**Guru Gobind Singh Indraprastha University**

**University School of Information and Technology**

****

**MCA(Se) 2024-2026**

**Object Oriented Programming Practical File**

**IT-611**

| **Submitted by** | **Amit Mamgai** |
| --- | --- |
| **Enrollment Number** | **2116404524** |
| **Section** | **1** |
| **Semester** | **1** |
| **Submitted to** | **Dr. Jaspreeti Singh** |

**Index**

| **S.no** | **Practical Name** | **Page no** | **Signature** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1. WAP to implement Call by Value and Call by Reference in C++. Use Call by Reference to swap two integer values. (C++).**

#include <iostream>

using namespace std;

void callByValue(int a) {

a = a + 10;

cout << "Inside callByValue - a: " << a << endl;

}

void callByReference(int &a) {

a = a + 10;

cout << "Inside callByReference - a: " << a << endl;

}

void swapByReference(int &a, int &b) {

int temp = a;

a = b;

b = temp;

cout << "Inside swapByReference - a: " << a << ", b: " << b << endl;

}

int main() {

int a,b;

cout<<"Enter the value of a and b"<<endl;

cin>>a >>b;

cout << "Before callByValue - a: " << a << endl;

callByValue(a);

cout << "After callByValue - a: " << a << endl;

cout << "\nBefore callByReference - a: " << a << endl;

callByReference(a);

cout << "After callByReference - a: " << a << endl;

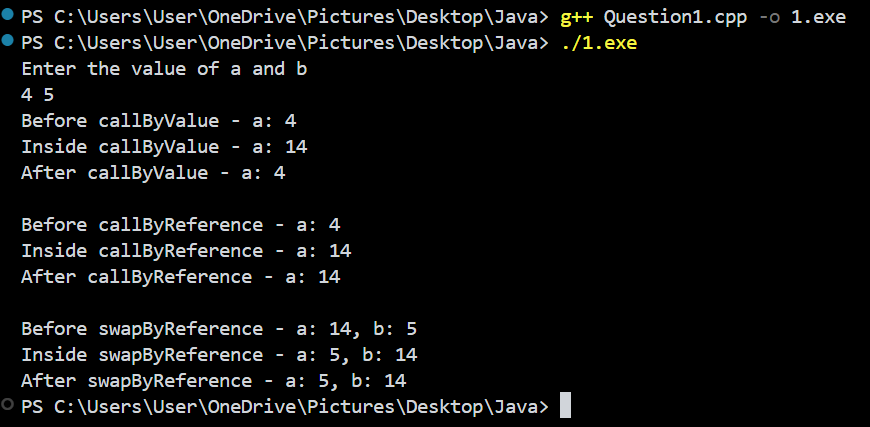
cout << "\nBefore swapByReference - a: " << a << ", b: " << b << endl;

swapByReference(a, b);

cout << "After swapByReference - a: " << a << ", b: " << b << endl;

return 0;}

**Output**



**2. WAP to implement a function to calculate the simple interest. Use the option of default value of rate of interest if it is not entered by the user. (C++).**

#include <iostream>

using namespace std;

double calculateSimpleInterest(double principal, double time, double rate = 5.0) { // Default rate is 5%

return (principal \* rate \* time) / 100;

}

int main() {

double principal, time, rate;

cout << "Enter the principal amount: ";

cin >> principal;

cout << "Enter the time period in years: ";

cin >> time;

char choice;

cout << "Do you want to enter a custom rate of interest? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

cout << "Enter the rate of interest: ";

cin >> rate;

double interest = calculateSimpleInterest(principal, time, rate);

cout << "The simple interest is: " << interest << endl;

} else {

double interest = calculateSimpleInterest(principal, time);

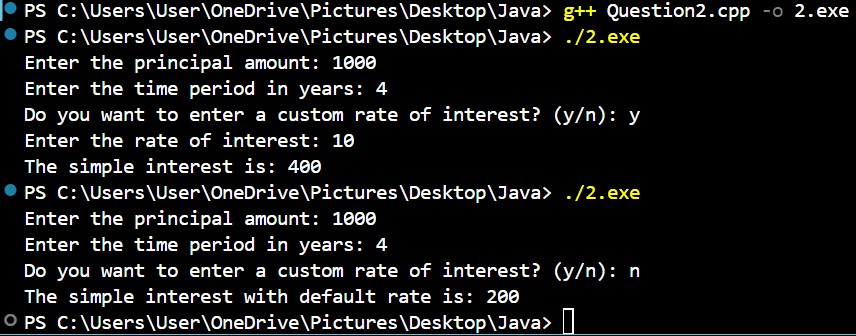
cout << "The simple interest with default rate is: " << interest << endl;

}

return 0;

}

**Output**



**3. WAP to implement the following types of constructors in a class (C++)**

**a. No-argument constructor.**

**b. One-argument constructor.**

**c. Two-argument constructor.**

**d. Copy constructor.**

#include <iostream>

using namespace std;

class Rectangle {

private:

double length;

double width;

public:

Rectangle() {

length = 1.0;

width = 1.0;

cout << "No-argument constructor called. Default Length: " << length << ", Default Width: " << width << endl;

}

Rectangle(double side) {

length = side;

width = side;

cout << "One-argument constructor called. Length: " << length << ", Width: " << width << endl;

}

Rectangle(double l, double w) {

length = l;

width = w;

cout << "Two-argument constructor called. Length: " << length << ", Width: " << width << endl;

}

Rectangle(const Rectangle &rect) {

length = rect.length;

width = rect.width;

cout << "Copy constructor called. Length: " << length << ", Width: " << width << endl;

}

double area() {

return length \* width;

}

};

int main() {

Rectangle rect1;

Rectangle rect2(5.0);

Rectangle rect3(4.0, 6.0);

Rectangle rect4(rect3);

cout << "Area of rect1: " << rect1.area() << endl;

cout << "Area of rect2: " << rect2.area() << endl;

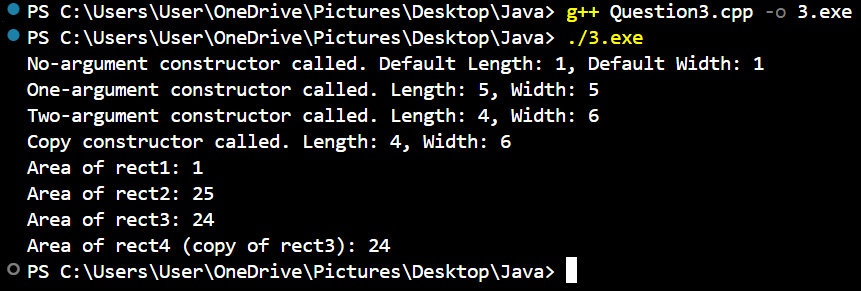
cout << "Area of rect3: " << rect3.area() << endl;

cout << "Area of rect4 (copy of rect3): " << rect4.area() << endl;

return 0;

}

**Output**



**4. Write a program to implement Multilevel Inheritance using C++.**

#include <iostream>

using namespace std;

class Animal {

public:

void speak() {

cout << "The animal makes a sound." << endl;

}

};

class Mammal : public Animal {

public:

void walk() {

cout << "The mammal walks on land." << endl;

}

};

class Dog : public Mammal {

public:

void bark() {

cout << "The dog barks." << endl;

}

};

int main() {

Dog myDog;

myDog.speak(); // From Animal class

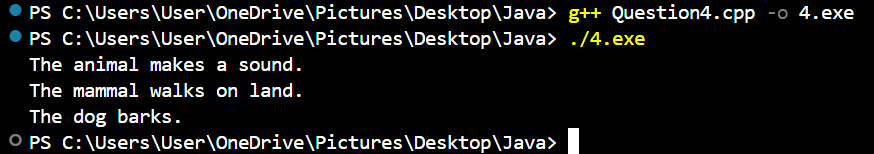
myDog.walk(); // From Mammal class

myDog.bark(); // From Dog class

return 0;

}

**Output**



**5.Write a Program to calculate the total mark of a student using the concept of virtual class(C++).**

#include <iostream>

using namespace std;

class Student {

public:

virtual void inputMarks() = 0;

virtual void displayTotalMarks() = 0;

};

class TotalMarks : public Student {

protected:

int marks1, marks2, marks3;

public:

void inputMarks() override {

cout << "Enter marks for Subject 1: ";

cin >> marks1;

cout << "Enter marks for Subject 2: ";

cin >> marks2;

cout << "Enter marks for Subject 3: ";

cin >> marks3;

}

void displayTotalMarks() override {

int total = marks1 + marks2 + marks3;

cout << "Total Marks: " << total << endl;

}

};

int main() {

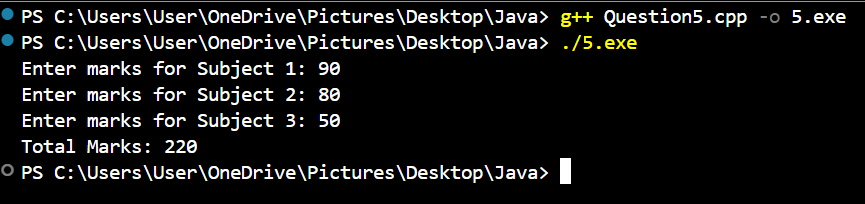
TotalMarks student;

student.inputMarks();

student.displayTotalMarks();

return 0;}

**Output**



**6. Program to print the reverse of the numbers; the numbers is taken as input from the user(Java).**

import java.util.Scanner;

public class Question6 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

int reverse = 0;

while (number != 0) {

int digit = number % 10;

reverse = reverse \* 10 + digit;

number /= 10;

}

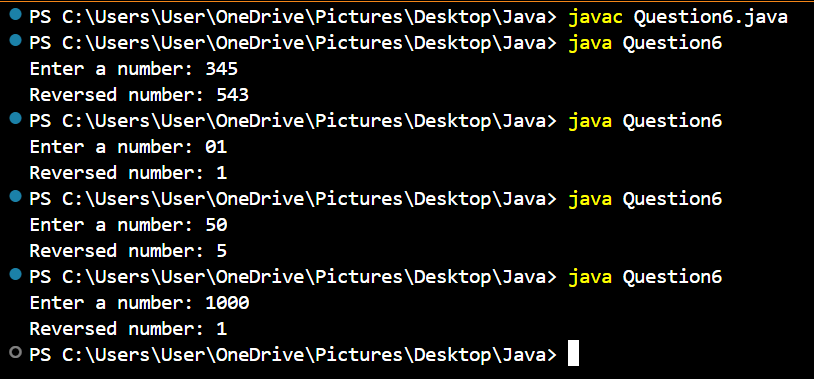
System.out.println("Reversed number: " + reverse);

scanner.close();

}

}

**Output**

****

**7. Program to maintain a Bank Account using packages (Java).**

class BankAccount{

protected String accountHolderName;

protected int accountNumber;

protected double balance;

public BankAccount(String accountHolderName,int accountNumber, double balance){

this.accountHolderName = accountHolderName;

this.accountNumber = accountNumber;

this.balance = balance;

}

public void deposit(double amount){

if(amount>0){

balance+=amount;

System.out.println("Rs"+amount+" Deposited.");

}else System.out.println("Invalid deposit amount.");

}

public void withdraw(double amount){

if(amount>0 && amount<=balance){

balance-=amount;

System.out.println("Rs"+amount+" Withdraw.");

}else System.out.println("Insufficient balance or invalid amount.");

}

public void showBalance(){

System.out.println("Account Holder: " + accountHolderName + ", Account Number: " + accountNumber + ", Current Balance: Rs" + balance);

}

}

class SavingAccount extends BankAccount{

final double interestRate = 6;

public SavingAccount( String accountHolderName,int accountNumber,double balance ){

super(accountHolderName,accountNumber,balance);

}

public void addInterest(){

double interest = balance\*interestRate/100;

balance+=interest;

System.out.println("Interest added: Rs"+ interest);

}

}

class CurrentAccount extends BankAccount{

final double miniBalanceRequired =1000.0;

public CurrentAccount(String accountHolderName,int accountNumber,double balance){

super(accountHolderName, accountNumber, balance);

}

@Override

public void withdraw(double amount){

if(amount>0 && (balance-amount)>=miniBalanceRequired){

balance-=amount;

System.out.println("Rs"+amount+" Withdraw.");

}

else System.out.println("Balance should be greater than Minimum Balance for withdraw.");

}

}

public class Question7{

public static void main(String[] args) {

// creating a saving account

SavingAccount savingAccount = new SavingAccount("Amit",112345,2000.0);

System.out.println("\n---Saving Account---");

savingAccount.showBalance();

savingAccount.deposit(1000.0);

savingAccount.addInterest();

savingAccount.showBalance();

//creating a current user

CurrentAccount currentAccount = new CurrentAccount("Ruka", 2116404524, 1800);

System.out.println("\n--- Current Account ---");

currentAccount.showBalance();

currentAccount.withdraw(1000);

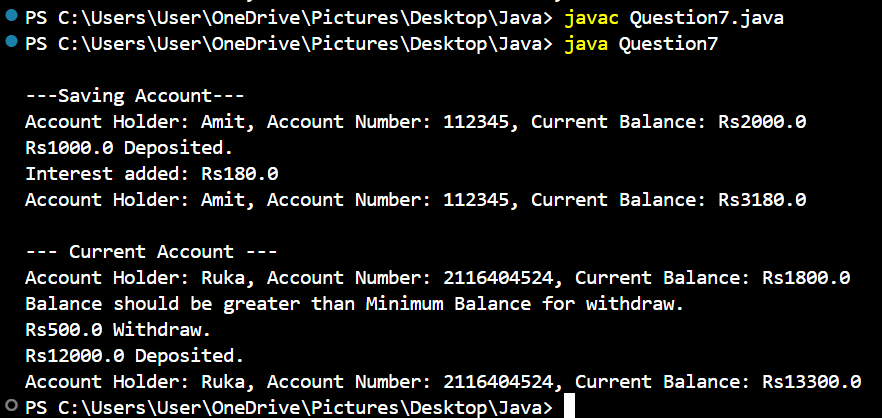
currentAccount.withdraw(500);

currentAccount.deposit(12000);

currentAccount.showBalance(); }

}

**Output**

****

**8. Program to maintain Bank Account using packages (Java).**

**9. Program to run the main thread and perform operations on it. Change the name and priority of the main thread (Java).**

public class Question9 {

public static void main(String[] args) {

Thread mainThread = Thread.currentThread();

System.out.println("Default Name of main thread: " + mainThread.getName());

System.out.println("Default Priority of main thread: " + mainThread.getPriority());

mainThread.setName("Task1");

mainThread.setPriority(Thread.MAX\_PRIORITY);

System.out.println("Updated Name of main thread: " + mainThread.getName());

System.out.println("Updated Priority of main thread: " + mainThread.getPriority());

System.out.println("Main thread is performing operation printing Number from 1 to 5");

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

System.out.println("Number " + i);

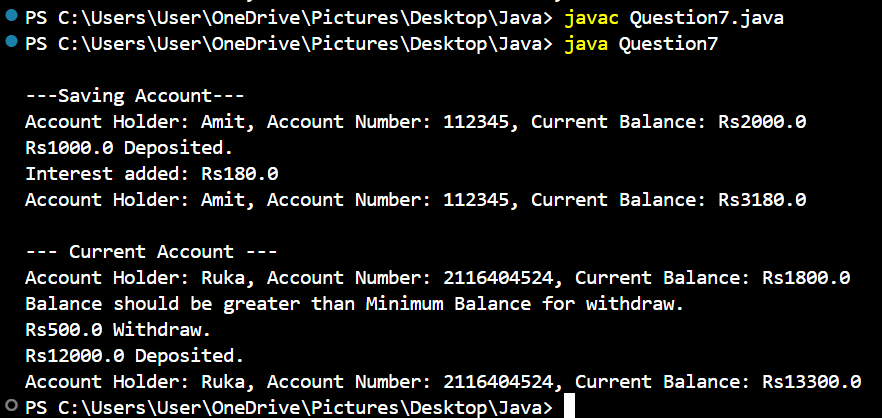
}

System.out.println("Main thread operations complete.");

}

}

**Output**



**10. Program to illustrate the working of child threads in concurrence with the main thread (Java)**

class ChildThread extends Thread {

public ChildThread(String name) {

super(name);

}

@Override

public void run() {

System.out.println(getName() + " is starting...");

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

System.out.println(getName() + " operation " + i);

try {

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

} catch (InterruptedException e) {

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

}

}

System.out.println(getName() + " is complete.");

}

}

public class Question10 {

public static void main(String[] args) {

System.out.println("Main thread is starting");

ChildThread childThread = new ChildThread("Child");

childThread.start();

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

System.out.println("Main thread operation " + i);

try {

Thread.sleep(700); // Pause for 700 milliseconds

} catch (InterruptedException e) {

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

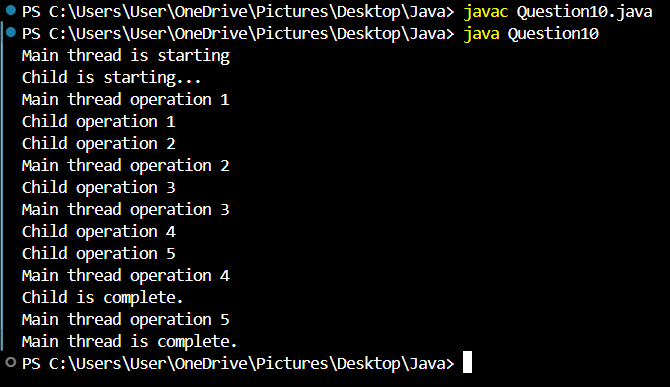
}

}

System.out.println("Main thread is complete.");

}}

**Output**

****

**11. Program to take a string array as “100”, “10.2”, “5.hello”, “100hello” and check whether it contains valid integer or double using exception handling (Java).**

public class Question11 {

public static void main(String[] args) {

String[] array = {"100", "10.2", "5.hello", "100hello"};

for (int i = 0; i < array.length; i++) {

String data = array[i]; // Get the current string

try {

int integervalue = Integer.parseInt(data);

System.out.println(data + " is a valid integer: " + integervalue);

} catch (NumberFormatException e) {

// If integer parsing fails, attempt to parse as a double

try {

double doubleValue = Double.parseDouble(data);

System.out.println(data + " is a valid double: " + doubleValue);

} catch (NumberFormatException ex) {

// If both parsing attempts fail

System.out.println(data + " is not a valid integer or double.");

}

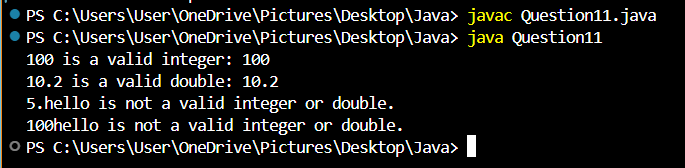
}

}

}

}

**Output**

****

**12. WAP to create a rectangle in an swing window and check if the mouse is inside or outside the rectangle and the swing window. (Java).**

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.\*;

public class Question12 {

static int rectX = 50;

static int rectY = 50;

static int rectWidth = 200;

static int rectHeight = 100;

static boolean isInsideRectangle = false;

public static void main(String[] args) {

JFrame frame = new JFrame("Rectangular Box");

frame.setSize(300, 300);

frame.setLayout(new FlowLayout());

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel drawingPanel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(Color.BLACK);

g.fillRect(rectX, rectY, rectWidth, rectHeight);

}

};

drawingPanel.setPreferredSize(new Dimension(300, 200));

drawingPanel.addMouseMotionListener(new MouseAdapter() {

@Override

public void mouseMoved(MouseEvent e) {

int mouseX = e.getX();

int mouseY = e.getY();

boolean currentlyInsideRectangle = (mouseX >= rectX && mouseX <= rectX + rectWidth &&

mouseY >= rectY && mouseY <= rectY + rectHeight);

if (currentlyInsideRectangle) {

if (!isInsideRectangle) {

System.out.println("Mouse is inside the rectangle.");

isInsideRectangle = true;

}

} else {

if (isInsideRectangle) {

System.out.println("Mouse is outside the rectangle but inside the Swing Window.");

isInsideRectangle = false;

}

}

}

});

drawingPanel.addMouseListener(new MouseAdapter() {

@Override

public void mouseEntered(MouseEvent e) {

System.out.println("Mouse is Inside Swing Window");

}

@Override

public void mouseExited(MouseEvent e) {

System.out.println("Mouse is Outside Swing Window");

}

});

frame.add(drawingPanel);

frame.setLocationRelativeTo(null);

frame.setVisible(true);

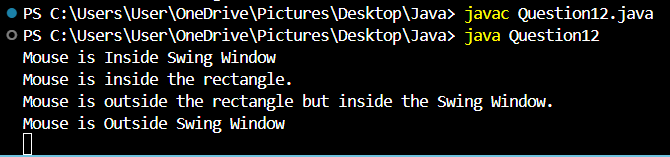
}

}

**Output**

| Fig 1 | Fig 2 |
| --- | --- |
| Fig 3 | Fig 4 |

**Different Mouse Position Output Shown below**

****

**13. WAP to create a standalone window and handle various mouse events. Also handle the closing of the frame(Java).**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Question13 {

public static void main(String[] args) {

JFrame frame = new JFrame("Question13");

frame.setSize(300, 300);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

JPanel panel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(Color.Gray);

g.fillRect(0, 0, getWidth(), getHeight());

}

};

panel.setPreferredSize(new Dimension(400, 200));

panel.addMouseListener(new MouseAdapter(){

@Override

public void mouseEntered(MouseEvent e) {

System.out.println("Mouse has entered the panel.");

}

@Override

public void mouseExited(MouseEvent e){

System.out.println("Mouse has Exited the panel."); }

@Override

public void mousePressed(MouseEvent e) {

System.out.println("Mouse pressed at: (" + e.getX() + ", " + e.getY() + ")");

}

@Override

public void mouseReleased(MouseEvent e) {

System.out.println("Mouse released at: (" + e.getX() + ", " + e.getY() + ")");

}

@Override

public void mouseClicked(MouseEvent e) {

System.out.println("Mouse clicked at: (" + e.getX() + ", " + e.getY() + ")");

}

});

panel.addMouseMotionListener(new MouseMotionAdapter(){

@Override

public void mouseMoved(MouseEvent e){

System.out.println("Mouse moved to: (" + e.getX() + ", " + e.getY() + ")");

}

@Override

public void mouseDragged(MouseEvent e) {

System.out.println("Mouse dragged to: (" + e.getX() + ", " + e.getY() + ")");

}

});

frame.addWindowListener(new WindowAdapter(){

@Override

public void windowClosing(WindowEvent e) {

int confirm = JOptionPane.showConfirmDialog(frame, "Are you sure you want to close?", "Confirm Exit",

JOptionPane.YES\_NO\_OPTION);

if (confirm == JOptionPane.YES\_OPTION) {

frame.dispose(); // Close the frame

}

}

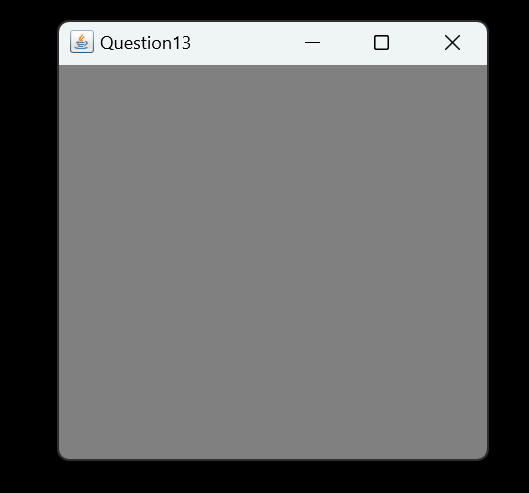
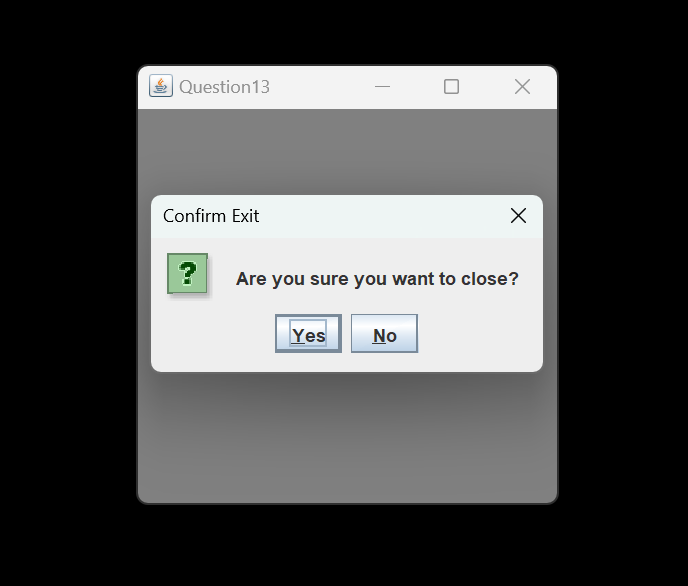
});

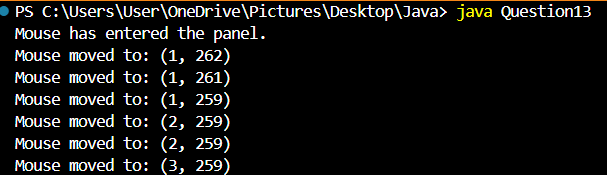
frame.add(panel);

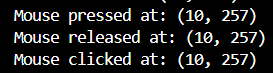
frame.setVisible(true);

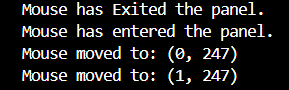
}}

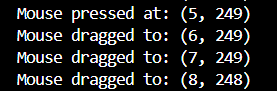
**Output**

****

****







**14. WAP to create a standalone window to handle ItemEvent corresponding to a choice component added to it using the concept of Anonymous Inner classes. Also add a button to open a child frame inside this frame(Java).**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Question14 {

public static void main(String[] args) {

JFrame mainFrame = new JFrame("Main Frame");

mainFrame.setSize(400, 300);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.setLayout(new FlowLayout());

JLabel label = new JLabel("Select an option:");

mainFrame.add(label);

Choice choice = new Choice();

choice.add("Option 1");

choice.add("Option 2");

choice.add("Option 3");

choice.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if (e.getStateChange() == ItemEvent.SELECTED) {

System.out.println("Selected: " + choice.getSelectedItem());

}

}

});

mainFrame.add(choice);

JPanel buttonPanel = new JPanel();

JButton openChildFrameButton = new JButton("Open Child Frame");

openChildFrameButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFrame childFrame = new JFrame("Child Frame");

childFrame.setSize(200, 150);

childFrame.setLocationRelativeTo(mainFrame);

childFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

childFrame.setLayout(new FlowLayout());

childFrame.add(new JLabel("This is the child frame!"));

childFrame.setVisible(true);

}

});

buttonPanel.add(openChildFrameButton);

mainFrame.add(buttonPanel); // Add the button panel to the main frame

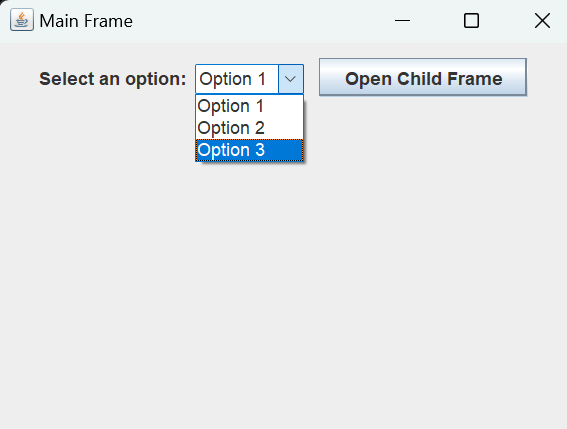
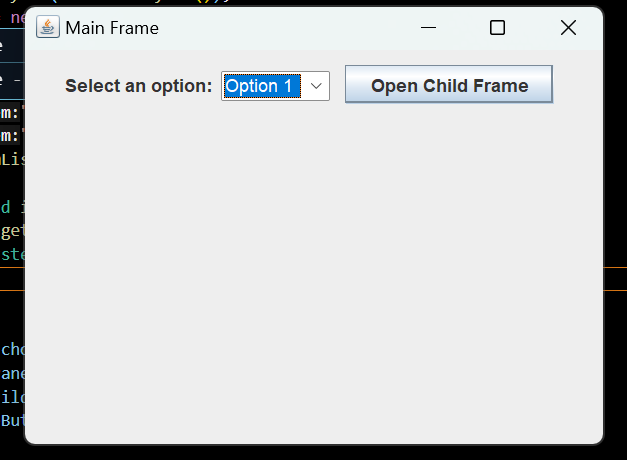
mainFrame.setLocationRelativeTo(null);

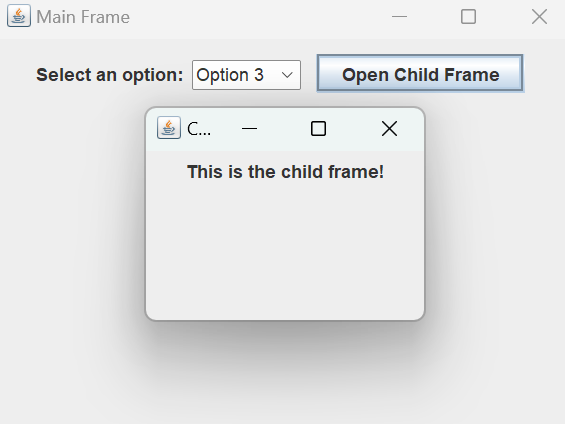
mainFrame.setVisible(true);

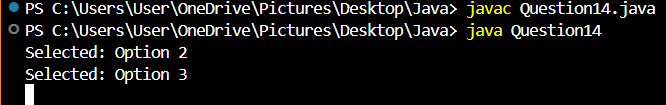
}

}

**Output**

****

****

****

**15. WAP to illustrate the concept of JDBC (Java).**