**OOP LAB – JIT2311**

EX. No:1(a)

Program:

#include <iostream> #include <string> using namespace std; class Student

{

private: int rollNumber; int age; string name; public: void inputInfo()

{

cout << "Enter Roll Number: "; cin >> rollNumber; cout << "Enter Age: "; cin >> age; cout << "Enter Name: "; cin >> name;

}

void displayInfo()

{

cout << "\nPersonal Information:\n"; cout << "Roll Number: " << rollNumber << endl; cout << "Age: " << age << endl; cout << "Name: " << name << endl;

}

};

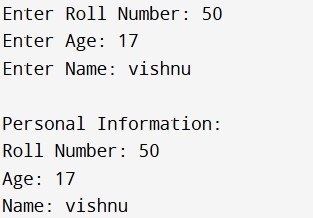
int main()

{

Student student; student.inputInfo(); student.displayInfo();

}

Output:



EX. No: 1(b)

Program:

#include <iostream> #include <string> using namespace std;

class Student

{

private: int rollno; double percentage;

public:

// Default Constructor

Student()

{

rollno = 0; percentage = 0.0;

}

// Parameterized Constructor

Student(int r, double p)

{ rollno = r; percentage = p;

}

// Copy Constructor

Student(const Student &s)

{

rollno = s.rollno; percentage = s.percentage;

}

void display()

{

cout << "Roll No: " << rollno << endl; cout << "Percentage: " << percentage << "%" << endl;

}

};

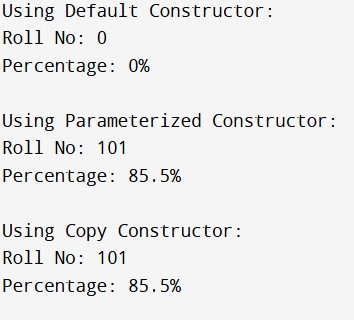
int main() { Student s1; cout << "Using Default Constructor:" << endl; s1.display(); cout << endl;

Student s2(101, 85.5); cout << "Using Parameterized Constructor:" << endl; s2.display(); cout << endl;

Student s3 = s2; cout << "Using Copy Constructor:" << endl; s3.display(); cout << endl;

}

Output:



Ex. No:2

Program:

#include <iostream> using namespace std;

class MyClass { private:

static int staticVariable;

public:

static void staticFunction() { cout << "Value of staticVariable: " << staticVariable << endl;

}

};

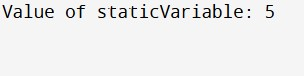
int MyClass::staticVariable = 5;

int main() {

MyClass::staticFunction(); return 0;

}

Output:



Ex. No: 3

Program:

#include <iostream> using namespace std;

class Square { private:

double side; public:

Square(double s) : side(s) {}

double area() { return side \* side;

}

};

class Rectangle { private:

double length, width; public:

Rectangle(double l, double w) : length(l), width(w) {}

double area() { return length \* width;

}

};

class Circle { private:

double radius; public:

Circle(double r) : radius(r) {}

double area() { return 3.14 \* radius \* radius; // Using direct value of π

}

};

int main() { int choice; double a, b, r;

do {

// Display menu cout << "\nArea Calculation Menu\n"; cout << "1. Calculate area of Square\n"; cout << "2. Calculate area of Rectangle\n"; cout << "3. Calculate area of Circle\n"; cout << "Enter your choice: "; cin >> choice;

switch (choice) { case 1:

cout << "Enter the side length of the square: ";

cin >> a;

{

Square sq(a); cout << "Area of the square: " << sq.area() << endl; } break; case 2:

cout << "Enter the length and width of the rectangle: ";

cin >> a >> b;

{

Rectangle rect(a, b); cout << "Area of the rectangle: " << rect.area() << endl;

} break; case 3:

cout << "Enter the radius of the circle: ";

cin >> r;

{

Circle cir(r); cout << "Area of the circle: " << cir.area() << endl;

} break;

default:

cout << "Invalid choice. Please enter a valid option.\n";

break;

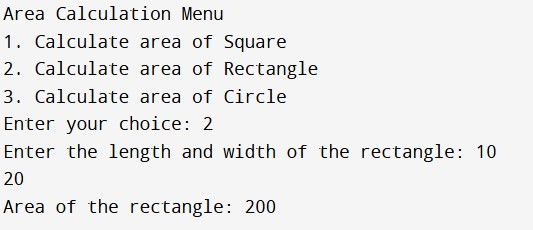
}

} while (choice != 4);

return 0;

}

Output:



Ex. No: 4(a)

Program

#include <iostream>

using namespace std;

class Counter { private: int count;

public:

Counter() : count(0) {}

Counter& operator++() { ++count; return \*this;

}

Counter& operator--() {

--count; return \*this;

}

int getCount() const { return count;

}

};

int main() {

Counter c;

cout << "Initial value: " << c.getCount() << endl;

++c; cout << "After increment: " << c.getCount() << endl;

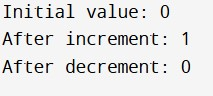
--c;

cout << "After decrement: " << c.getCount() << endl;

return 0;

}

Output:



Ex. No: 4(b)

Program

#include <iostream>

#include <string>

using namespace std;

class StrConc { private:

string concatenatedStr;

public:

StrConc() : concatenatedStr("") {}

StrConc operator+(const string& str) { StrConc result; result.concatenatedStr = concatenatedStr + str; return result;

}

void display() { cout << "Concatenated String: " << concatenatedStr << endl;

}

string getConcatenatedStr() const { return concatenatedStr;

}

};

int main() { int numStrings; cout << "Enter the number of strings to concatenate: "; cin >> numStrings;

StrConc concatenator;

for (int i = 0; i < numStrings; ++i) { string input; cout << "Enter string " << i + 1 << ": "; cin >> input;

// Use the overloaded + operator concatenator = concatenator + input;

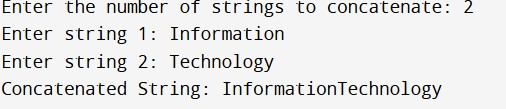
}

concatenator.display();

return 0;

}

Output



Ex. No: 5 (a)

Program

#include <iostream>

template <typename T> void swapValues(T &a, T &b) { T temp = a; a = b; b = temp;

}

class Swapper { public:

template <typename T> void swapAndDisplay(T &a, T &b) { std::cout << "Original values:" << std::endl; std::cout << "a = " << a << std::endl; std::cout << "b = " << b << std::endl;

swapValues(a, b);

std::cout << "Swapped values:" << std::endl; std::cout << "a = " << a << std::endl; std::cout << "b = " << b << std::endl; std::cout << std::endl;

}

};

int main() {

Swapper swapper;

int intA, intB;

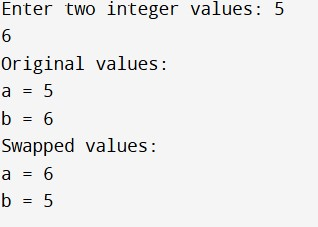
std::cout << "Enter two integer values: "; std::cin >> intA >> intB;

swapper.swapAndDisplay(intA, intB);

return 0;

}

Output:



Ex. No 5(b)

Program

#include <iostream>

using namespace std;

template <typename T> class MaxFinder { private:

T num1, num2, num3;

public:

MaxFinder(T a, T b, T c) : num1(a), num2(b), num3(c) {}

T getMax() { T max = num1; if (num2 > max) max = num2; if (num3 > max) max = num3; return max;

}

};

int main() {

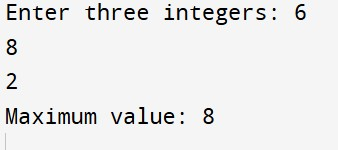
int a, b, c; cout << "Enter three integers: "; cin >> a >> b >> c; MaxFinder<int> finder(a, b, c); int max = finder.getMax();

cout << "Maximum value: " << max << endl;

return 0;

}

Output:



Ex no 6(A)

Program:

#include <iostream>

using namespace std;

int main() {

int p, q;

cout << "Enter two integers to divide: ";

cin >> p >> q;

try {

if (q != 0) {

float result = p / (float)q;

cout << "Result: " << result << endl;

} else {

throw q;

}

} catch (int) {

cout << "Division by zero" << endl;

}

return 0;

}

Output:

Enter two integers to divide: 50

10

Result: 5

Ex no 6(B)

#include <iostream>

Using namespace std;

Int main() {

Int arr[5] = {10, 20, 30, 40, 50};

Try {

Int index;

Cout << “Enter index (0-4): “;

Cin >> index;

If (index == 5)

Throw out\_of\_range(“Index out of range”);

Cout << “Value: “ << arr[index] << endl;

}

Catch (const out\_of\_range& e) {

Cout << “Error: “ << e.what() << endl;

}

Return 0;

}

Output

Enter index (0-4): 5

Error: Index out of range

7A

import java.util.Scanner;

public class SimpleArithmetic {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int a = sc.nextInt(), b = sc.nextInt();

System.out.println("Sum: " + (a + b));

System.out.println("Difference: " + (a - b));

System.out.println("Product: " + (a \* b));

System.out.println("Quotient: " + (a / b));

System.out.println("Remainder: " + (a % b));

sc.close();

}

}

Output:

10

3

Sum: 13

Difference: 7

Product: 30

Quotient: 3

Remainder: 1

7B

class Rectangle {

private double length, width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public Rectangle getInstance() {

return this;

}

public void printDetails() {

System.out.println("Length: " + length);

System.out.println("Width: " + width);

System.out.println("Area: " + (length \* width));

System.out.println("Perimeter: " + (2 \* (length + width)));

}

}

public class Main {

public static void main(String[] args) {

Rectangle rectangle = new Rectangle(5.0, 3.0).getInstance();

rectangle.printDetails();

}

}

Output

Length: 5.0

Width: 3.0

Area: 15.0

Perimeter: 16.0

8

import java.util.Scanner;

class Employee {

protected int id;

protected String name;

protected int age;

protected double basicSalary;

public void getData() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter Employee ID: ");

id = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Employee Name: ");

name = scanner.nextLine();

System.out.print("Enter Employee Age: ");

age = scanner.nextInt();

System.out.print("Enter Basic Salary: ");

basicSalary = scanner.nextDouble();

}

public void displayData() {

System.out.println("\n--- Employee Details ---");

System.out.println("ID: " + id);

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

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

System.out.println("Basic Salary: $" + basicSalary);

}

public double calculateSalary() {

return basicSalary;

}

}

class Programmer extends Employee {

@Override

public void getData() {

super.getData();

}

@Override

public void displayData() {

super.displayData();

}

@Override

public double calculateSalary() {

return basicSalary;

}

}

class AssistantProfessor extends Employee {

@Override

public void getData() {

super.getData();

}

@Override

public void displayData() {

super.displayData();

}

@Override

public double calculateSalary() {

return basicSalary;

}

}

class Professor extends Employee {

@Override

public void getData() {

super.getData();

}

@Override

public void displayData() {

super.displayData();

}

@Override

public double calculateSalary() {

return basicSalary;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

System.out.println("=== Employee Management System ===");

System.out.println("Select Employee Type:");

System.out.println("1. Programmer");

System.out.println("2. Assistant Professor");

System.out.println("3. Professor");

System.out.print("Enter your choice (1-3): ");

choice = scanner.nextInt();

scanner.nextLine();

Employee emp = null;

switch(choice) {

case 1:

emp = new Programmer();

break;

case 2:

emp = new AssistantProfessor();

break;

case 3:

emp = new Professor();

break;

default:

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

System.exit(0);

}

emp.getData();

emp.displayData();

double salary = emp.calculateSalary();

System.out.println("Total Salary: $" + salary);

scanner.close();

}

}

output

=== Employee Management System ===

Select Employee Type:

1. Programmer

2. Assistant Professor

3. Professor

Enter your choice (1-3): 1

Enter Employee ID: 101

Enter Employee Name: Alice

Enter Employee Age: 30

Enter Basic Salary: 5000

--- Employee Details ---

ID: 101

Name: Alice

Age: 30

Basic Salary: $5000.0

Total Salary: $5000.0

9

NOTE:

create a folder named geometry. Inside that folder, create a Java file for the rectangle class,write the package code in the rectangle class, then create a main class beside it and write the main program in it, open terminal and do " javac geometry/Rectangle.java Main.java ", then "java Main

"

PACKAGE:

package geometry;

public class Rectangle {

private double length;

private double width;

// Constructor

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

// Method to calculate area

public double area() {

return length \* width;

}

// Method to calculate perimeter

public double perimeter() {

return 2 \* (length + width);

}

// Method to display dimensions

public void display() {

System.out.println("Length: " + length);

System.out.println("Width: " + width);

System.out.println("Area: " + area());

System.out.println("Perimeter: " + perimeter());

}

}

MAIN:

import geometry.\*;

public class Main {

public static void main(String[] args) {

// Create a rectangle object

Rectangle rect = new Rectangle(5.0, 3.0);

// Display the rectangle's details

rect.display();

}

}

OUTPUT

Length: 5.0

Width: 3.0

Area: 15.0

Perimeter: 16.0

10

interface Playable {

void play();

}

class Football implements Playable {

@Override

public void play() {

System.out.println("Playing football: A team sport played with a spherical ball.");

}

}

class Volleyball implements Playable {

@Override

public void play() {

System.out.println("Playing volleyball: A team sport in which two teams are separated by a net.");

}

}

class Basketball implements Playable {

@Override

public void play() {

System.out.println("Playing basketball: A game played by two teams of five players each on a rectangular court.");

}

}

public class SportsMain {

public static void main(String[] args) {

Playable football = new Football();

Playable volleyball = new Volleyball();

Playable basketball = new Basketball();

football.play();

volleyball.play();

basketball.play();

}

}

Output:

Playing football: A team sport played with a spherical ball.

Playing volleyball: A team sport in which two teams are separated by a net.

Playing basketball: A game played by two teams of five players each on a rectangular court.

Ex no 11 A

class Table {

void printTable() { // Removed synchronized

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

System.out.println("Value: " + i);

try {

// Sleep for a while to simulate a time-consuming task

Thread.sleep(100);

} catch (InterruptedException e) {

System.out.println(e);

}

}

System.out.println();

}

}

class Thread1 extends Thread {

Table table;

Thread1(Table table) {

this.table = table;

}

public void run() {

table.printTable(); // Print simple table of values

}

}

class Thread2 extends Thread {

Table table;

Thread2(Table table) {

this.table = table;

}

public void run() {

table.printTable(); // Print simple table of values

}

}

public class ThreadSynchronizationExample {

public static void main(String[] args) {

Table table = new Table(); // Create a single Table object

Thread1 thread1 = new Thread1(table);

Thread2 thread2 = new Thread2(table);

thread1.start(); // Start thread1

thread2.start(); // Start thread2

try {

thread1.join(); // Wait for thread1 to finish

thread2.join(); // Wait for thread2 to finish

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println("tables printed.");

}

}

Output

Value: 1

Value: 1

Value: 2

Value: 2

Value: 3

Value: 3

Value: 4

Value: 4

Value: 5

Value: 5

tables printed.

Ex no 11 B

class Table {

synchronized void printTable() {

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

System.out.println("Value: " + i);

try {

// Sleep for a while to simulate a time-consuming task

Thread.sleep(100);

} catch (InterruptedException e) {

System.out.println(e);

}

}

System.out.println();

}

}

class Thread1 extends Thread {

Table table;

Thread1(Table table) {

this.table = table;

}

public void run() {

table.printTable(); // Print simple table of values

}

}

class Thread2 extends Thread {

Table table;

Thread2(Table table) {

this.table = table;

}

public void run() {

table.printTable(); // Print simple table of values

}

}

public class ThreadSynchronizationExample {

public static void main(String[] args) {

Table table = new Table(); // Create a single Table object

Thread1 thread1 = new Thread1(table);

Thread2 thread2 = new Thread2(table);

thread1.start(); // Start thread1

thread2.start(); // Start thread2

try {

thread1.join(); // Wait for thread1 to finish

thread2.join(); // Wait for thread2 to finish

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println("tables printed.");

}

}

Output

Value: 1

Value: 2

Value: 3

Value: 4

Value: 5

Value: 1

Value: 2

Value: 3

Value: 4

Value: 5

tables printed.

Ex no 12

(NOTE: its complicated to get the output of this program, you need to setup javafx and define its library paths then compline and run it)

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

public class LoginApp extends Application {

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Login Form");

GridPane grid = new GridPane();

grid.setAlignment(Pos.CENTER);

grid.setPadding(new Insets(20));

grid.setHgap(10);

grid.setVgap(10);

Label userNameLabel = new Label("Username:");

TextField userNameField = new TextField();

Label passwordLabel = new Label("Password:");

PasswordField passwordField = new PasswordField();

Button signInButton = new Button("Sign In");

grid.add(userNameLabel, 0, 0);

grid.add(userNameField, 1, 0);

grid.add(passwordLabel, 0, 1);

grid.add(passwordField, 1, 1);

grid.add(signInButton, 1, 2);

signInButton.setOnAction(event -> {

String username = userNameField.getText();

String password = passwordField.getText();

if (username.isEmpty() || password.isEmpty()) {

showAlert("Input Error", "Username and password cannot be empty.");

} else if (username.equals("admin") && password.equals("password")) {

showInfo("Login Successful", "Welcome, " + username + "!");

} else {

showAlert("Login Error", "Invalid username or password.");

}

});

Scene scene = new Scene(grid, 300, 200);

primaryStage.setScene(scene);

primaryStage.show();

}

private void showAlert(String title, String message) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

private void showInfo(String title, String message) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

public static void main(String[] args) {

launch(args);

}

}

Output

