {

System.out.println("File already exists.");

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}

2. Write to a File

import java.io.FileWriter;

public class WriteFile {

public static void main(String[] args) {

try {

FileWriter writer = new FileWriter("myfile.txt");

writer.write("Hello, this is line 1.\n");

writer.write("This is line 2.");

writer.close();

System.out.println("Successfully wrote to file.");

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}

3. Read from a File

import java.io.File;

import java.util.Scanner;

public class ReadFile {

public static void main(String[] args) {

try {

File file = new File("myfile.txt");

Scanner reader = new Scanner(file);

while (reader.hasNextLine()) {

String line = reader.nextLine();

System.out.println(line);

}

reader.close();

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}