**Object Oriented Programming 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 : ThakurRudraksha Singh**

**Assistant Professor Enrollment No : 2016404524**

**Semester - I**

**Section -1 ( Group A )**

****

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

**Guru Gobind Singh Indraprastha University**

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

**(2024-2026)**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No.** | **TOPIC** | **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++). | 3 |  |
| 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++). | 5 |  |
| 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. | 7 |  |
| 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 are taken as input from the user(Java). | 14 |  |
| 7 | Program to maintain a bank account. Extend Bank account details to current and saving accounts (Java). | 16 |  |
| 8 | Program to maintain a 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 a 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). | 30 |  |
| 13 | WAP to create a standalone window and handle various mouse events. Also handle the closing of the frame(Java). | 34 |  |
| 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). |  |  |
| 15 | WAP to illustrate the concept of JDBC (Java). |  |  |

**Program-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 x, int y) {

    int temp = x;

    x = y;

    y = temp;

    cout << "After swapping in callByValue: x = " << x << ", y = " << y << endl;

}

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

    int temp = x;

    x = y;

    y = temp;

    cout << "After swapping in callByReference: x = " << x << ", y = " << y << endl;

}

int main() {

    int a, b;

    cout << "Enter value for a: ";

    cin >> a;

    cout << "Enter value for b: ";

    cin >> b;

    cout << "CALL BY VALUE:" << endl;

    cout << "Original values: a = " << a << ", b = " << b << endl;

    callByValue(a, b);

    cout << "Values outside scope of function in callByValue: a = " << a << ", b = " << b << endl;

    cout << "\nCALL BY REFERENCE:" << endl;

    cout << "Original values: a = " << a << ", b = " << b << endl;

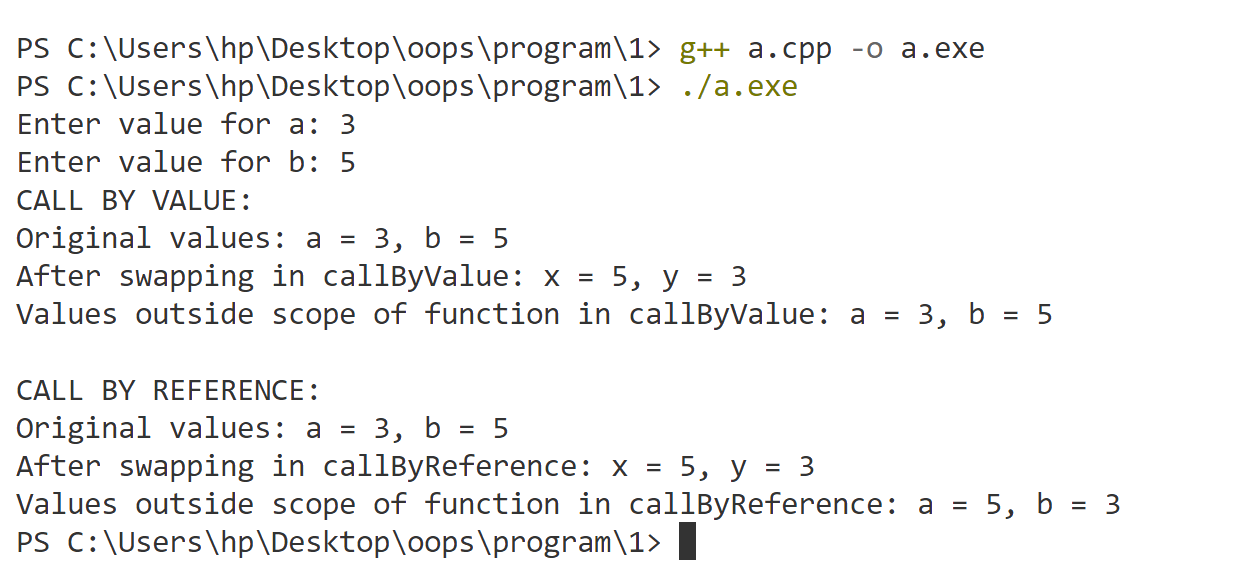
    callByReference(a, b);

    cout << "Values outside scope of function in callByReference: a = " << a << ", b = " << b << endl;

    return 0;

}

**Output**



**Program-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) {

    return (principal \* rate \* time) / 100.0;

}

int main() {

    double principal, time, rate;

    cout << "Enter the principal amount: ";

    cin >> principal;

    cout << "Enter the time (in years): ";

    cin >> time;

    char choice;

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

    cin >> choice;

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

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

        cin >> rate;

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

        cout << "The Simple Interest with rate of "<<rate<<"% is: " << simpleInterest << endl;

    } else {

        double simpleInterest = calculateSimpleInterest(principal, time);

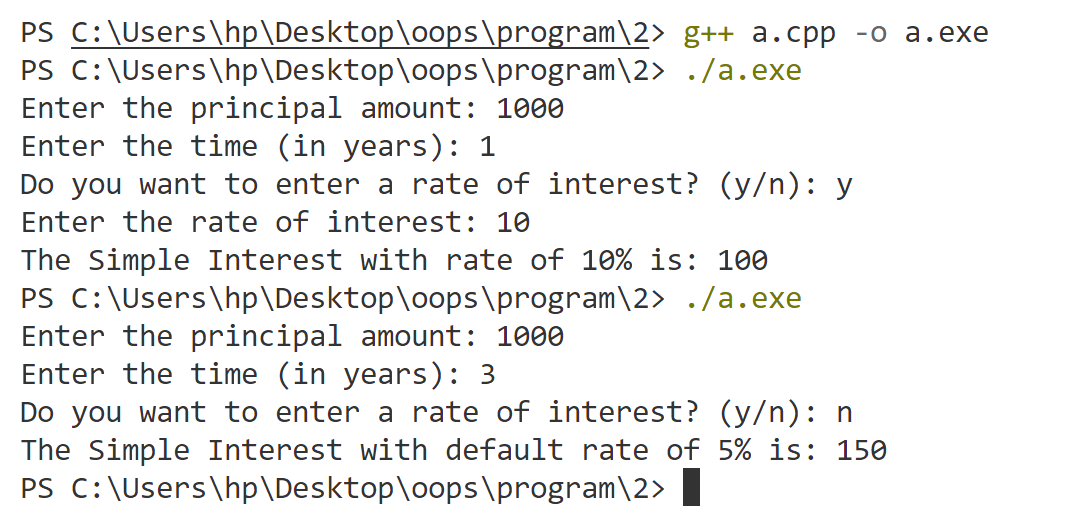
        cout << "The Simple Interest with default rate of 5% is: " << simpleInterest << endl;

    }

    return 0;

}

**Output**



**Program-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 Triangle {

private:

    double base;

    double height;

public:

    Triangle() {

        base = 1.0;

        height = 1.0;

        cout << "No-argument constructor called. Default Base: " << base << ", Default Height: " << height << endl;

    }

    Triangle(double side) {

        base = side;

        height = side;

        cout << "One-argument constructor called. Base: " << base << ", Height: " << height << endl;

    }

    Triangle(double b, double h) {

        base = b;

        height = h;

        cout << "Two-argument constructor called. Base: " << base << ", Height: " << height << endl;

    }

    Triangle(const Triangle &tri) {

        base = tri.base;

        height = tri.height;

        cout << "Copy constructor called. Base: " << base << ", Height: " << height << endl;

    }

    double area() {

        return 0.5 \* base \* height;

    }

};

int main() {

    Triangle tri1;// No-argument constructor

    Triangle tri2(5.0);// One-argument constructor

    Triangle tri3(4.0, 6.0);// Two-argument constructor

    Triangle tri4(tri3);// Copy constructor

    cout << "Area of tri1: " << tri1.area() << endl;

    cout << "Area of tri2: " << tri2.area() << endl;

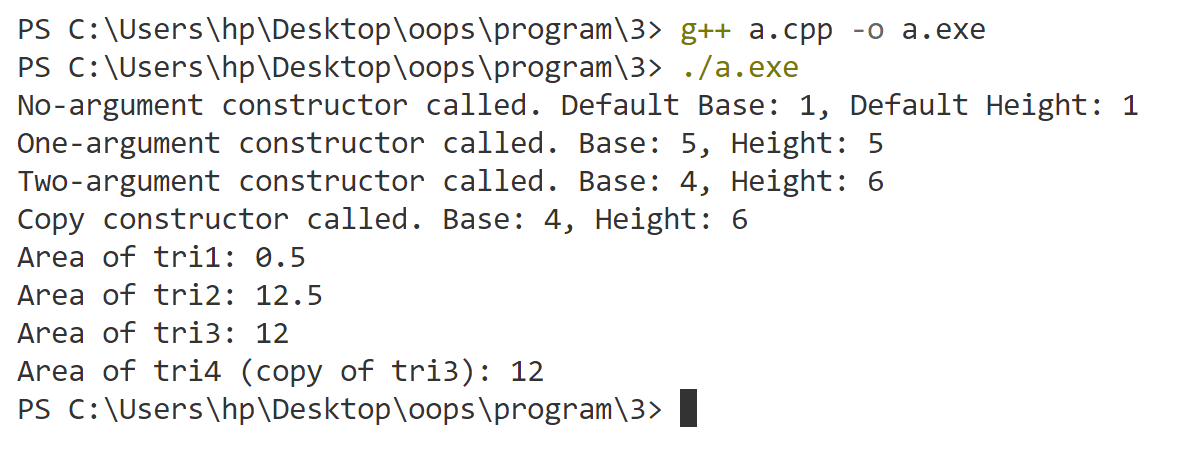
    cout << "Area of tri3: " << tri3.area() << endl;

    cout << "Area of tri4 (copy of tri3): " << tri4.area() << endl;

    return 0;

}

**Output**



**Program-4**

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

#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 dogObj;

    dogObj.eat();  // Inherited from Animal

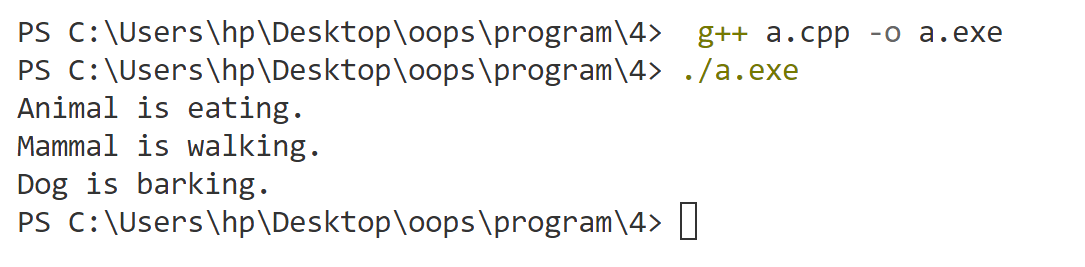
    dogObj.walk(); // Inherited from Mammal

    dogObj.bark(); // Defined in Dog

    return 0;

}

**Output**



**Program-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(){

        cout << "Input is not provided yet." << endl;

    };

    virtual void displayTotalMarks(){

        cout << "Marks is not given yet." << endl;

    };

};

class TotalMarks : public Student {

protected:

    int marks1, marks2, marks3;

public:

    void inputMarks()  {

        marks1 = 85;

        marks2 = 90;

        marks3 = 78;

        cout << "Marks have been set automatically." << endl;

    }

    void displayTotalMarks()  {

        int total = marks1 + marks2 + marks3;

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

    }

};

int main() {

    Student\* student;   //  or // Student\* student = new TotalMarks();

    TotalMarks t;

    student = &t;

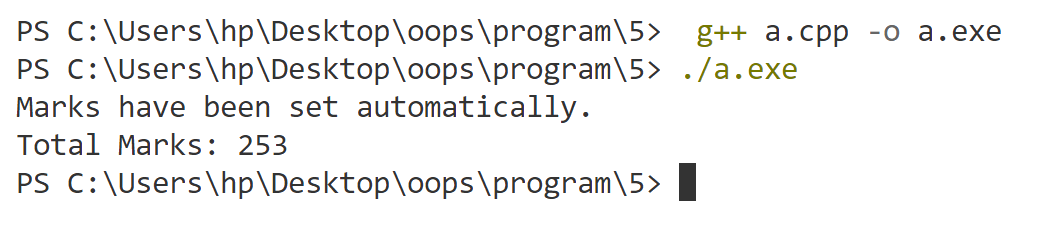
    student->inputMarks();

    student->displayTotalMarks();

    return 0;

}

**Output**

****

**Program-6**

**Program to print the reverse of the numbers. The numbers are taken as input from the user(Java).**

import java.util.Scanner;

public class ReverseNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int number = sc.nextInt();

        int reversedNumber = 0;

        while (number != 0) {

            int digit = number % 10;

            reversedNumber = reversedNumber \* 10 + digit;

            number = number / 10;

        }

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

        sc.close();

    }

}

**Output**



**Program-7**

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

abstract 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 abstract void displayAccountType();

    public void depositMoney(double amount) {

        if (amount > 0) {

            balance += amount;

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

        } else

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

    }

    public void withdrawMoney(double amount) {

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

            balance -= amount;

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

        } else

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

    }

    public void displayAccountInfo() {

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

    }

    public void getBalance() {

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

    }

}

class SavingAccount extends BankAccount {

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

        super(accountHolderName, accountNumber, balance);

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account type:Saving Account");

    }

    public void addInterest() {

        double interest = balance \* 5 / 100;

        balance += interest;

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

    }

}

class CurrentAccount extends BankAccount {

    private double overdraftLimit = 1000.00;

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

        super(accountHolderName, accountNumber, balance); // invoke the constructor of the superclass (BankAccount).

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account Type: Current Account");

    }

    @Override

    public void withdrawMoney(double amount) {

        if (amount > 0 && (balance + overdraftLimit) >= amount) {

            balance -= amount;

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

        } else {

            System.out.println("Overdraft limit exceeded or invalid amount.");

        }

    }

}

public class BankSystem {

    public static void main(String[] args) {

        SavingAccount savingAccount = new SavingAccount("rudra", 3263634, 3000.0);

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

        savingAccount.displayAccountInfo();

        savingAccount.displayAccountType();

        savingAccount.getBalance();

        savingAccount.depositMoney(1000.0);

        savingAccount.addInterest();

        savingAccount.getBalance();

        savingAccount.withdrawMoney(2000.56);

        savingAccount.getBalance();

        savingAccount.withdrawMoney(3000);

        CurrentAccount currentAccount = new CurrentAccount("pretti", 32573543, 2800);

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

        currentAccount.displayAccountInfo();

        currentAccount.displayAccountType();

        currentAccount.depositMoney(200);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(3100);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(900);

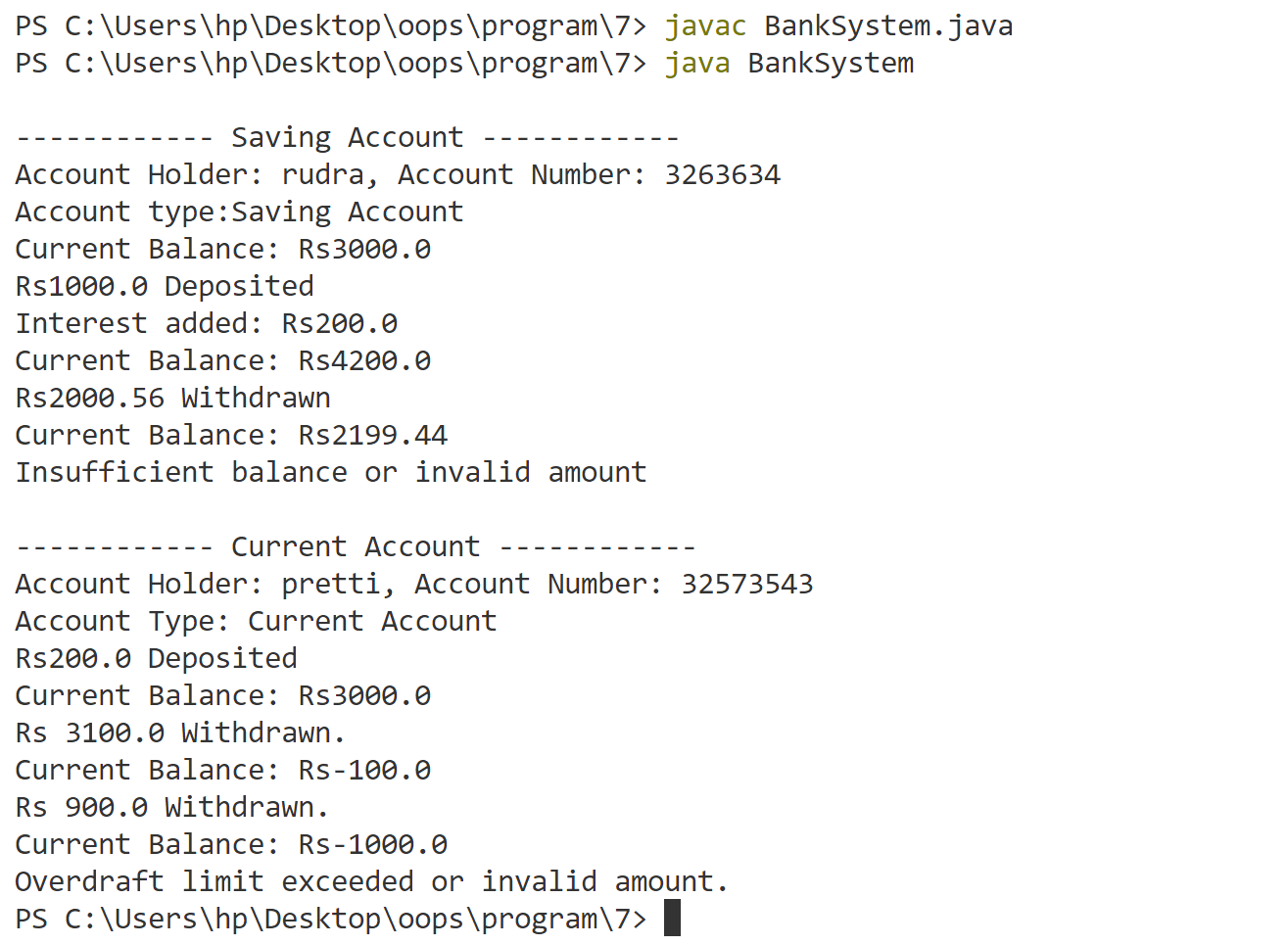
        currentAccount.getBalance();

        currentAccount.withdrawMoney(1);

    }

}

**Output**



**Program-8**

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

Structure in which files are stored



**BankSystem.java**

import bank.savings.SavingAccount;

import bank.current.CurrentAccount;

public class BankSystem {

    public static void main(String[] args) {

        SavingAccount savingAccount = new SavingAccount("rudra", 3263634, 3000.0);

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

        savingAccount.displayAccountInfo();

        savingAccount.displayAccountType();

        savingAccount.getBalance();

        savingAccount.depositMoney(1000.0);

        savingAccount.addInterest();

        savingAccount.getBalance();

        savingAccount.withdrawMoney(2000.56);

        savingAccount.getBalance();

        savingAccount.withdrawMoney(3000);

        CurrentAccount currentAccount = new CurrentAccount("pretti", 32573543, 2800);

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

        currentAccount.displayAccountInfo();

        currentAccount.displayAccountType();

        currentAccount.depositMoney(200);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(3100);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(900);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(1);

    }

}

**BankAccount.java**

package bank;

public abstract 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 abstract void displayAccountType();

    public void depositMoney(double amount) {

        if (amount > 0) {

            balance += amount;

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

        } else

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

    }

    public void withdrawMoney(double amount) {

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

            balance -= amount;

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

        } else

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

    }

    public void displayAccountInfo() {

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

    }

    public void getBalance() {

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

    }

}

**SavingAccount.java**

package bank.savings;

import bank.BankAccount;

public class SavingAccount extends BankAccount {

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

        super(accountHolderName, accountNumber, balance);

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account type:Saving Account");

    }

    public void addInterest() {

        double interest = balance \* 5 / 100;

        balance += interest;

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

    }

}

**CurrentAccount.java**

package bank.current;

import bank.BankAccount;

public class CurrentAccount extends BankAccount {

    private double overdraftLimit = 1000.00;

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

        super(accountHolderName, accountNumber, balance); //constructor of superclass (BankAccount)

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account Type: Current Account");

    }

    @Override

    public void withdrawMoney(double amount) {

        if (amount > 0 && (balance + overdraftLimit) >= amount) {

            balance -= amount;

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

        } else {

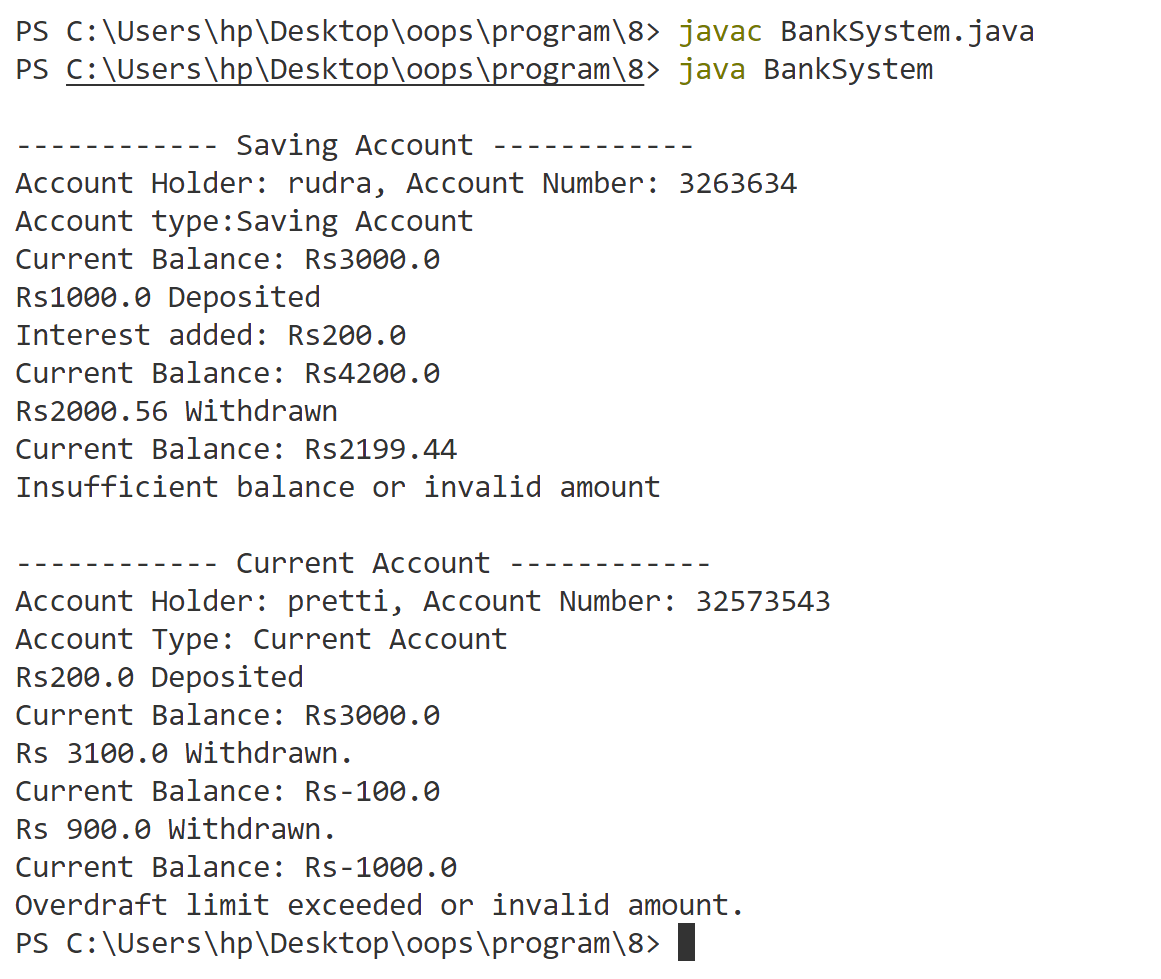
            System.out.println("Overdraft limit exceeded or invalid amount.");

        }

    }

}

**Output**

****

**Program-9**

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

public class MainThreadExample {

    public static void main(String[] args) {

        Thread mainThread = Thread.currentThread();

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

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

        System.out.println();

        mainThread.setName("CustomMainThread");

        mainThread.setPriority(Thread.MAX\_PRIORITY); // Settin priority 10

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

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

        System.out.println();

        System.out.println("Performing operations within main thread");

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

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

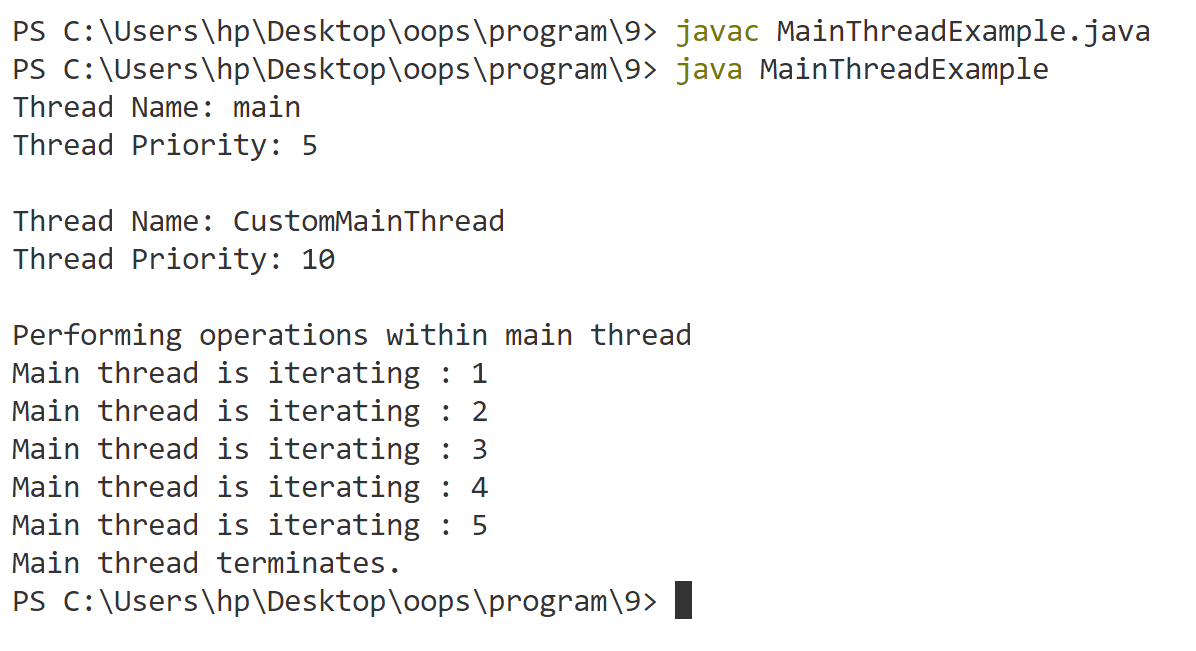
        }

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

    }

}

**Output**

****

**Program-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); // Call the parent (Thread) class constructor to set thread name

    }

    @Override

    public void run() {

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

        for (int i = 1; i <= 3; i++) {

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

            try {

                Thread.sleep(500);

            } 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 firstChildThread = new ChildThread("Child\_1");

        ChildThread secondChildThread = new ChildThread("Child\_2");

        firstChildThread.start();

        secondChildThread.start();

        for(int i=1;i<=3;i++){

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

            try {

                Thread.sleep(1000);

            } catch (InterruptedException e) {

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

            }

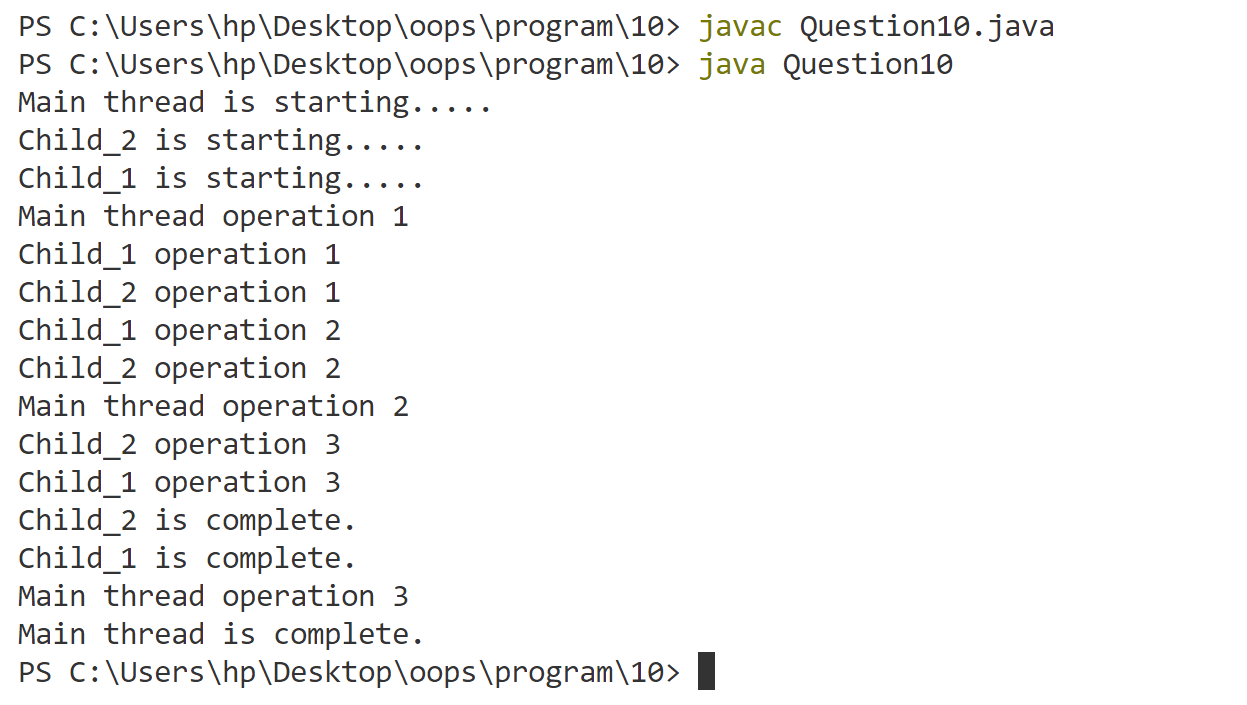
        }

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

    }

}

**Output**

****

**Program-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 ValidateNumbers {

    public static void main(String[] args) {

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

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

            String value = values[i];

            try {

                Integer.parseInt(value);

                System.out.println(value + " is a valid integer.");

            } catch (NumberFormatException eint) {

                try {

                    Double.parseDouble(value);

                    System.out.println(value + " is a valid double.");

                } catch (NumberFormatException edouble) {

                    // If both parsing attempts fail, it's neither an integer nor a double

                    System.out.println(value + " is neither a valid integer nor a valid double.");

                }

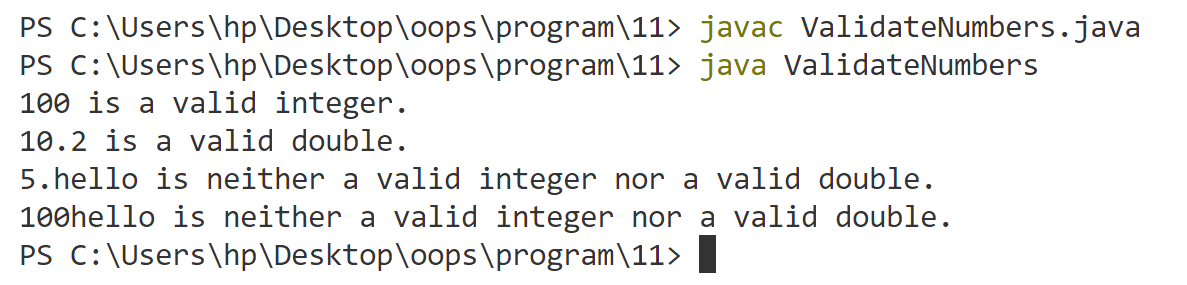
            }

        }

    }

}

**Output**

****

**Program-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 javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class RectangleMouseCheck extends JFrame {

    static int RECT\_X = 100;

    static int RECT\_Y = 100;

    static int RECT\_WIDTH =200;

    static int RECT\_HEIGHT = 100;

    static boolean isInsideRectangle = false;

    public RectangleMouseCheck() {//constructor

        setTitle("Rectangle Mouse Check");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        add(new panel());

        setVisible(true);

    }

    class panel extends JPanel {

        public panel() {

            setPreferredSize(new Dimension(400, 300));

            addMouseMotionListener(new MouseMotionAdapter() {

                @Override

                public void mouseMoved(MouseEvent e) {

                    int mouseX = e.getX();

                    int mouseY = e.getY();

                    boolean currentlyInsideRectangle = (mouseX >= RECT\_X && mouseX <= RECT\_X

                        +RECT\_WIDTH &&mouseY>=RECT\_Y&&mouseY <=RECT\_Y+RECT\_HEIGHT);

                        if (currentlyInsideRectangle) {

                            if (!isInsideRectangle) {

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

                                isInsideRectangle = true;

                            }

                        } else {

                            if (isInsideRectangle) {

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

                                isInsideRectangle = false;

                            }

                        }

                }

            });

            addMouseListener(new MouseAdapter() {

                @Override

                public void mouseEntered(MouseEvent e) {

                    System.out.println("Mouse entered Swing Window ");

                }

                @Override

                public void mouseExited(MouseEvent e) {

                    System.out.println("Mouse exited Swing Window ");

                }

            });

        }

        //paintComponent is a specialized method intended for custom painting within Swing

        @Override

        protected void paintComponent(Graphics g) {

            super.paintComponent(g); // Ensure the background is cleared before drawing

            setBackground(Color.LIGHT\_GRAY);

            g.setColor(Color.BLUE);

            g.fillRect(RECT\_X, RECT\_Y, RECT\_WIDTH, RECT\_HEIGHT);

        }

    }

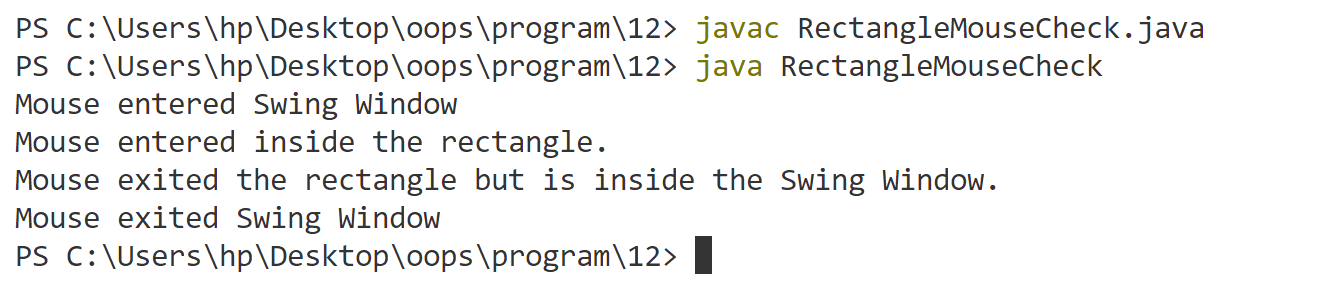
    public static void main(String[] args) {

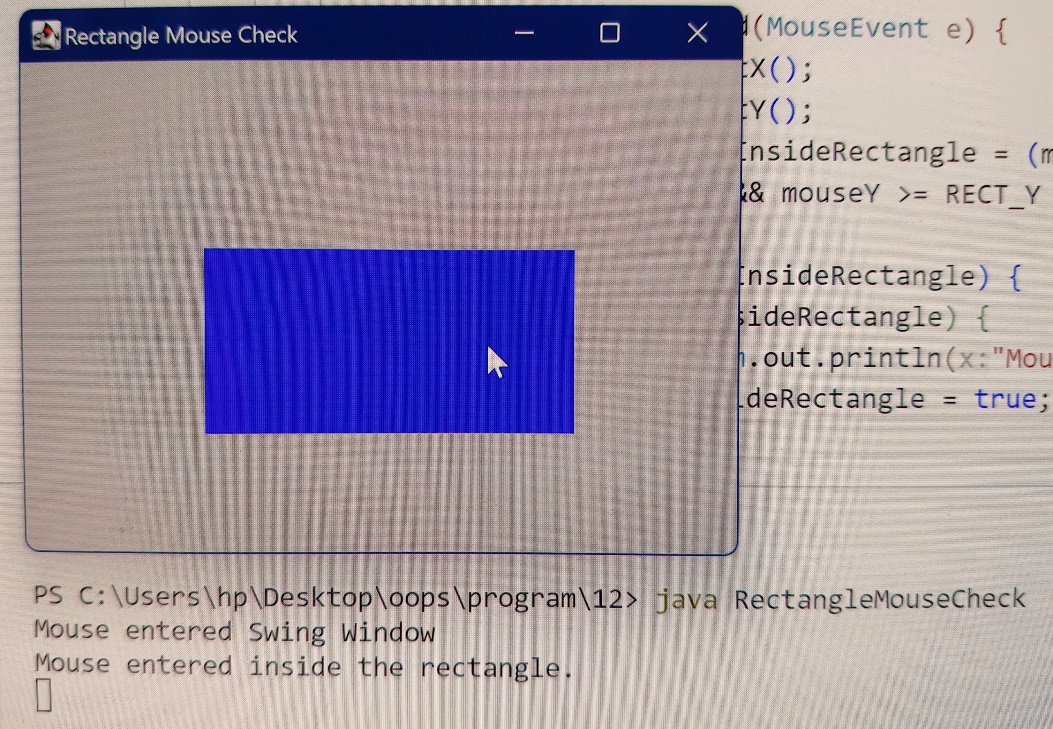
        new RectangleMouseCheck();

    }

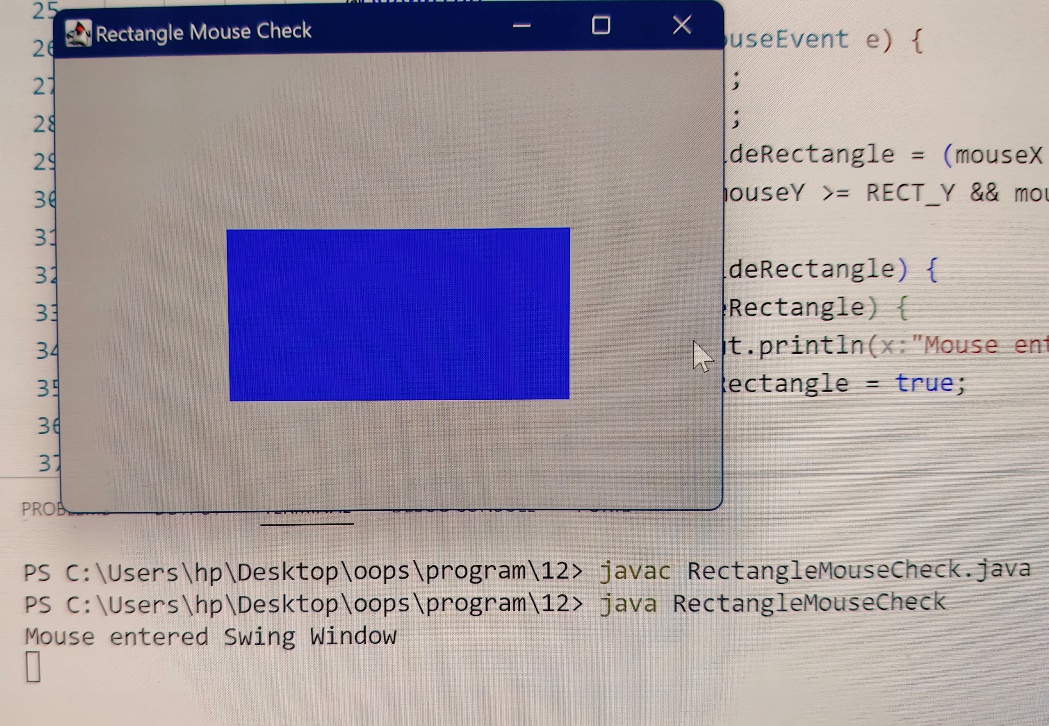
}

**Output**

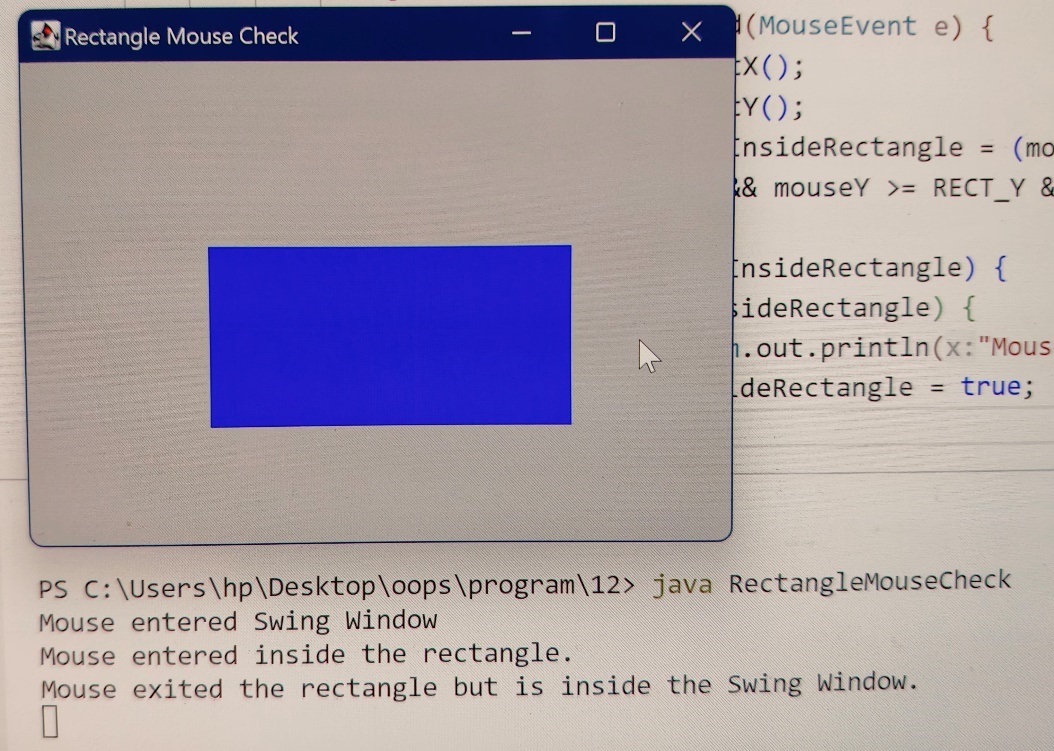
****

****

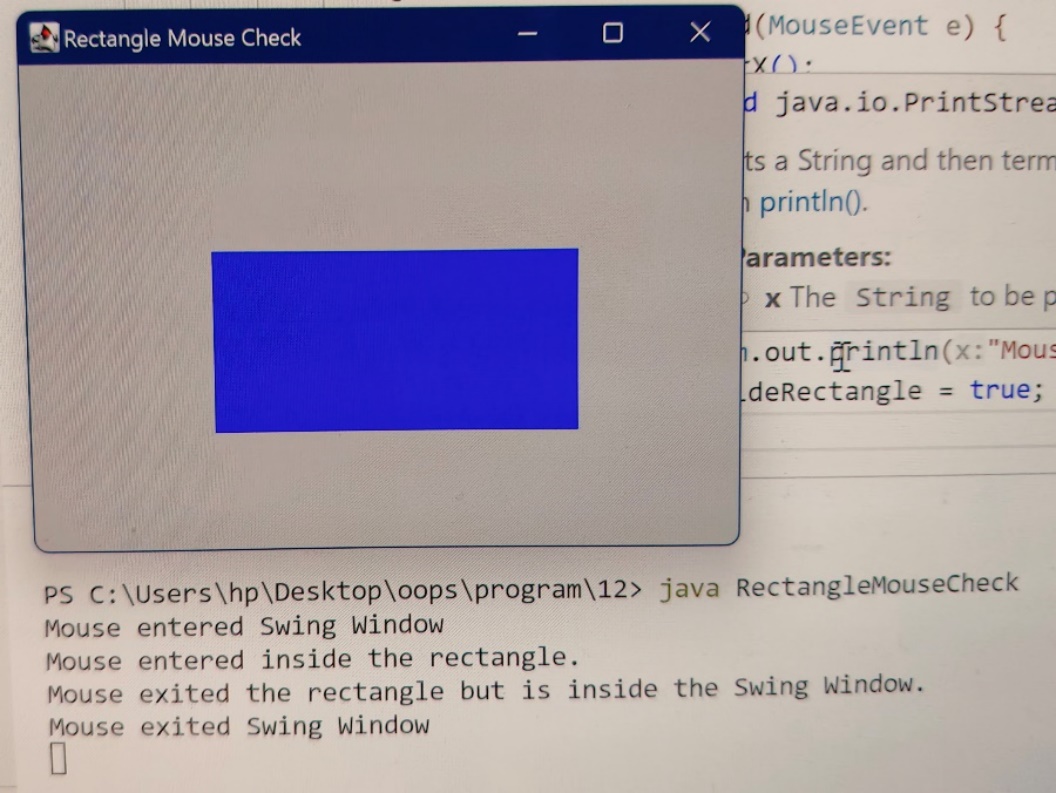
**Fig 1**

****

**Fig 2**

****

**Fig 3**



**Fig 4**

**Program-13**

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

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class MouseEventExample extends JFrame {

    static int timer = 100;

    static int timer2 = 30;

    public MouseEventExample() {// constructor

        setTitle("Handle Mouse Events");

        setSize(400, 300);

        setLocationRelativeTo(null); // Center the window on windows screen

        addWindowListener(new WindowAdapter() {

            @Override

            public void windowClosing(WindowEvent e) {

                int confirm = JOptionPane.showConfirmDialog(MouseEventExample.this,

                        "Are you sure you want to close?",

                        "Confirm Exit",

                        JOptionPane.YES\_NO\_OPTION);

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

                    dispose(); // Close the frame

                }

            }

        });

        setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE); // Prevent default close

        add(new panel());

        setVisible(true);

    }

    class panel extends JPanel {

        public panel() {

            addMouseListener(new MouseAdapter() {

                @Override

                public void mousePressed(MouseEvent e) {

                    System.out.println("Mouse pressed at: " + e.getPoint());

                }

                @Override

                public void mouseReleased(MouseEvent e) {

                    System.out.println("Mouse released at: " + e.getPoint());

                }

                @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 mouseClicked(MouseEvent e) {

                    System.out.println("Mouse clicked at: " + e.getPoint());

                }

            });

            // Add a MouseMotionListener to handle mouse movement

            addMouseMotionListener(new MouseAdapter() {

                @Override

                public void mouseMoved(MouseEvent e) {

                    if (timer > 0) {

                        timer = timer - 1;

                    } else {

                        System.out.println("Mouse moved at: " + e.getPoint());

                        timer = 100;

                    }

                }

                @Override

                public void mouseDragged(MouseEvent e) {

                    if (timer2 > 0) {

                        timer2 = timer2 - 1;

                    } else {

                        System.out.println("Mouse dragged at: " + e.getPoint());

                        timer2 = 30;

                    }

                }

            });

        }

        @Override

        protected void paintComponent(Graphics g) {

            super.paintComponent(g);

            setBackground(Color.GRAY);

        }

    }

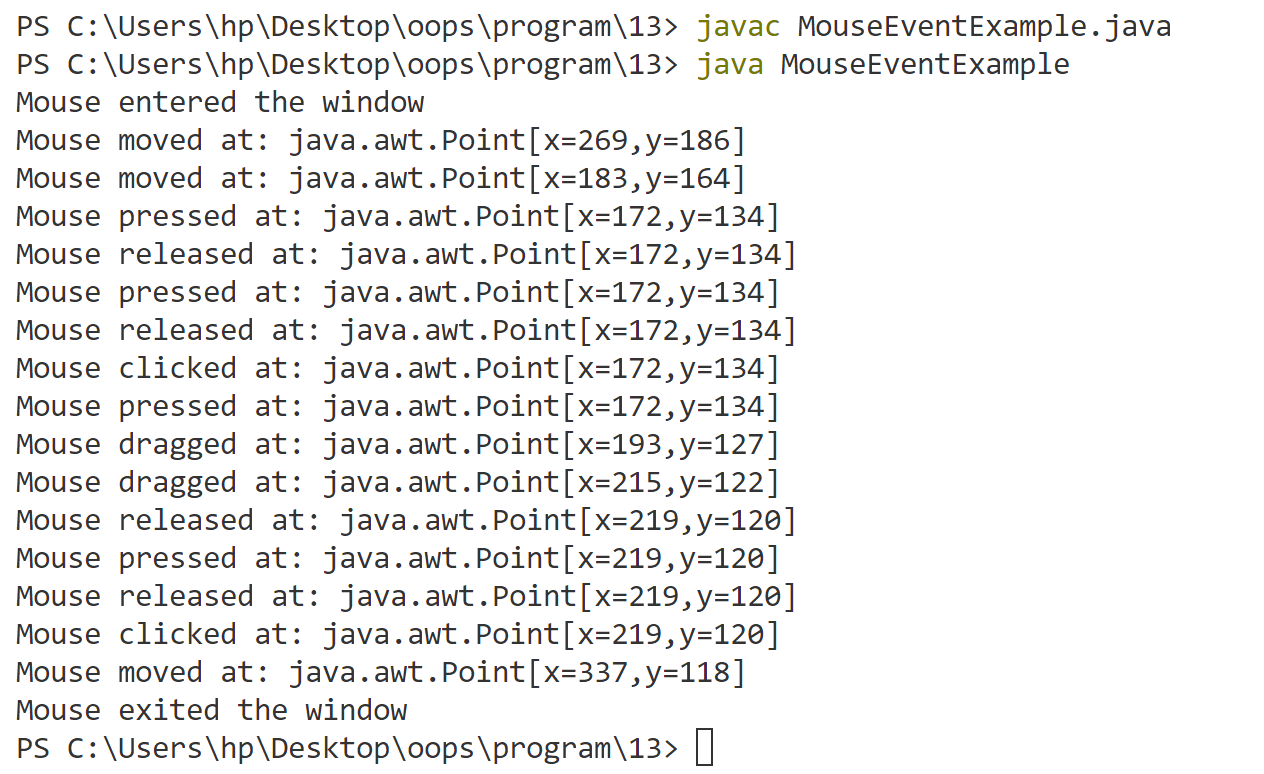
    public static void main(String[] args) {

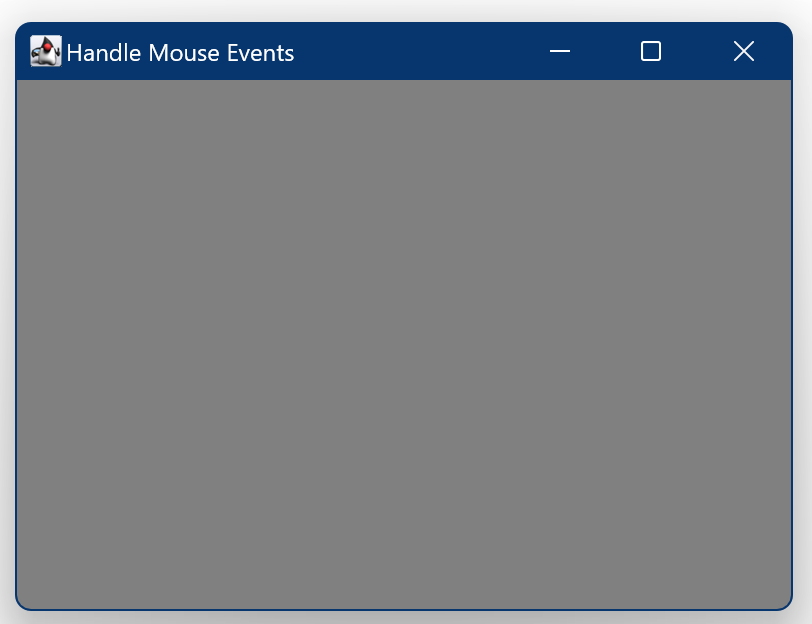
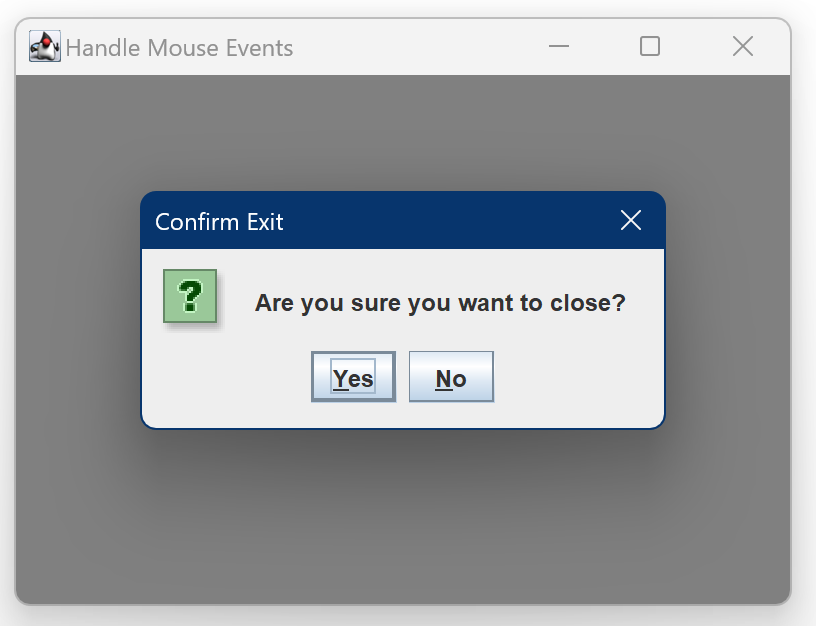
        new MouseEventExample();

    }

}

**Output**

****

****

**Fig 1 Fig 2**

**Program-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 javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class MainFrame extends JFrame {

    public MainFrame() {

        setTitle("ItemEvent and Child Frame Example");

        setSize(400, 300);

        setLocationRelativeTo(null);  // Center the window

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new FlowLayout());

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

        Choice choice = new Choice();

        choice.add("Item 1");

        choice.add("Item 2");

        choice.add("Item 3");

        choice.add("Item 4");

        // Add ItemListener using an anonymous inner class

        choice.addItemListener(new ItemListener() {

            @Override

            public void itemStateChanged(ItemEvent e) {

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

                    System.out.println("Selected Item: " + e.getItem());

                }

            }

        });

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

        // Add ActionListener to the button using an anonymous inner class

        childFrameButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                openChildFrame();

            }

        });

        add(label);

        add(choice);

        add(childFrameButton);

        setVisible(true);

    }

    public void openChildFrame() {

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

        childFrame.setSize(300, 200);

        childFrame.setLocationRelativeTo(this);// 'this' refers to the current MainFrame instance

        childFrame.setLayout(new FlowLayout());

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

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

        childFrame.setVisible(true);

    }

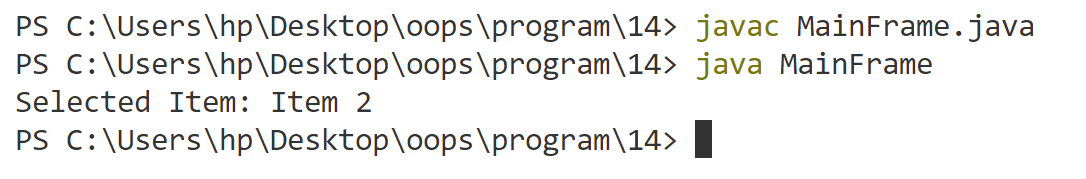
    public static void main(String[] args) {

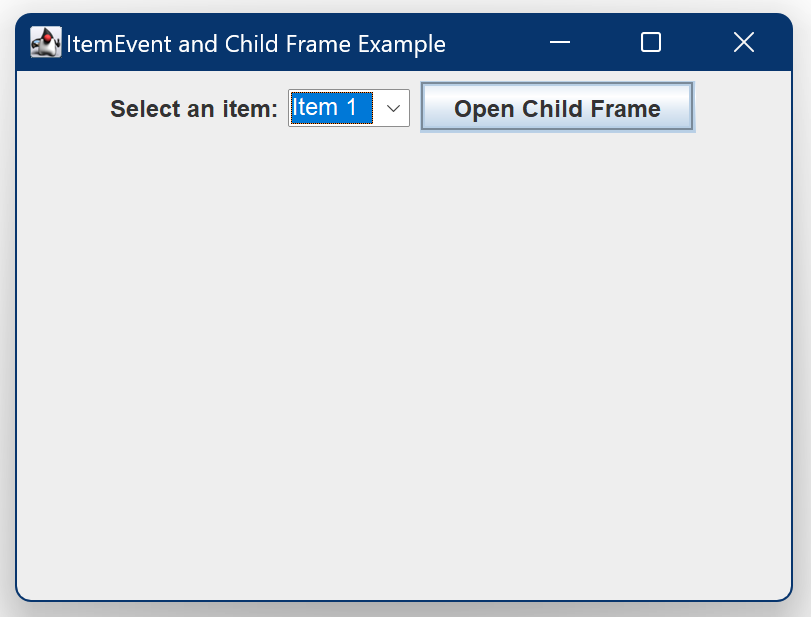
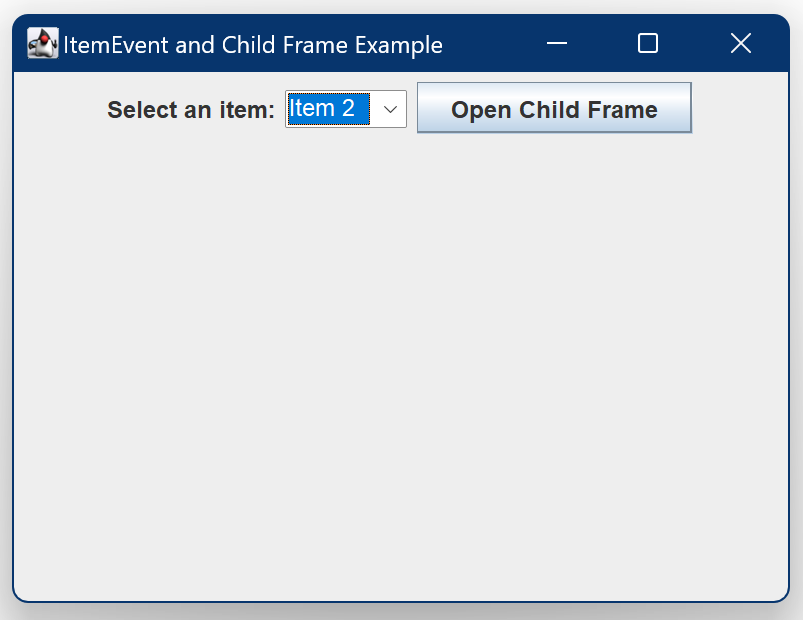
        new MainFrame();

    }

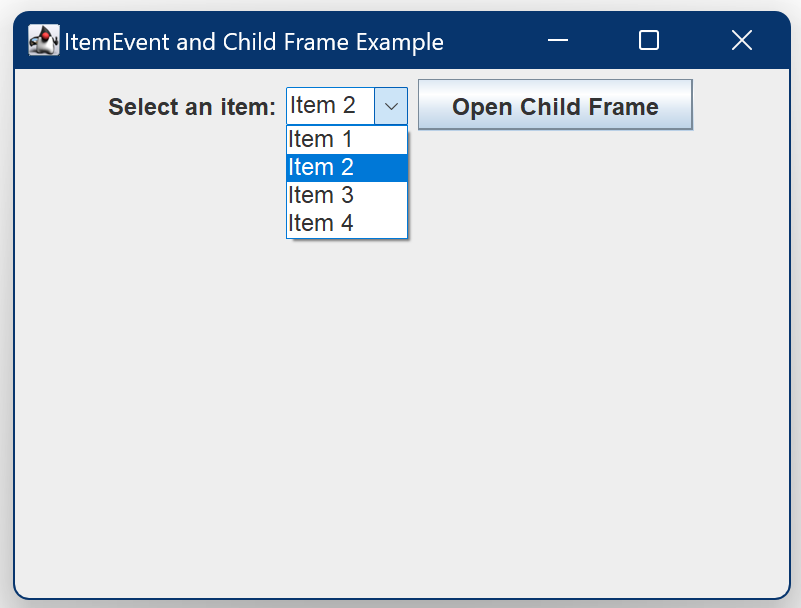
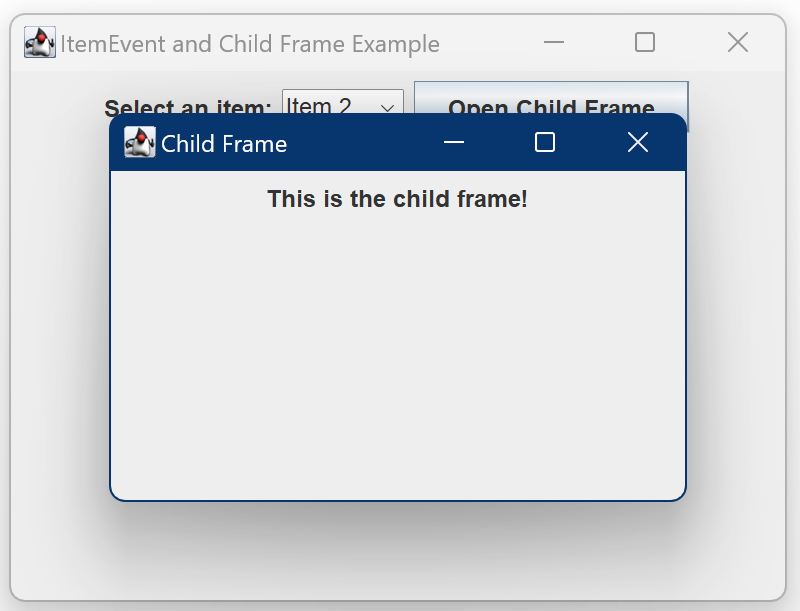
}

**Output**

****



**Fig 1 Fig 2**



**Fig 3 Fig 4**

**Program-15**

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

import java.sql.\*;

import java.util.Scanner;

public class Student {

    private static final String URL = "jdbc:mysql://localhost:3306/";

    private static final String USER = "root";

    private static final String PASSWORD = "Rudvcs@145";

    private Connection conn;

    public Student() {

        try {

            conn = DriverManager.getConnection(URL, USER, PASSWORD);

            System.out.println("Connected to MySQL!");

        } catch (Exception e) {

            System.out.println("Connection Failed!");

        }

    }

    public void createDatabase(String dbName) {

        try {

            Statement statement = conn.createStatement();

            statement.execute("CREATE DATABASE " + dbName);

            System.out.println("Database " + dbName + " created successfully.");

        } catch (SQLException e) {

            System.out.println("Database not created");

        }

    }

    public void createTable(String dbName) {

        String query = "CREATE TABLE Student (" + "rollno INT PRIMARY KEY, " +

                    "name VARCHAR(50), " + "age INT)";

        try {

            Statement statement = conn.createStatement();

            statement.execute("USE " + dbName);

            statement.execute(query);

            System.out.println("Table Student created successfully in " + dbName);

        } catch (Exception e) {

            System.out.println("Table not created");

        }

    }

    public void insertData(String dbName, int rollno, String name, int age) {

        String query = "INSERT INTO Student (rollno, name, age) VALUES (?, ?, ?)";

        try {

            Statement statement = conn.createStatement();

            PreparedStatement pstatement = conn.prepareStatement(query);

            statement.execute("USE " + dbName);

            pstatement.setInt(1, rollno);

            pstatement.setString(2, name);

            pstatement.setInt(3, age);

            pstatement.execute();

            System.out.println("Data inserted successfully.");

        } catch (Exception e) {

            System.out.println("Error in insertion");

        }

    }

    public void updateData(String dbName, int rollno, String name, int age) {

        String query = "UPDATE Student SET name = ?, age = ? WHERE rollno = ?";

        try {

            Statement statement = conn.createStatement();

            PreparedStatement pstatement = conn.prepareStatement(query);

            statement.execute("USE " + dbName);

            pstatement.setString(1, name);

            pstatement.setInt(2, age);

            pstatement.setInt(3, rollno);

            int rowsUpdated = pstatement.executeUpdate();

            if (rowsUpdated > 0) {

                System.out.println("Data updated successfully.");

            } else {

                System.out.println("Roll number not found.");

            }

        } catch (Exception e) {

            System.out.println("Error in updation");

        }

    }

    public void readTable(String dbName) {

        String query = "SELECT \* FROM Student";

        try (Statement stmt = conn.createStatement()) {

            stmt.execute("USE " + dbName);

            ResultSet rs = stmt.executeQuery(query);

            System.out.println("Roll No    Name    Age");

            while (rs.next()) {

                int rollno = rs.getInt(1);

                String name = rs.getString(2);

                int age = rs.getInt(3);

                System.out.println(rollno + "    " + name + "    " + age);

            }

        } catch (Exception e) {

            System.out.println("error while retrieving data");

        }

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Student student = new Student();

        String dbName = "StudentDB"; // Customize your DB name if needed

        student.createDatabase(dbName);

        while (true) {

            System.out.println("\nMenu:");

            System.out.println("1. Create Table");

            System.out.println("2. Insert Data");

            System.out.println("3. Update Data");

            System.out.println("4. Display Table");

            System.out.println("5. Exit");

            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    student.createTable(dbName);

                    break;

                case 2:

                    System.out.print("Enter Roll No: ");

                    int rollno = scanner.nextInt();

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

                    String name = scanner.next();

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

                    int age = scanner.nextInt();

                    student.insertData(dbName, rollno, name, age);

                    break;

                case 3:

                    System.out.print("Enter Roll No to Update: ");

                    rollno = scanner.nextInt();

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

                    name = scanner.next();

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

                    age = scanner.nextInt();

                    student.updateData(dbName, rollno, name, age);

                    break;

                case 4:

                    student.readTable(dbName);

                    break;

                case 5:

                    System.out.println("Exiting program.");

                    scanner.close();

                    return;

                default:

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

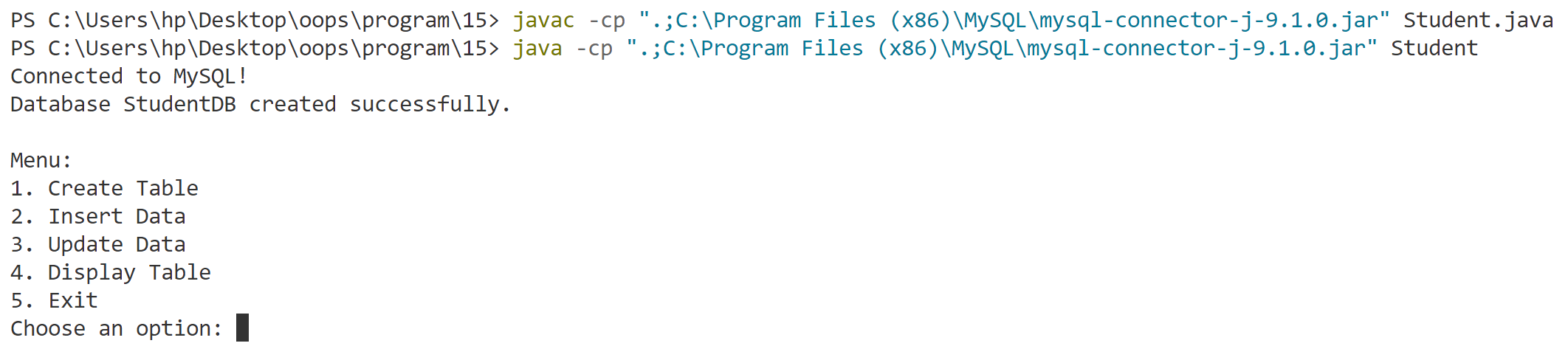
            }

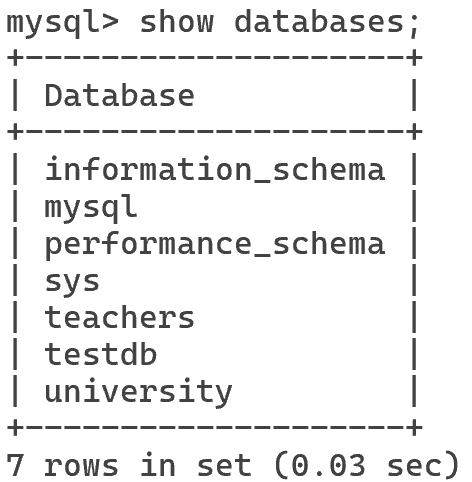
        }

    }

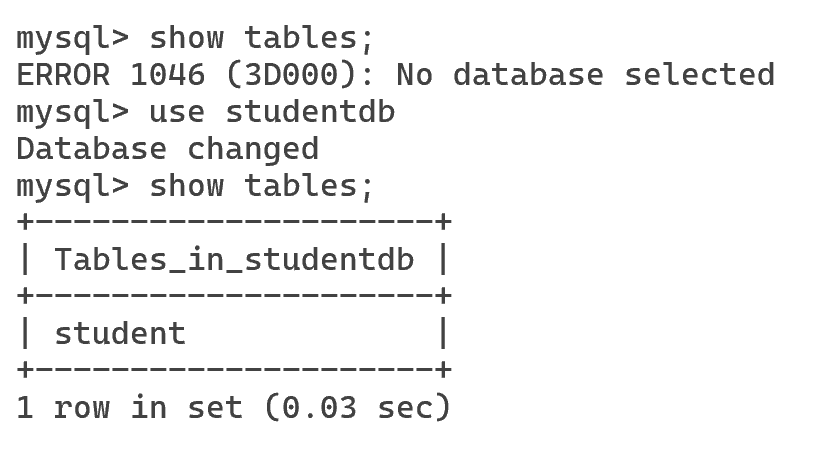
}

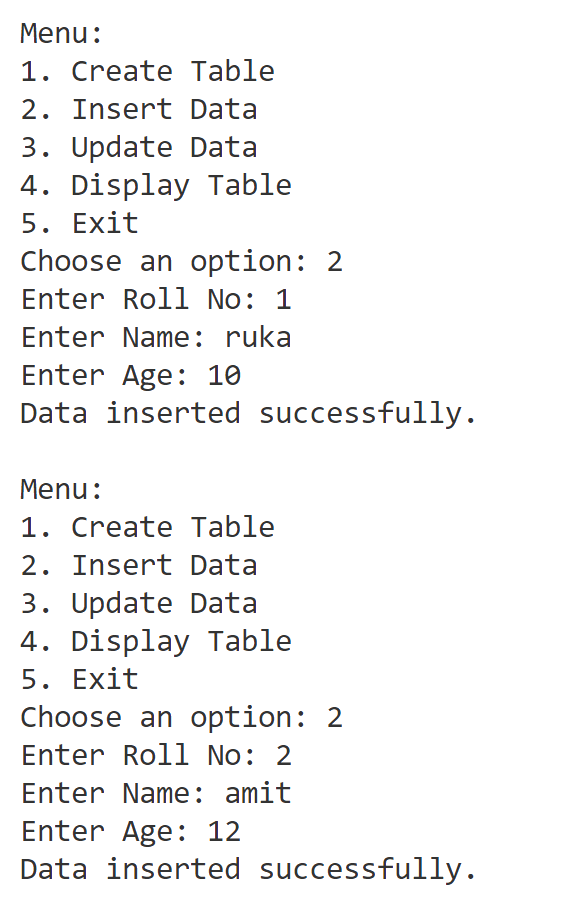
**Output**

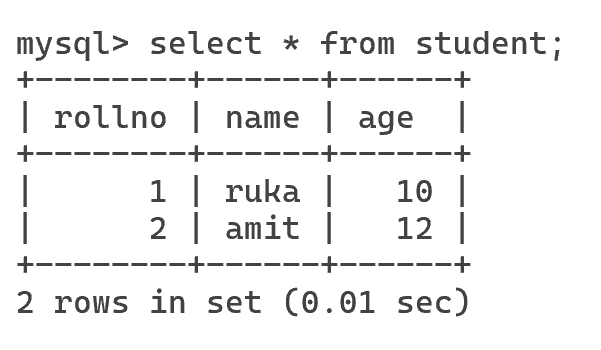
****

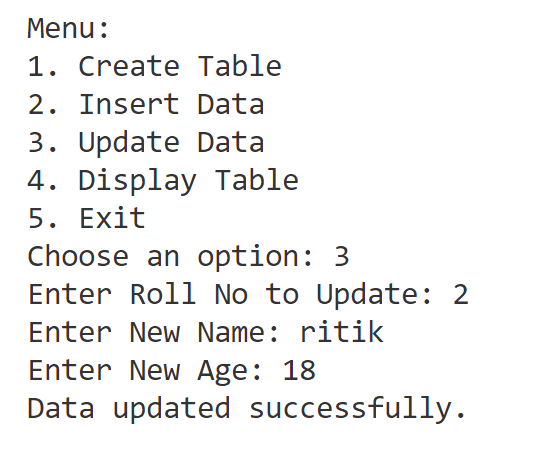
****

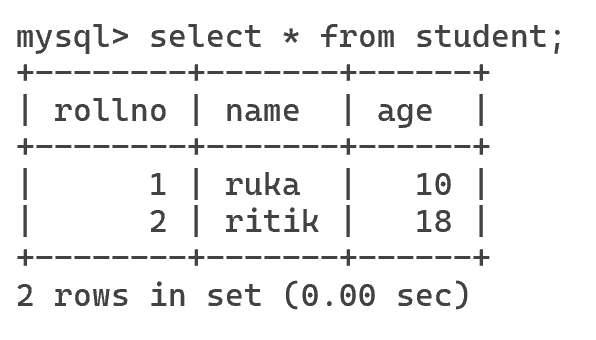
****

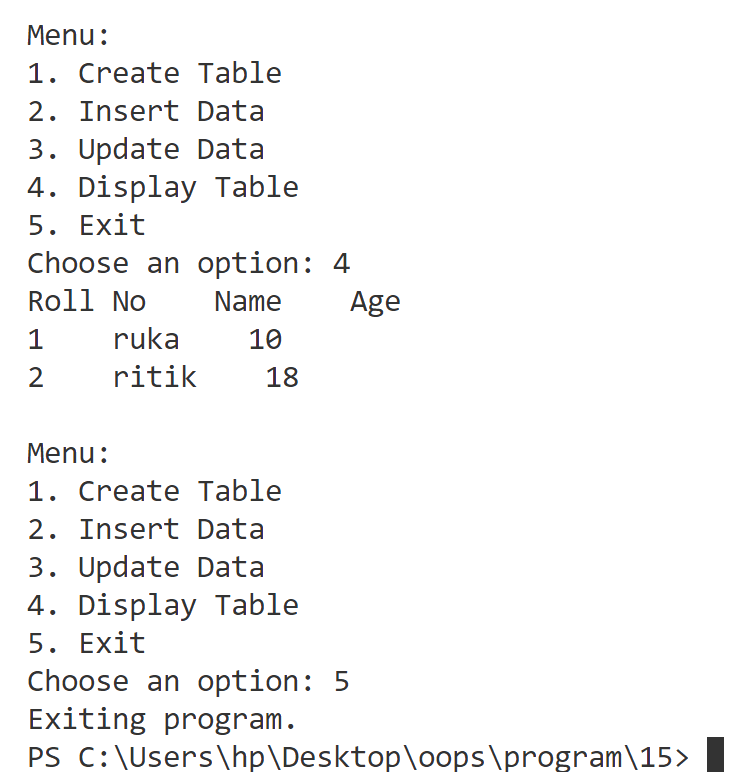
****

****

****

****

****

****