Name: Rishabh Pandey

Roll Number: 87

Subject: Java-Practical

Course: 5 Years M.Sc CS

Semester: 5

1. Create a Class and Object

o Define a Student class with attributes (name, rollNo), create objects and display

details.

Code:-

import java.util.Scanner;

public class Student {

    private String name;

    private int rollNo;

    private int age;

    public Student(String name, int rollNo, int age) {

        this.name = name;

        this.rollNo = rollNo;

        this.age = age;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setRollNo(int rollNo) {

        this.rollNo = rollNo;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public String getName() {

        return this.name;

    }

    public int getRollNo() {

        return this.rollNo;

    }

    public int getAge() {

        return this.age;

    }

    public void display() {

        System.out.println("Name: " + this.name + "\nRoll Number: " + this.rollNo + "\nAge: " + this.age);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the name of the student: ");

        String name = scanner.next();

        System.out.print("Enter the roll number of " + name + ": ");

        int rollNo = scanner.nextInt();

        System.out.print("Enter the age of " + name + ": ");

        int age = scanner.nextInt();

        scanner.close();

        Student student = new Student(name, rollNo, age);

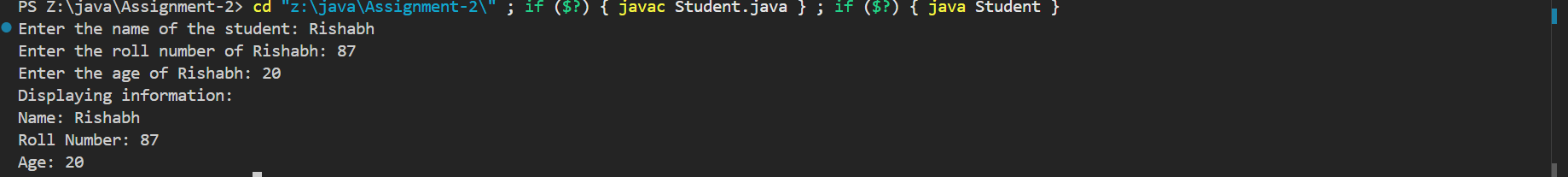
        System.out.println("Displaying information: ");

        student.display();

    }

}

Output:-



2. Constructor Example

o Create a Book class with a constructor to initialize book name and author, and a

method to display them.

Code:-

import java.util.Scanner;

public class Book {

    private String bookName;

    private String authorName;

    public Book(String bookName, String authorName) {

        this.bookName = bookName;

        this.authorName = authorName;

    }

    public void display() {

        System.out.println("Book Name: " + this.bookName + "\nAuthor Name: " + this.authorName);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the name of book: ");

        String bookName = scanner.nextLine();

        System.out.print("Enter the name of author: ");

        String authorName = scanner.nextLine();

        scanner.close();

        Book book = new Book(bookName, authorName);

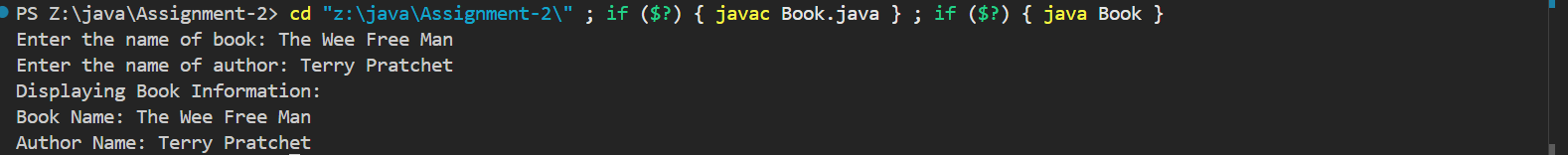
        System.out.println("Displaying Book Information: ");

        book.display();

    }

}

Output:-



3. Default and Parameterized Constructor

o Car class with two constructors: one default and one parameterized.

Code:-

import java.util.Scanner;

public class Car {

    private String brandName;

    private String modelName;

    public Car() {

        this.brandName = "NULL";

        this.modelName = "NULL";

    }

    public Car(String brandName, String modelName) {

        this.brandName = brandName;

        this.modelName = modelName;

    }

    public void display() {

        System.out.println("Brand Name: " + this.brandName + "\nModel Name: " + this.modelName);

    }

    public static void main(String[] args) {

        Car car1 = new Car();

        System.out.println("Created a car object with default constructor: ");

        car1.display();

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the brand of car: ");

        String brandName = scanner.nextLine();

        System.out.print("Enter the model name of car: ");

        String modelName = scanner.nextLine();

        scanner.close();

        Car car2 = new Car(brandName, modelName);

        System.out.println("Created a car object parameterized constructor: ");

        car2.display();

    }

}

Output:-



4. Function Overloading

o Calculator class with multiple add() methods:

 add(int, int), add(double, double), add(int, int, int)

Code:-

public class Calculator {

    public static int add(int a, int b) {

        return a + b;

    }

    public static int add(int a, int b, int c) {

        return a + b + c;

    }

    public static double add(double a, double b) {

        return a + b;

    }

    public static void main(String[] args) {

        System.out.println("Adding two integers (5 and 6): " + Calculator.add(5, 6));

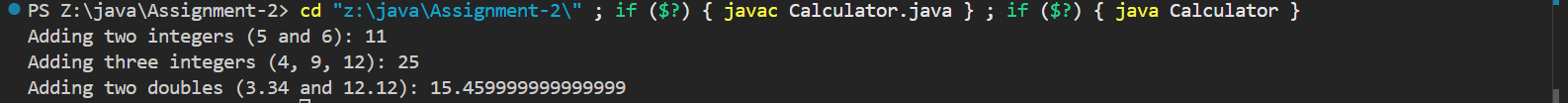
        System.out.println("Adding three integers (4, 9, 12): " + Calculator.add(4, 9, 12));

        System.out.println("Adding two doubles (3.34 and 12.12): " + Calculator.add(3.34, 12.12));

    }

}

Output:-



5. Constructor Overloading

o Employee class with overloaded constructors to initialize with different sets of

data (e.g., name only, name and id, name, id, and salary).

Code:-

import java.util.Scanner;

public class Employee {

    private int id;

    private String name;

    private double salary;

    public Employee(String name) {

        this(name, 0, 0.0);

    }

    public Employee(String name, int id) {

        this(name, id, 0.0);

    }

    public Employee(String name, int id, double salary) {

        if (salary < 0) {

            throw new ArithmeticException("Salary cannot be less than 0");

        }

        this.name = name;

        this.id = id;

        this.salary = salary;

    }

    public void display() {

        System.out.println("Employee Name: " + this.name + "\nID: " + this.id + "\nSalary: " + this.salary);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Creating an employee with only name\nEnter the name of employee: ");

        String name = scanner.nextLine();

        Employee emp1 = new Employee(name);

        System.out.println("\nCreating an employee with name and id\nEnter the name of employee: ");

        name = scanner.nextLine();

        System.out.println("Enter the ID of employee: ");

        int id = scanner.nextInt();

        scanner.nextLine();

        Employee emp2 = new Employee(name, id);

        System.out.println("\nCreating an employee with name, id and salary\nEnter the name of employee: ");

        name = scanner.nextLine();

        System.out.println("Enter the ID of employee: ");

        id = scanner.nextInt();

        System.out.println("Enter the salary of employee: ");

        double salary = scanner.nextDouble();

        Employee emp3 = new Employee(name, id, salary);

        scanner.close();

        System.out.println("Employee created with only name:-");

        emp1.display();

        System.out.println("Employee created with name and id:-");

        emp2.display();

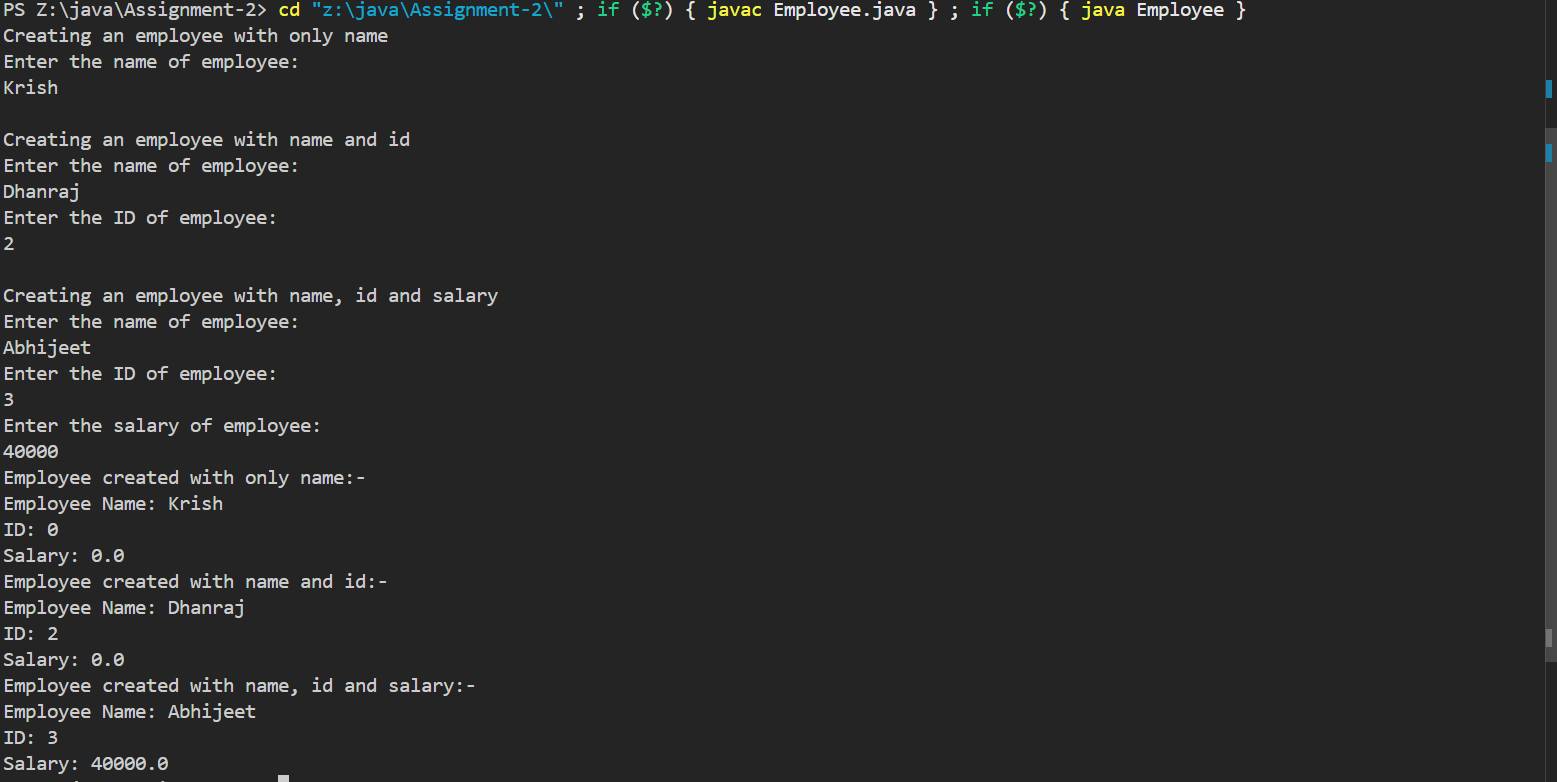
        System.out.println("Employee created with name, id and salary:-");

        emp3.display();

    }

}

Output:-



6. Class with Method to Calculate Area

o Create a Rectangle class with length and width, and a method

calculateArea().

Code:-

import java.util.Scanner;

public class Rectangle {

    private int length;

    private int width;

    public Rectangle(int length, int width) {

        this.length = length;

        this.width = width;

    }

    public int calculateArea() {

        return this.length \* this.width;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the length of rectangle: ");

        int length = scanner.nextInt();

        System.out.print("Enter the width of rectangle: ");

        int width = scanner.nextInt();

        scanner.close();

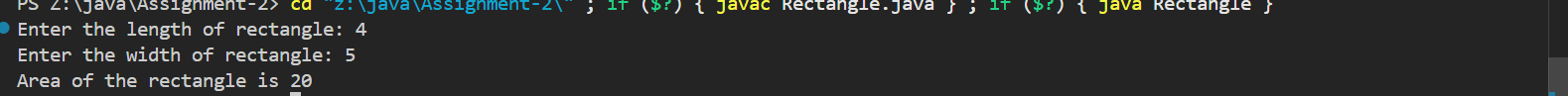
        Rectangle rectangle = new Rectangle(length, width);

        System.out.println("Area of the rectangle is " + rectangle.calculateArea());

    }

}

Output:-



7. Student Class with Marks and Average

o Accept marks of 3 subjects using constructor, calculate average using method.

Code:-

import java.util.Scanner;

public class StudentMarks {

    private int mathsMarks;

    private int scienceMarks;

    private int englishMarks;

    public StudentMarks(int mathsMarks, int scienceMarks, int englishMarks) {

        this.mathsMarks = mathsMarks;

        this.scienceMarks = scienceMarks;

        this.englishMarks = englishMarks;

    }

    public double getAverage() {

        return (this.mathsMarks + this.scienceMarks + this.englishMarks) / 3.0;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the marks of maths, science and english subjects respectively: ");

        int mathsMarks = scanner.nextInt();

        int scienceMarks = scanner.nextInt();

        int englishMarks = scanner.nextInt();

        scanner.close();

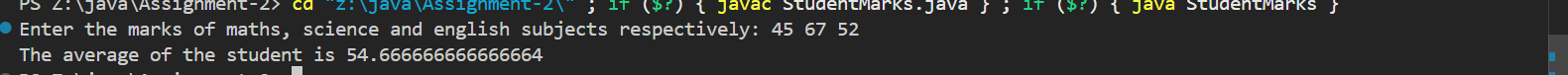
        StudentMarks studentMarks = new StudentMarks(mathsMarks, scienceMarks, englishMarks);

        System.out.println("The average of the student is " + studentMarks.getAverage());

    }

}

Output:-



8. Bank Account Class

o Class BankAccount with deposit, withdraw, and showBalance methods; use

constructors to initialize account.

Code:-

import java.util.Scanner;

class InvalidDeposit extends Exception {

    public InvalidDeposit(String message) {

        super(message);

    }

}

class InvalidWithdraw extends Exception {

    public InvalidWithdraw(String message) {

        super(message);

    }

}

public class BankAccount {

    private double balance;

    public BankAccount(double balance) {

        this.balance = balance;

    }

    public void depositAmount(double depositAmount) throws InvalidDeposit {

        if (depositAmount < 0) {

            throw new InvalidDeposit("Deposit amount cannot be negative");

        }

        this.balance += depositAmount;

    }

    public void withdrawAmount(double withdrawAmount) throws InvalidWithdraw {

        if (withdrawAmount < 0) {

            throw new InvalidWithdraw("Withdrawn amount cannot be negative");

        }

        if (this.balance < withdrawAmount) {

            throw new InvalidWithdraw("Not enough balance in the account");

        }

        this.balance -= withdrawAmount;

    }

    public void showBalance() {

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

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        BankAccount bankAccount = new BankAccount(0);

        boolean continueLoop = true;

        while (continueLoop) {

            System.out.print("1. Deposit\n2. Withdraw\n3. Show Balance\n4. Exit\nChoose: ");

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    System.out.println("Enter the amount to deposit");

                    int depositAmount = scanner.nextInt();

                    try {

                        bankAccount.depositAmount(depositAmount);

                    } catch (InvalidDeposit e) {

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

                    }

                    break;

                case 2:

                    System.out.println("Enter the amount to withdraw");

                    int withdrawAmount = scanner.nextInt();

                    try {

                        bankAccount.withdrawAmount(withdrawAmount);

                    } catch (InvalidWithdraw e) {

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

                    }

                    break;

                case 3:

                    bankAccount.showBalance();

                    break;

                case 4:

                    continueLoop = false;

                    break;

                default:

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

            }

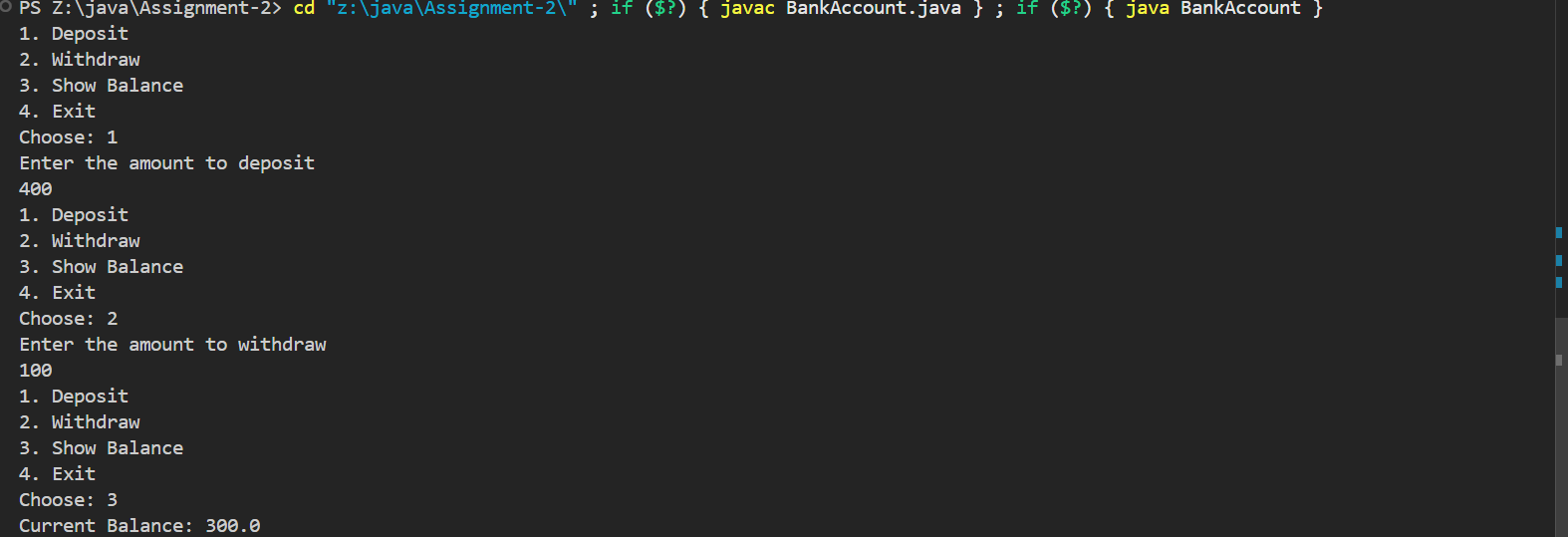
        }

        scanner.close();

    }

}

Output:-



9. Class with Object as a Member

o Create Address and Employee classes. Employee has an Address object as a

member.

Code:-

class Address {

    private String area;

    private String state;

    private String country;

    private long pinCode;

    public Address(String area, String state, String country, long pinCode) {

        this.area = area;

        this.state = state;

        this.country = country;

        this.pinCode = pinCode;

    }

    public String getArea() {

        return this.area;

    }

    public String getState() {

        return this.state;

    }

    public String getCountry() {

        return this.country;

    }

    public long getPinCode() {

        return this.pinCode;

    }

}

public class EmployeeDetail {

    private int empId;

    private String empName;

    private Address empAddress;

    private double empSalary;

    public EmployeeDetail(int id, String name, String area, String state, String country, long pinCode, double salary) {

        this.empId = id;

        this.empName = name;

        this.empAddress = new Address(area, state, country, pinCode);

        this.empSalary = salary;

    }

    public void display() {

        System.out.println("Employee ID: " + this.empId + "\nEmployee Name: " + this.empName +

                           "\nArea of Residence: " + this.empAddress.getArea() +

                           "\nState of Residence: " + this.empAddress.getState() +

                           "\nCountry of Residence: " + this.empAddress.getCountry() +

                           "\nPin-Code: " + this.empAddress.getPinCode() +

                           "\nSalary: " + this.empSalary);

    }

    public static void main(String[] args) {

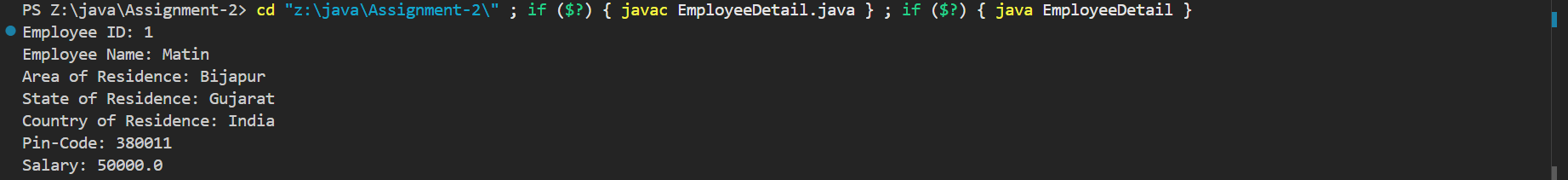
        EmployeeDetail empDetail = new EmployeeDetail(1, "Matin", "Bijapur", "Gujarat", "India", 380011, 50000);

        empDetail.display();

    }

}

Output:-



10.Function Overloading in Constructor and Method

 Shape class with overloaded constructors for circle and rectangle. Also overload

area() method to handle both shapes.

Code:-

import java.util.Scanner;

public class Shape {

    private double radius;

    private double length;

    private double breadth;

    public Shape(double radius) {

        this.radius = radius;

        this.length = this.breadth = -1;

        this.area(this.radius);

    }

    public Shape(double length, double breadth) {

        this.length = length;

        this.breadth = breadth;

        this.radius = -1;

        this.area(this.length, this.breadth);

    }

    public void area(double radius) {

        System.out.println("Area of Circle with radius " + radius + " is " + 3.14 \* radius \* radius);

    }

    public void area(double length, double breadth) {

        System.out.println("Area of Rectangle with length " + length + " and breadth " + breadth + " is " + (length \* breadth));

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the radius of circle: ");

        double r = scanner.nextDouble();

        Shape circle = new Shape(r);

        System.out.println("Enter the length and breadth of rectangle: ");

        double l = scanner.nextDouble();

        double b = scanner.nextDouble();

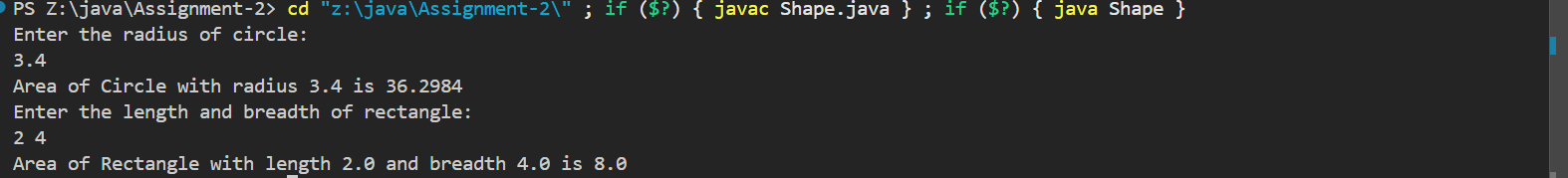
        Shape rectangle = new Shape(l, b);

        scanner.close();

    }

}

Output:-



11.Class with Private Members and Public Getters/Setters

 Student class with private fields (name, age) and public methods to access them using

getter/setter methods. Use constructor to initialize.

Code:-

import java.util.Scanner;

public class Student {

    private String name;

    private int rollNo;

    private int age;

    public Student(String name, int rollNo, int age) {

        this.name = name;

        this.rollNo = rollNo;

        this.age = age;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setRollNo(int rollNo) {

        this.rollNo = rollNo;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public String getName() {

        return this.name;

    }

    public int getRollNo() {

        return this.rollNo;

    }

    public int getAge() {

        return this.age;

    }

    public void display() {

        System.out.println("Name: " + this.name + "\nRoll Number: " + this.rollNo + "\nAge: " + this.age);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the name of the student: ");

        String name = scanner.next();

        System.out.print("Enter the roll number of " + name + ": ");

        int rollNo = scanner.nextInt();

        System.out.print("Enter the age of " + name + ": ");

        int age = scanner.nextInt();

        scanner.close();

        Student student = new Student(name, rollNo, age);

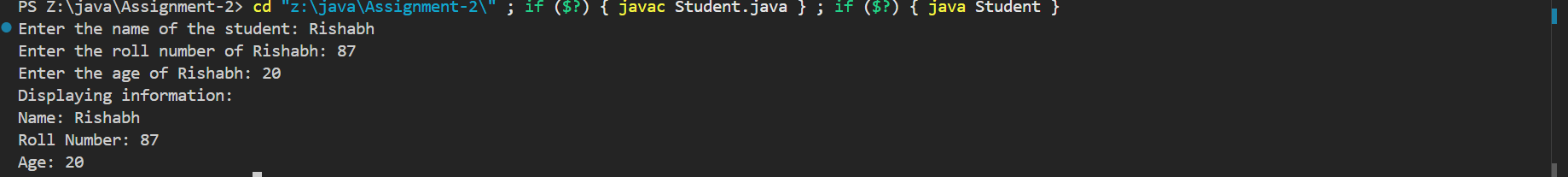
        System.out.println("Displaying information: ");

        student.display();

    }

}

Output:-



12.Array of Objects

 Create a Product class and an array of Product objects. Accept data and display all

products using loop.

Code:-

import java.util.Scanner;

public class Product {

    private String productName;

    private double productPrice;

    public Product(String productName, double productPrice) {

        this.productName = productName;

        this.productPrice = productPrice;

    }

    public void display() {

        System.out.println("Product Name: " + this.productName + "\nProduct Price: " + this.productPrice);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Product []products = new Product[3];

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

            System.out.println("\nEnter the information of product number " + (i + 1));

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

            String name = scanner.nextLine();

            System.out.print("Enter the product price: ");

            double price = scanner.nextDouble();

            scanner.nextLine();

            products[i] = new Product(name, price);

        }

        scanner.close();

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

            System.out.println("\nInformation of product number " + (i + 1) + ":-");

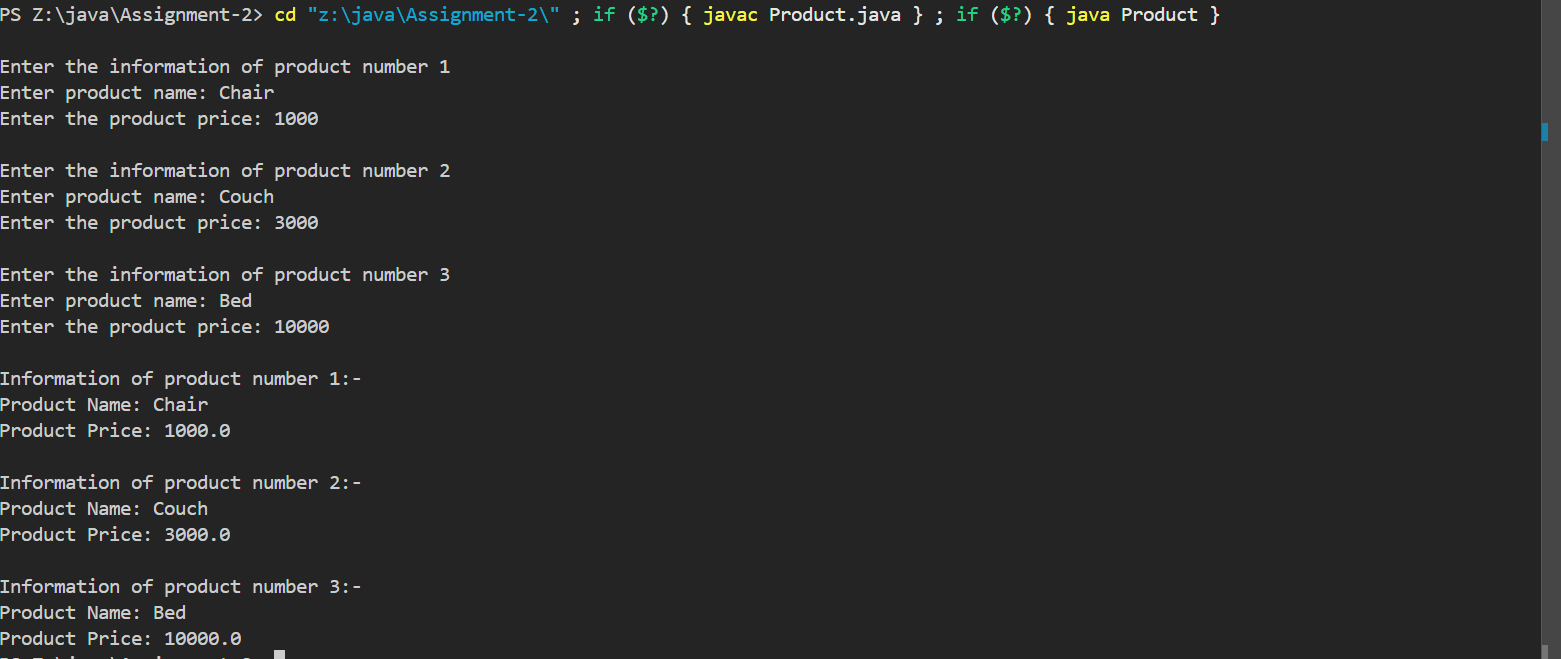
            products[i].display();

        }

    }

}

Output:-



13.Constructor with Validation using Exception

 Employee constructor throws an exception if salary is negative.

Code:-

import java.util.Scanner;

public class Employee {

    private int id;

    private String name;

    private double salary;

    public Employee(String name) {

        this(name, 0, 0.0);

    }

    public Employee(String name, int id) {

        this(name, id, 0.0);

    }

    public Employee(String name, int id, double salary) {

        if (salary < 0) {

            throw new ArithmeticException("Salary cannot be less than 0");

        }

        this.name = name;

        this.id = id;

        this.salary = salary;

    }

    public void display() {

        System.out.println("Employee Name: " + this.name + "\nID: " + this.id + "\nSalary: " + this.salary);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the name of employee: ");

        String name = scanner.nextLine();

        System.out.println("Enter the ID of employee: ");

        int id = scanner.nextInt();

        System.out.println("Enter the salary of employee: ");

        double salary = scanner.nextDouble();

        Employee emp3 = new Employee(name, id, salary);

        scanner.close();

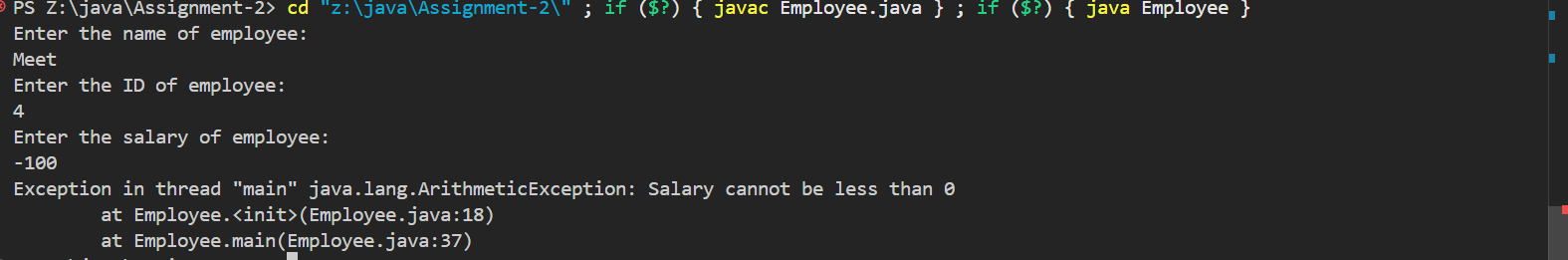
        System.out.println("Employee created with name, id and salary:-");

        emp3.display();

    }

}

Output:-



14.Custom Exception Handling

 Create a custom exception InvalidAgeException. Throw it if age < 18 in a method

checkEligibility().

Code:-

import java.util.Scanner;

class InvalidAgeException extends Exception {

    public InvalidAgeException(String message) {

        super(message);

    }

}

public class Person {

    private String name;

    private int age;

    public Person(String name, int age) throws InvalidAgeException {

        if (age < 18) {

            throw new InvalidAgeException("Age must be more than or equal to 18");

        }

        this.name = name;

        this.age = age;

    }

    public void display() {

        System.out.println("Name: " + this.name + "\nAge: " + this.age);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String name = scanner.nextLine();

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

        int age = scanner.nextInt();

        scanner.close();

        try {

            Person person = new Person(name, age);

            person.display();

        } catch (InvalidAgeException e) {

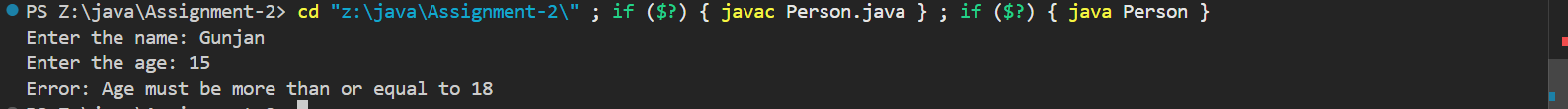
            System.out.println("Error: " + e.getMessage());

        }

    }

}

Output:-



15.Static vs Non-static Members

 University class with static universityName and non-static studentName.

Demonstrate calling static vs non-static members.

Code:-

import java.util.Scanner;

public class University {

    public final static String universityName = "Gujarat University";

    private String studentName;

    public University(String name) {

        this.studentName = name;

    }

    public void display() {

        System.out.println("Student Name: " + this.studentName + "\nUniversity: " + University.universityName);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        University student = new University("Rishabh");

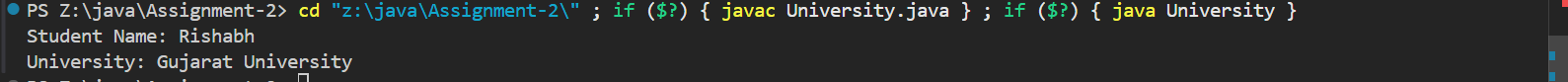
        scanner.close();

        student.display();

    }

}

Output:-



16.Multiple Classes with Relationships

 Department and Professor class. Each Professor is linked to a Department object.

Code:-

class Department {

    private String departmentName;

    private Professor []professors;

    private int professorCount;

    public Department(String name) {

        this.departmentName = name;

        this.professors = new Professor[5];

        this.professorCount = 0;

    }

    public void addProfessor(Professor professor) {

        this.professors[this.professorCount++] = professor;

    }

    public void display() {

        System.out.println("Department Name: " + this.departmentName);

        System.out.println("Faculties:- ");

        for (int i = 0; i < this.professorCount; i++) {

            this.professors[i].display();

        }

    }

}

class Professor {

    private String professorName;

    private Department department;

    public Professor(String name, Department department) {

        this.professorName = name;

        this.department = department;

        department.addProfessor(this);

    }

    public void display() {

        System.out.println("Professor Name: " + this.professorName);

    }

}

public class Main {

    public static void main(String[] args) {

        Department dept1 = new Department("Department of Botany");

        Department dept2 = new Department("Department of Computer Science");

        Professor prof1 = new Professor("Jatin Shah", dept1);

        Professor prof2 = new Professor("Kyora Sazanami", dept2);

        Professor prof3 = new Professor("Parth Patil", dept2);

        dept1.display();

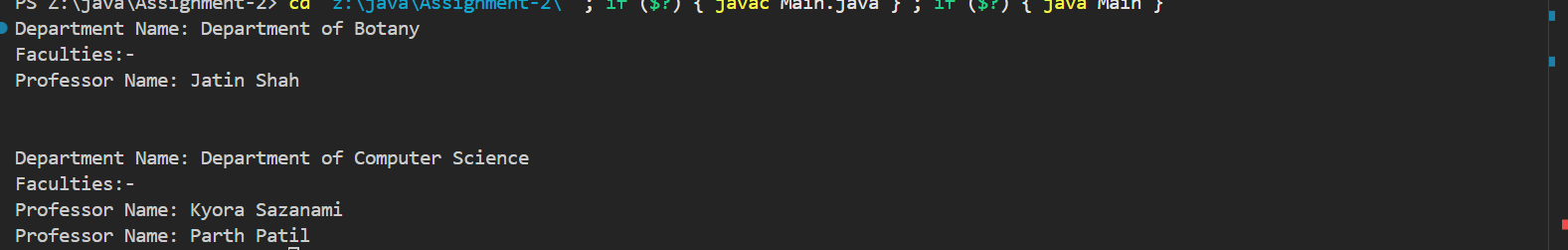
        System.out.println("\n");

        dept2.display();

    }

}

Output:-



17.Array of Objects with Total Calculation

 Marks class having subject marks, use array of students to calculate and display total

and average marks.

Code:-

class Marks {

    private String subjectName;

    private int marks;

    public Marks(String subjectName, int marks) {

        this.subjectName = subjectName;

        this.marks = marks;

    }

    public String getSubjectName() {

        return this.subjectName;

    }

    public int getMarks() {

        return this.marks;

    }

}

class Student {

    private String studentName;

    private Marks[] marks;

    public Student(String name, Marks[] marks) {

        this.studentName = name;

        this.marks = marks;

    }

    public void display() {

        System.out.println("Name: " + this.studentName);

        for (Marks mark : marks) {

            System.out.println("Subject Name: " + mark.getSubjectName() + "\t\t\tMarks: " + mark.getMarks());

        }

        System.out.println("Total Marks: " + this.getTotalMarks() + "\nAverage Marks: " + this.getAverageMarks());

    }

    public int getTotalMarks() {

        int sum = 0;

        for (Marks mark : marks) {

            sum += mark.getMarks();

        }

        return sum;

    }

    public double getAverageMarks() {

        return (double) this.getTotalMarks() / this.marks.length;

    }

}

public class MarksMain {

    public static void main(String[] args) {

        Marks []student1Marks = {

            new Marks("Social Science", 82),

            new Marks("Science", 73),

            new Marks("Maths", 91)

        };

        Marks []student2Marks = {

            new Marks("Java - Theory", 82),

            new Marks("Data Analytics", 88),

            new Marks("Machine Learning - Theory", 90)

        };

        Student []students = {

            new Student("Krish", student1Marks),

            new Student("Sumer", student2Marks)

        };

        for (Student student : students) {

            student.display();

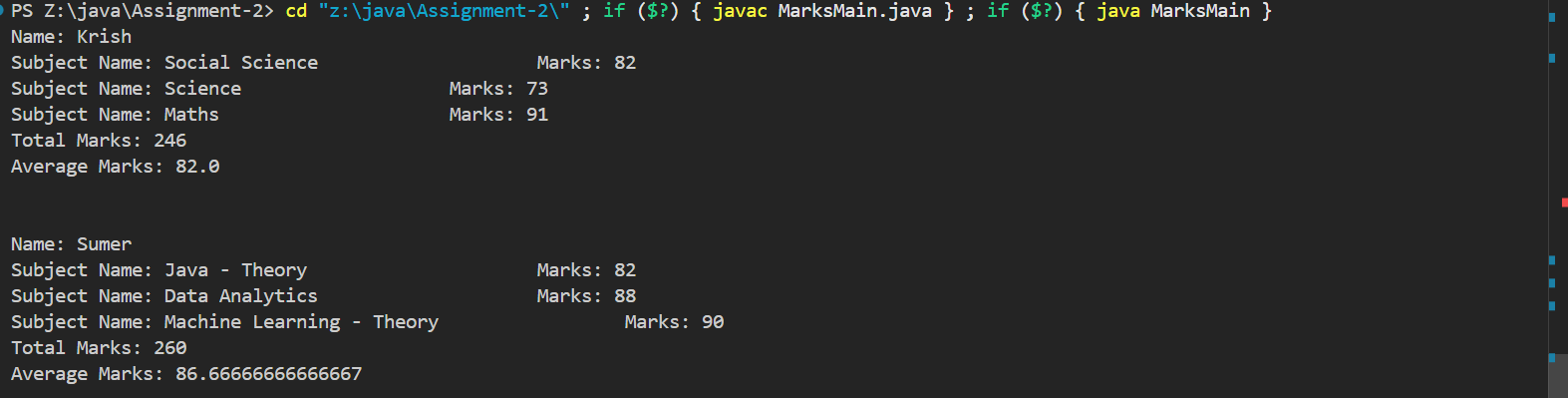
            System.out.println("\n");

        }

    }

}

Output:-



18.Banking System with Exception and Access Modifiers

 Create a BankAccount class with private balance, public deposit() and withdraw().

Throw exception if withdrawal amount > balance.

Code:-

import java.util.Scanner;

class InvalidDeposit extends Exception {

    public InvalidDeposit(String message) {

        super(message);

    }

}

class InvalidWithdraw extends Exception {

    public InvalidWithdraw(String message) {

        super(message);

    }

}

public class BankAccount {

    private double balance;

    public BankAccount(double balance) {

        this.balance = balance;

    }

    public void depositAmount(double depositAmount) throws InvalidDeposit {

        if (depositAmount < 0) {

            throw new InvalidDeposit("Deposit amount cannot be negative");

        }

        this.balance += depositAmount;

    }

    public void withdrawAmount(double withdrawAmount) throws InvalidWithdraw {

        if (withdrawAmount < 0) {

            throw new InvalidWithdraw("Withdrawn amount cannot be negative");

        }

        if (this.balance < withdrawAmount) {

            throw new InvalidWithdraw("Not enough balance in the account");

        }

        this.balance -= withdrawAmount;

    }

    public void showBalance() {

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

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        BankAccount bankAccount = new BankAccount(0);

        boolean continueLoop = true;

        while (continueLoop) {

            System.out.print("1. Deposit\n2. Withdraw\n3. Show Balance\n4. Exit\nChoose: ");

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    System.out.println("Enter the amount to deposit");

                    int depositAmount = scanner.nextInt();

                    try {

                        bankAccount.depositAmount(depositAmount);

                    } catch (InvalidDeposit e) {

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

                    }

                    break;

                case 2:

                    System.out.println("Enter the amount to withdraw");

                    int withdrawAmount = scanner.nextInt();

                    try {

                        bankAccount.withdrawAmount(withdrawAmount);

                    } catch (InvalidWithdraw e) {

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

                    }

                    break;

                case 3:

                    bankAccount.showBalance();

                    break;

                case 4:

                    continueLoop = false;

                    break;

                default:

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

            }

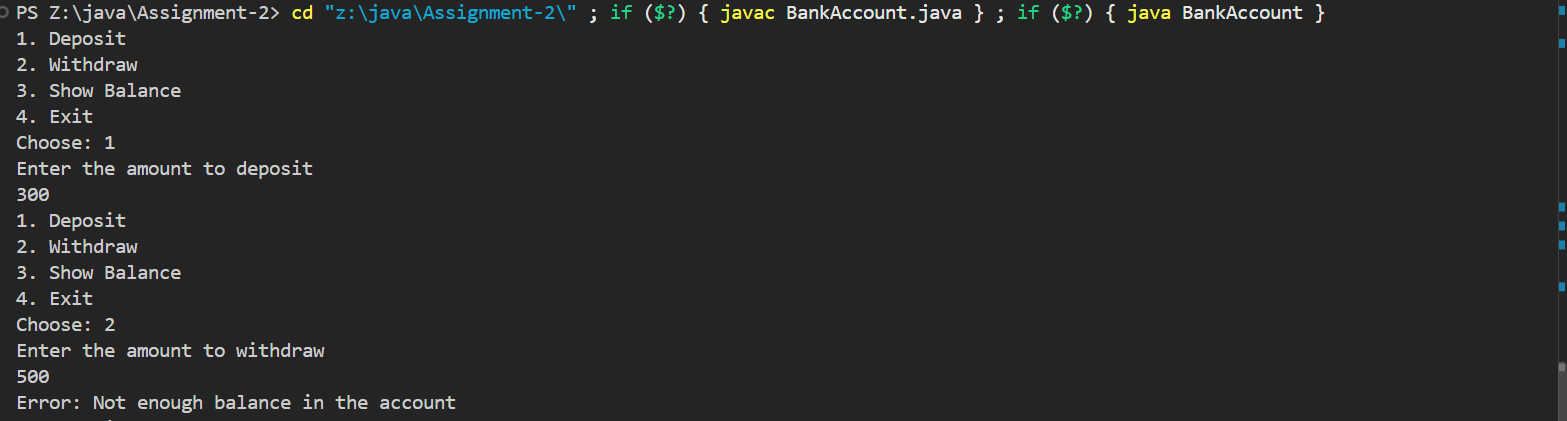
        }

        scanner.close();

    }

}

Output:-



19.Constructor Calling Another Constructor (this())

 Use this() to chain constructors inside a Customer class.

Code:-

public class Customer {

    private String name;

    private int age;

    public Customer(String name, int age) {

        this.name = name;

        this.age = age;

    }

    public Customer(String name) {

        this(name, 0);

    }

    public Customer(int age) {

        this("NULL", age);

    }

    public Customer() {

        this("NULL", 0);

    }

    public void display() {

        System.out.println("Customer Name: " + this.name + "\nAge: " + this.age);

    }

    public static void main(String[] args) {

        Customer cust1 = new Customer("Rohan", 22);

        System.out.println("Created a customer with both name and age:-");

        cust1.display();

        Customer cust2 = new Customer("Kirtan");

        System.out.println("Created a customer with only name:-");

        cust2.display();

        Customer cust3 = new Customer();

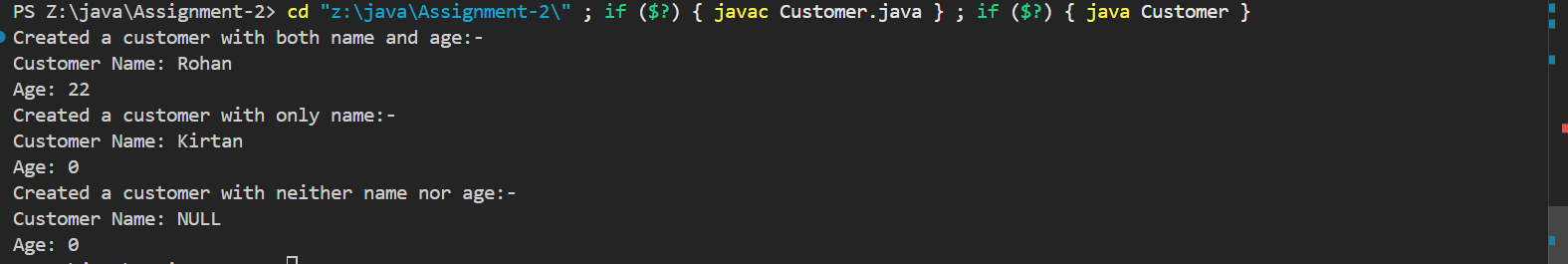
        System.out.println("Created a customer with neither name nor age:-");

        cust3.display();

    }

}

Output:-



20.Library Management with Object Array and Search

 Book class with ID, title, author. Store multiple books and allow searching by book

title.

Code:-

import java.util.Scanner;

class Book {

    private int id;

    private String title;

    private String author;

    public Book(int id, String title, String author) {

        this.id = id;

        this.title = title;

        this.author =  author;

    }

    public String getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    public int getId() {

        return id;

    }

}

class Library {

    private String libraryName;

    private Book[] books;

    public Library(String name, Book []books) {

        this.libraryName = name;

        this.books = books;

    }

    public Book searchBookByTitle(String bookTitle) {

        for (Book book : books) {

            if (book.getTitle().equals(bookTitle)) {

                return book;

            }

        }

        return null;

    }

}

public class LibraryMain {

    public static void main(String[] args) {

        Book []books = {

            new Book(1, "Lightbringer", "C.J. Charlie"),

            new Book(2, "Red Rising", "Ritcher Thomson"),

            new Book(3, "Do Sheep Dreams?", "Roger Faraday")

        };

        Library lib = new Library("Dewevilley Library", books);

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the title of book you want to search: ");

        String title = scanner.nextLine();

        scanner.close();

        Book searchedBook = lib.searchBookByTitle(title);

        if (searchedBook != null) {

            System.out.println("Book written by " + searchedBook.getAuthor() + " with ID " + searchedBook.getId());

        } else {

            System.out.println("We dont have the book in our library!");

        }

    }

}

Output:-

