EX:1 Class & Objects

Program:

public class HelloWorld{

void dispaly()

{

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

}

public static void main(String args[])

{

HelloWorld obj = new HelloWorld();

obj.dispaly();

}

}

Output:

Hello World!

EX:2 Constructor & Inheritance

2.1 Constructor

Program:

class Person {

String name;

int age;

Person() {

name = "Unknown";

age = 0;

System.out.println("Default constructor called");

}

Person(String name, int age) {

this.name = name;

this.age = age;

System.out.println("Parameterized constructor called");

}

Person(Person p) {

this.name = p.name;

this.age = p.age;

System.out.println("Copy constructor called");

}

void display() {

System.out.println("Name: " + name + ", Age: " + age);

}

}

public class Constructor {

public static void main(String[] args) {

Person p1 = new Person();

p1.display();

Person p2 = new Person("Alice", 25);

p2.display();

Person p3 = new Person(p2);

p3.display();

}

}

Output:

Default constructor called

Name: Unknown, Age: 0

Parameterized constructor called

Name: Alice, Age: 25

Copy constructor called

Name: Alice, Age: 25

2.2 Inheritance

Program:

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...");

}

}

class Cat extends Animal {

void meow() {

System.out.println("Meowing...");

}

}

public class Inheritence {

public static void main(String[] args) {

Dog dog = new Dog();

dog.eat();

dog.bark();

Puppy puppy = new Puppy();

puppy.eat();

puppy.bark();

puppy.weep();

Cat cat = new Cat();

cat.eat();

cat.meow(); }

}

Output:

Eating...

Barking...

Eating...

Barking...

Weeping...

Eating...

Meowing...

EX:3 Interfaces & Packages

3.1 Interfaces

interface Animal {

void sound();

}

class Dog implements Animal {

public void sound() {

System.out.println("Dog barks");

}

}

public class Inter {

public static void main(String[] args) {

Dog myDog = new Dog();

myDog.sound();

}

}

Output:

Dog barks

3.2 Packages

package MyPackage;

public class Helper {

public void showMessage() {

System.out.println("Hello from the package!");

}

}

import MyPackage.Helper;

public class Main {

public static void main(String[] args) {

Helper helper = new Helper();

helper.showMessage();

}

}

Output:

Hello from the package!

EX:4 Applet & Java Swing

4.1 Applet

import java.applet.Applet;

import java.awt.Graphics;

/\*<applet code="appl.class" width="300" height="100"></applet>\*/

public class appl extends Applet {

int num1 = 10;

int num2 = 20;

int sum;

public void init() {

sum = num1 + num2;

}

public void paint(Graphics g) {

g.drawString("Number 1: " + num1, 20, 20);

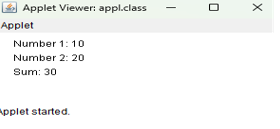
g.drawString("Number 2: " + num2, 20, 40);

g.drawString("Sum: " + sum, 20, 60);

}

}

Output:



4.2 Java Swing

import javax.swing.\*;

import java.awt.event.\*;

public class Swing extends JFrame {

JTextField n1 = new JTextField(5), n2 = new JTextField(5), res = new JTextField(5);

JButton add = new JButton("Add");

public Swing() {

res.setEditable(false);

add.addActionListener(e -> {

int sum = Integer.parseInt(n1.getText()) + Integer.parseInt(n2.getText());

res.setText(String.valueOf(sum));

});

JPanel p = new JPanel();

p.add(n1); p.add(new JLabel("+")); p.add(n2); p.add(add); p.add(res);

add(p);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

pack();

setLocationRelativeTo(null);

}

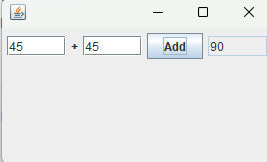
public static void main(String[] args) {

new Swing().setVisible(true);

}

}

Output:



EX:5 Event Handling Mechanism

import javax.swing.\*;

import java.awt.event.\*;

public class SimpleEvent extends JFrame implements ActionListener {

JButton button;

public SimpleEvent() {

button = new JButton("Click Me");

button.addActionListener(this);

add(button);

setSize(200, 100);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

}

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(this, "Button was clicked!");

}

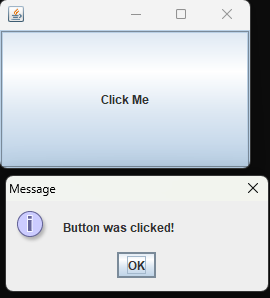
public static void main(String[] args) {

new SimpleEvent().setVisible(true);

}

}

Output:



EX:6 Exception Handling

import java.util.Scanner;

public class ExceptionExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter a number: ");

int num = scanner.nextInt();

int result = 100 / num; // Could cause ArithmeticException if num = 0

System.out.println("100 divided by " + num + " is " + result);

} catch (ArithmeticException e) {

System.out.println("Error: Cannot divide by zero!");

} catch (Exception e) {

System.out.println("Invalid input!");

} finally {

System.out.println("Program ended.");

scanner.close();

}

}

}

Output:

Enter a number: 10

100 divided by 10 is 10

Program ended.

EX:7 I/O Streams

Program:

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class FileCopyExample {

public static void main(String[] args) {

String sourceFile = "input.txt"; // Source file to read from

String destFile = "output.txt"; // Destination file to write to

try (FileInputStream inputStream = new FileInputStream(sourceFile);

FileOutputStream outputStream = new FileOutputStream(destFile)) {

int byteData;

// Read one byte at a time from input and write to output

while ((byteData = inputStream.read()) != -1) {

outputStream.write(byteData);

}

System.out.println("File copied successfully.");

} catch (IOException e) {

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

}

}

}

Output:

FileInputStream

FileOutputStream

EX:8 Java Multithreading

Program:

class MyThread extends Thread {

private String threadName;

public MyThread(String name) {

this.threadName = name;

}

@Override

public void run() {

// This code runs in a separate thread

for (int i = 1; i <= 5; i++) {

System.out.println(threadName + " is running: " + i);

try {

Thread.sleep(500); // Sleep for 500 milliseconds

} catch (InterruptedException e) {

System.out.println(threadName + " interrupted.");

}

}

System.out.println(threadName + " finished.");

}

}

public class MultiThreadExample {

public static void main(String[] args) {

MyThread t1 = new MyThread("Thread A");

MyThread t2 = new MyThread("Thread B");

t1.start(); // Start first thread

t2.start(); // Start second thread

System.out.println("Main thread finished.");

} }

Output:

Main thread finished.

Thread B is running: 1

Thread A is running: 1

Thread A is running: 2

Thread B is running: 2

Thread A is running: 3

Thread B is running: 3

Thread A is running: 4

Thread B is running: 4

Thread A is running: 5

Thread B is running: 5

Thread B finished.

Thread A finished.

EX:9 Java Networking

Program:

(EchoServer.java)

import java.io.\*;

import java.net.\*;

public class EchoServer {

public static void main(String[] args) {

int port = 12345; // Port number

try (ServerSocket serverSocket = new ServerSocket(port)) {

System.out.println("Server started. Listening on port " + port);

while (true) {

Socket clientSocket = serverSocket.accept();

System.out.println("Client connected.");

new Thread(() -> handleClient(clientSocket)).start();

}

} catch (IOException e) {

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

}

}

private static void handleClient(Socket clientSocket) {

try (

BufferedReader in = new BufferedReader(

new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)

) {

String line;

while ((line = in.readLine()) != null) {

System.out.println("Received from client: " + line);

out.println("Echo: " + line); // Echo back

if ("bye".equalsIgnoreCase(line)) {

break;

}

}

clientSocket.close();

System.out.println("Client disconnected.");

} catch (IOException e) {

System.out.println("Client handler exception: " + e.getMessage());

}

}

}

(EchoClient.java):

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class EchoClient {

public static void main(String[] args) {

String hostname = "localhost";

int port = 12345;

try (Socket socket = new Socket(hostname, port);

BufferedReader in = new BufferedReader(

new InputStreamReader(socket.getInputStream()));

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

Scanner scanner = new Scanner(System.in)) {

System.out.println("Connected to the server");

String userInput;

while (true) {

System.out.print("Enter message (bye to quit): ");

userInput = scanner.nextLine();

out.println(userInput); // Send to server

String response = in.readLine(); // Read server response

System.out.println("Server response: " + response);

if ("bye".equalsIgnoreCase(userInput)) {

break;

}

}

} catch (IOException e) {

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

}

}

}

Output: (Server terminal)

Server started. Listening on port 12345

Client connected.

Received from client: Hello server

Received from client: bye

Client disconnected.

Output:( Client terminal):

Connected to the server

Enter message (bye to quit): Hello server

Server response: Echo: Hello server

Enter message (bye to quit): bye

Server response: Echo: bye

Java Database Connectivity

EX:10

Program:

import java.sql.\*;

import java.util.Scanner;

public class SQLiteExample {

public static void main(String[] args) {

String url = "jdbc:sqlite:people.db";

String createTableSQL = "CREATE TABLE IF NOT EXISTS persons ("

+ "id INTEGER PRIMARY KEY AUTOINCREMENT,"

+ "name TEXT NOT NULL,"

+ "age INTEGER NOT NULL"

+ ");";

String insertSQL = "INSERT INTO persons(name, age) VALUES(?, ?)";

String selectSQL = "SELECT \* FROM persons";

try (Connection conn = DriverManager.getConnection(url);

Statement stmt = conn.createStatement();

Scanner scanner = new Scanner(System.in)) {

stmt.execute(createTableSQL);

System.out.print("Enter name: ");

String name = scanner.nextLine();

System.out.print("Enter age: ");

int age = Integer.parseInt(scanner.nextLine());

try (PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

pstmt.setString(1, name);

pstmt.setInt(2, age);

pstmt.executeUpdate();

}

System.out.println("\nStored Data:");

try (ResultSet rs = stmt.executeQuery(selectSQL)) {

while (rs.next()) {

int id = rs.getInt("id");

String personName = rs.getString("name");

int personAge = rs.getInt("age");

System.out.printf("ID: %d, Name: %s, Age: %d%n", id, personName, personAge);

}

}

} catch (SQLException e) {

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

} catch (NumberFormatException e) {

System.out.println("Invalid age input. Please enter a number.");

}

}

}

Output:

Enter name: leo joel

Enter age: 20

Stored Data:

ID: 1, Name: Alice, Age: 25

ID: 2, Name: Bob, Age: 30

ID: 3, Name: leo joel, Age: 20

EX:11 Java Servlets

Program:

import java.io.IOException;

import java.io.PrintWriter;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/hello")

public class HelloServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

try (PrintWriter out = response.getWriter()) {

out.println("<html><body>");

out.println("<h1>Hello from Servlet!</h1>");

out.println("</body></html>");

}

}

}

OutPut: (View in browser)

Hello from Servlet!

EX:12 Java Beans

Program:

(Person.java)

import java.io.Serializable;

public class Person implements Serializable {

private String name;

private int age;

public Person() { }

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

OutPut:

Name: Alice

Age: 25