**Object Oriented Programming’s Practical File**

**(IT-667)**

***Submitted in partial fulfillment of the requirements for the award of the degree***

***Of***

**Masters of Computer Application (SE)**

**Submitted to: Submitted By:**

**Dr. Jaspreeti Singh Name: Nabin Kumar Pal**

**Assistant Professor Enrollment No: 07016404524**

**Semester - I**

**Section -2 ( Group A )**

****

**University School of Information , Communication & Technology**

**Guru Gobind Singh Indraprastha University**

**Sector-16C , Dwarka New Delhi - 110078**

**(2024-2026)**

**INDEX**

| **S No.** | **Practical Description** | **Pg No.** | **Sign** |
| --- | --- | --- | --- |
| 1 | WAP to implement Call by Value and Call by Reference in C++. Use Call by Reference to swap two integer values. (C++). | **4** |  |
| 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++). | **6** |  |
| 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. | **8** |  |
| 4 | Write a Program to implement Multilevel Inheritance using C++. | **10** |  |
| 5 | Write a Program to calculate the total mark of a student using the concept of virtual class(C++). | **12** |  |
| 6 | Program to print the reverse of the numbers; the numbers is taken as input from the user(Java). | **15** |  |
| 7 | Program to maintain bank account. Extend Bank account details to current and saving account (Java). | **16** |  |
| 8 | Program to maintain Bank Account using packages (Java). | **20** |  |
| 9 | Program to run the main thread and perform operations on it. Change the name and priority of the main thread (Java). | **24** |  |
| 10 | Program to illustrate the working of child threads in concurrence with the main thread (Java). | **26** |  |
| 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). | **28** |  |
| 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). | **29** |  |
| 13 | WAP to create a standalone window and handle various mouse events. Also handle the closing of the frame(Java). | **31** |  |
| 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). | **35** |  |
| 15 | WAP to illustrate the concept of JDBC (Java). | **38** |  |

**Practical No.1**

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

**Code:**

#include <iostream>

using namespace std;

void callByValue(int a) {

a = 20;

cout << "Inside callByValue function: a = " << a << endl;

}

void callByReference(int &a) {

a = 30;

cout << "Inside callByReference function: a = " << a << endl;

}

void swapByReference(int &x, int &y) {

int temp = x;

x = y;

y = temp;

}

int main() {

int num1 = 10;

int num2 = 40;

cout << "Before callByValue, num1 = " << num1 << endl;

callByValue(num1);

cout << "After callByValue, num1 = " << num1 << endl;

cout << "\nBefore callByReference, num2 = " << num2 << endl;

callByReference(num2);

cout << "After callByReference, num2 = " << num2 << endl;

int a = 5, b = 15;

cout << "\nBefore swap, a = " << a << ", b = " << b << endl;

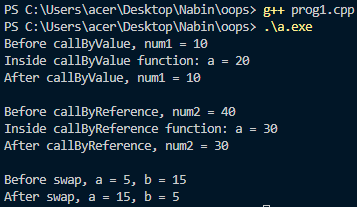
swapByReference(a, b);

cout << "After swap, a = " << a << ", b = " << b << endl;

return 0;

}

**Output:**



**Practical No.2**

**Aim:** 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++).

**Code:**

#include <iostream>

using namespace std;

double calculateSimpleInterest(double principal, double time, double rate = 5.0) {

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

}

int main() {

double principal, time, rate;

cout << "Enter principal amount: ";

cin >> principal;

cout << "Enter time in years: ";

cin >> time;

cout << "Enter rate of interest (or press 0 to use the default rate of 5%): ";

cin >> rate;

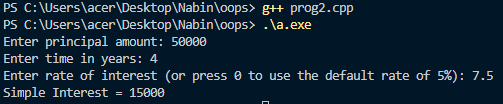
double interest = (rate == 0) ? calculateSimpleInterest(principal, time) : calculateSimpleInterest(principal, time, rate);

cout << "Simple Interest = " << interest << endl;

return 0;

}

**Output:**



**Practical No.3**

**Aim:** 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.

**Code:**

#include <iostream>

using namespace std;

class Example {

int x, y;

public:

Example() : x(19), y(35) {}

Example(int val) : x(val), y(0) {}

Example(int val1, int val2) : x(val1), y(val2) {}

Example(const Example &obj) : x(obj.x), y(obj.y) {}

void display() const {

cout << "x = " << x << ", y = " << y << endl;

}

};

int main() {

Example obj1;

obj1.display();

Example obj2(10);

obj2.display();

Example obj3(20, 30);

obj3.display();

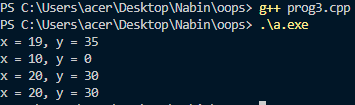
Example obj4 = obj3;

obj4.display();

return 0;

}

**Output:**



**Practical No.4**

**Aim:** Write a Program to implement Multilevel Inheritance using C++.

**Code:**

#include <iostream>

using namespace std;

class Animal {

public:

void eat() {

cout << "Animal is eating" << endl;

}

};

class Mammal : public Animal {

public:

void walk() {

cout << "Mammal is walking" << endl;

}

};

class Dog : public Mammal {

public:

void bark() {

cout << "Dog is barking" << endl;

}

};

int main() {

Dog myDog;

myDog.eat();

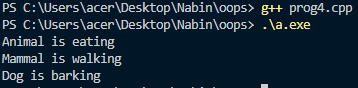
myDog.walk();

myDog.bark();

return 0;

}

**Output:**



**Practical No.5**

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

**Code:**

#include <iostream>

using namespace std;

class Student {

protected:

int studentID;

public:

void setStudentID(int id) {

studentID = id;

}

void displayStudentID() const {

cout << "Student ID: " << studentID << endl;

}

};

class Marks : virtual public Student {

protected:

int mark1, mark2;

public:

void setMarks(int m1, int m2) {

mark1 = m1;

mark2 = m2;

}

void displayMarks() const {

cout << "Marks: " << mark1 << ", " << mark2 << endl;

}

};

class Sports : virtual public Student {

protected:

int sportsMark;

public:

void setSportsMark(int smark) {

sportsMark = smark;

}

void displaySportsMark() const {

cout << "Sports Mark: " << sportsMark << endl;

}

};

class Result : public Marks, public Sports {

public:

void displayTotal() const {

int total = mark1 + mark2 + sportsMark;

displayStudentID();

displayMarks();

displaySportsMark();

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

}

};

int main() {

Result student;

student.setStudentID(101);

student.setMarks(85, 90);

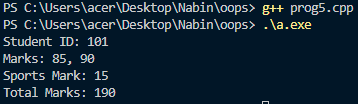
student.setSportsMark(15);

student.displayTotal();

return 0;

}

**Output:**



**Practical No.6**

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

**Code:**

import java.util.Scanner;

// ReverseNumber

public class prog6{

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;

}

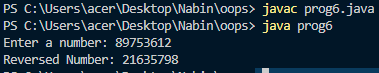
System.out.println("Reversed Number: " + reverse);

scanner.close();

}

}

**Output:**



**Practical No.7**

**Aim:** Program to maintain bank account. Extend Bank account details to current and saving account (Java).

**Code:**

import java.util.Scanner;

class BankAccount {

protected int accountNumber;

protected String accountHolderName;

protected double balance;

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

this.accountNumber = accountNumber;

this.accountHolderName = accountHolderName;

this.balance = balance;

}

public void deposit(double amount) {

balance += amount;

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

System.out.println("New Balance: " + balance);

}

public void withdraw(double amount) {

if (amount > balance) {

System.out.println("Insufficient balance.");

} else {

balance -= amount;

System.out.println("Withdrawn: " + amount);

System.out.println("Remaining Balance: " + balance);

}

}

public void displayAccountDetails() {

System.out.println("Account Number: " + accountNumber);

System.out.println("Account Holder Name: " + accountHolderName);

System.out.println("Balance: " + balance);

}

}

class SavingAccount extends BankAccount {

private double interestRate;

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

super(accountNumber, accountHolderName, balance);

this.interestRate = interestRate;

}

public void addInterest() {

double interest = balance \* interestRate / 100;

balance += interest;

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

System.out.println("New Balance after interest: " + balance);

}

}

class CurrentAccount extends BankAccount {

private double overdraftLimit;

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

super(accountNumber, accountHolderName, balance);

this.overdraftLimit = overdraftLimit;

}

@Override

public void withdraw(double amount) {

if (amount > balance + overdraftLimit) {

System.out.println("Overdraft limit exceeded.");

} else {

balance -= amount;

System.out.println("Withdrawn: " + amount);

System.out.println("Remaining Balance: " + balance);

}

}

}

// BankAccountDemo

public class prog7 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Creating Saving Account...");

SavingAccount savingAccount = new SavingAccount(1001, "Nabin", 50000.0, 7.5);

savingAccount.displayAccountDetails();

savingAccount.deposit(1000);

savingAccount.withdraw(2000);

savingAccount.addInterest();

savingAccount.displayAccountDetails();

System.out.println("\nCreating Current Account...");

CurrentAccount currentAccount = new CurrentAccount(2001, "Ashish", 30000.0, 1000.0);

currentAccount.displayAccountDetails();

currentAccount.deposit(500);

currentAccount.withdraw(4000);

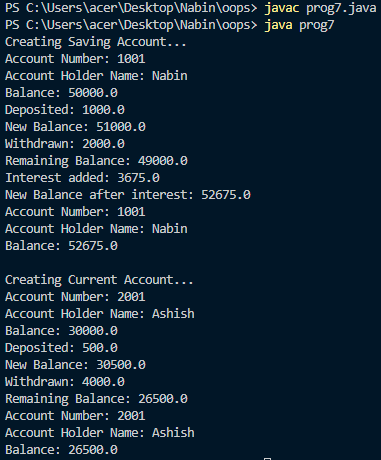
currentAccount.displayAccountDetails();

scanner.close();

}

}

**Output:**



**Practical No.8**

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

**Code:**

**In Bank folder**

**BankAccount.java**

package bank;

public class BankAccount {

protected int accountNumber;

protected String accountHolderName;

protected double balance;

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

this.accountNumber = accountNumber;

this.accountHolderName = accountHolderName;

this.balance = balance;

}

public void deposit(double amount) {

balance += amount;

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

System.out.println("New Balance: " + balance);

}

public void withdraw(double amount) {

if (amount > balance) {

System.out.println("Insufficient balance.");

} else {

balance -= amount;

System.out.println("Withdrawn: " + amount);

System.out.println("Remaining Balance: " + balance);

}

}

public void displayAccountDetails() {

System.out.println("Account Number: " + accountNumber);

System.out.println("Account Holder Name: " + accountHolderName);

System.out.println("Balance: " + balance);

}

}

**CurrentAccount.java**

package bank;

public class CurrentAccount extends BankAccount {

private double overdraftLimit;

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

super(accountNumber, accountHolderName, balance);

this.overdraftLimit = overdraftLimit;

}

@Override

public void withdraw(double amount) {

if (amount > balance + overdraftLimit) {

System.out.println("Overdraft limit exceeded.");

} else {

balance -= amount;

System.out.println("Withdrawn: " + amount);

System.out.println("Remaining Balance: " + balance);

}

}

}

**SavingAccount.java**

package bank;

public class SavingAccount extends BankAccount {

private double interestRate;

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

super(accountNumber, accountHolderName, balance);

this.interestRate = interestRate;

}

public void addInterest() {

double interest = balance \* interestRate / 100;

balance += interest;

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

System.out.println("New Balance after interest: " + balance);

}

}

**In Main Folder**

**BankAccountDemo.java**

package main;

import bank.BankAccount;

import bank.SavingAccount;

import bank.CurrentAccount;

public class BankAccountDemo {

public static void main(String[] args) {

System.out.println("Creating Saving Account...");

SavingAccount savingAccount = new SavingAccount(1001, "Nabin", 50000.0, 7.5);

savingAccount.displayAccountDetails();

savingAccount.deposit(1000);

savingAccount.withdraw(2000);

savingAccount.addInterest();

savingAccount.displayAccountDetails();

System.out.println("\nCreating Current Account...");

CurrentAccount currentAccount = new CurrentAccount(2001, "Ashish", 31000.0, 1000.0);

currentAccount.displayAccountDetails();

currentAccount.deposit(500);

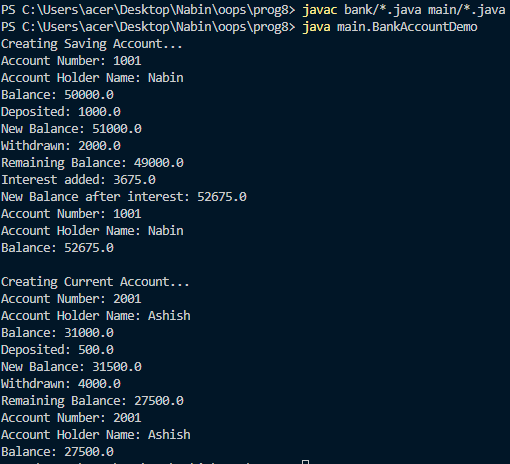
currentAccount.withdraw(4000);

currentAccount.displayAccountDetails();

}

}

**Output:**



**Practical No.9**

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

**Code:**

// MainThreadOperations

public class prog9 {

public static void main(String[] args) {

Thread mainThread = Thread.currentThread();

mainThread.setName("My Main Thread");

mainThread.setPriority(Thread.MAX\_PRIORITY);

System.out.println("Thread Name: " + mainThread.getName());

System.out.println("Thread Priority: " + mainThread.getPriority());

for (int i = 0; i < 20; i++) {

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

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

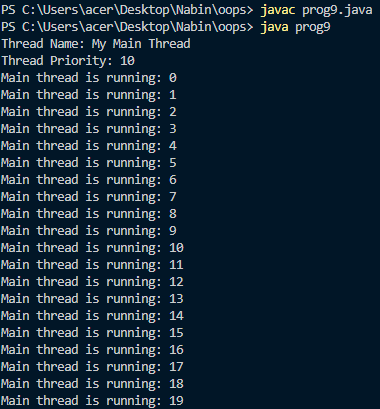
}

}

}

}

**Output:**



**Practical No.10**

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

**Code:**

// ChildThreadExample

public class prog10 {

public static void main(String[] args) {

Thread childThread = new Thread(new Runnable() {

@Override

public void run() {

for (int i = 0; i < 5; i++) {

System.out.println("Child Thread: " + i);

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

});

childThread.start();

for (int i = 0; i < 10; i++) {

System.out.println("Main Thread: " + i);

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

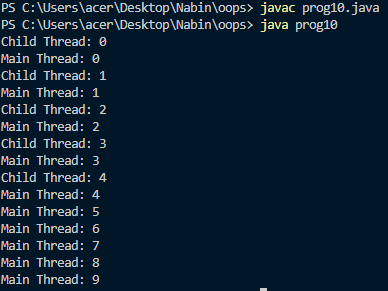
}

}

}

}

**Output:**



**Practical No.11**

**Aim:** 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).

**Code:**

import java.util.Arrays;

// StringArrayValidation

public class prog11 {

public static void main(String[] args) {

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

for (String str : inputArray) {

try {

double number = Double.parseDouble(str);

System.out.println(str + " is a valid number.");

} catch (NumberFormatException e) {

System.out.println(str + " is not a valid number.");

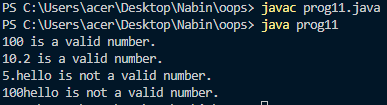
}

}

}

}

**Output:**



**Practical No.12**

**Aim:** 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).

**Code:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

// RectangleCheck

public class prog12 extends JFrame implements MouseMotionListener {

private Rectangle rectangle;

public prog12() {

setTitle("Rectangle Check");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

addMouseMotionListener(this);

rectangle = new Rectangle(100, 100, 150, 100);

}

@Override

public void paint(Graphics g) {

super.paint(g);

g.setColor(Color.RED);

g.drawRect(rectangle.x, rectangle.y, rectangle.width, rectangle.height);

}

@Override

public void mouseMoved(MouseEvent e) {

int x = e.getX();

int y = e.getY();

if (rectangle.contains(x, y)) {

setTitle("Mouse is inside the rectangle");

} else if (getBounds().contains(x, y)) {

setTitle("Mouse is inside the window, but outside the rectangle");

} else {

setTitle("Mouse is outside the window");

}

}

@Override

public void mouseDragged(MouseEvent e) {

public static void main(String[] args) {

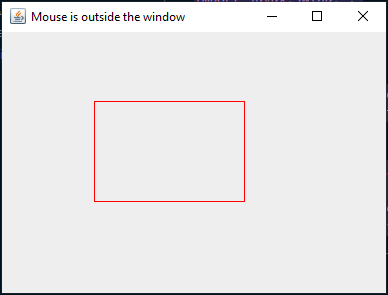
prog12 frame = new prog12();

frame.setVisible(true);

}

}

**Output:**



**Practical No.13**

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

**Code:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

// MouseEventDemo {

public class prog13 extends JFrame implements MouseListener, WindowListener {

public prog13() {

setTitle("Mouse Event Demo");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

addWindowListener(this);

addMouseListener(this);

}

@Override

public void paint(Graphics g) {

super.paint(g);

g.drawString("Click and move your mouse here!", 100, 100);

}

@Override

public void mouseClicked(MouseEvent e) {

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

}

@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 mouseEntered(MouseEvent e) {

System.out.println("Mouse entered the window.");

}

@Override

public void mouseExited(MouseEvent e) {

System.out.println("Mouse exited the window.");

}

@Override

public void windowOpened(WindowEvent e) {

System.out.println("Window opened.");

}

@Override

public void windowClosing(WindowEvent e) {

System.out.println("Window closing.");

System.exit(0); // Exit the application

}

@Override

public void windowClosed(WindowEvent e) {

System.out.println("Window closed.");

}

@Override

public void windowIconified(WindowEvent e) {

System.out.println("Window iconified.");

}

@Override

public void windowDeiconified(WindowEvent e) {

System.out.println("Window deiconified.");

}

@Override

public void windowActivated(WindowEvent e) {

System.out.println("Window activated.");

}

@Override

public void windowDeactivated(WindowEvent e) {

System.out.println("Window deactivated.");

}

public static void main(String[] args) {

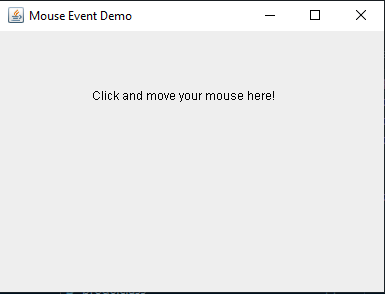
prog13 frame = new prog13();

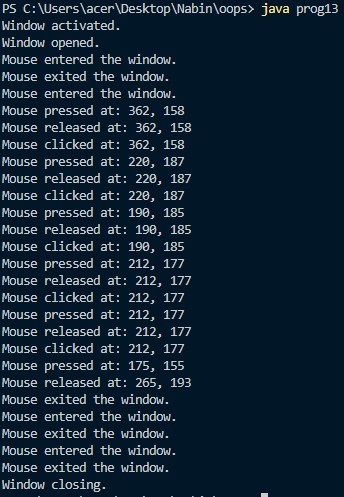
frame.setVisible(true);

}

}

**Output:**





**Practical No.14**

**Aim:** 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).

**Code:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

// ItemEventDemo

public class prog14 extends JFrame {

private JComboBox<String> comboBox;

private JButton openChildButton;

public prog14() {

setTitle("Item Event Demo");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new FlowLayout());

comboBox = new JComboBox<>(new String[]{"Option 1", "Option 2", "Option 3"});

comboBox.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

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

String selectedItem = (String) e.getItem();

System.out.println("Selected item: " + selectedItem);

}

}

});

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

openChildButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ChildFrame childFrame = new ChildFrame();

childFrame.setVisible(true);

}

});

add(comboBox);

add(openChildButton);

}

public static void main(String[] args) {

prog14 frame = new prog14();

frame.setVisible(true);

}

class ChildFrame extends JFrame {

public ChildFrame() {

setTitle("Child Frame");

setSize(300, 200);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

setLayout(new FlowLayout());

JLabel label = new JLabel("This is a child frame.");

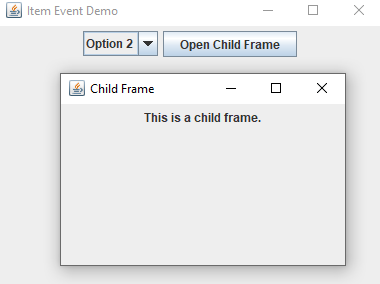
add(label);

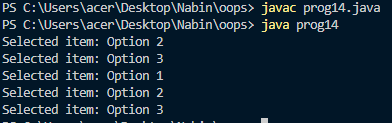
}

}

}

**Output:**





**Practical No.15**

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

**Code:**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class JDBCExample {

static final String DB\_URL = "jdbc:mysql://localhost:3306/testdb";

static final String USER = "root";

static final String PASS = "nabin";

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

Class.forName("com.mysql.cj.jdbc.Driver");

conn = DriverManager.getConnection(DB\_URL, USER, PASS);

stmt = conn.createStatement();

String sql = "CREATE TABLE IF NOT EXISTS users (" +

"id INT PRIMARY KEY AUTO\_INCREMENT, " +

"name VARCHAR(50), " +

"email VARCHAR(50))";

stmt.executeUpdate(sql);

System.out.println("Table created successfully...");

sql = "INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com')";

stmt.executeUpdate(sql);

System.out.println("Inserted record...");

sql = "SELECT \* FROM users";

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

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

String email = rs.getString("email");

System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email);

}

rs.close();

stmt.close();

conn.close();

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

**Output:**

