CORE JAVA PROJECT

GYM MANAGEMENT SYSTEM

MASTER OF COMPUTER APPLICATION

(MCA)

by

Bala Krishna Chennaiah

(Regd. No: 2024016258)

Under the esteemed guidance of

Dr.Santosh Kumar Uppada

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

GITAM INSTITUTE OF SCIENCE

GITAM (Deemed to be University)

VISAKHAPATNAM – 530045, A.P

(2024-2026)

Project Report: Gym Management System

Objective:

The primary goal of the Gym Management System is to manage gym operations efficiently through a console-based Java application. This includes functionalities like member and trainer registration, payment handling, and membership status management.

Technologies Used:

- Language: Java

Concepts Implemented:

- Object-Oriented Programming (Inheritance, Abstraction, Interfaces, Exception Handling)
- Arrays for data storage
- User interaction via Scanner
- Custom Exception (InvalidOperationException)

Key Classes & Interfaces:

1. Person (Abstract Class):

- Base class for Member and Trainer
- Contains common attributes (id, name) and an abstract method greet()

2. Displayable (Interface):

Contains displayInfo() method implemented by Member and Trainer for polymorphic behavior.

3. Member (Extends Person, Implements Displayable):

Stores member details, activation status, and fee info based on membership type.

4. Trainer (Extends Person, Implements Displayable):

Holds trainer-specific info like specialization.

5. Payment:

Processes and logs member payments.

6. InvalidOperationException:

Custom exception to handle invalid operations gracefully.

Main Functionalities:

1. Register Member: Adds a new member with unique ID and type (Basic/Premium).
2. View Members: Displays all registered members.
3. Add/View Trainer: Similar functionality for gym trainers.
4. Make Payment: Computes and processes payments based on membership type.
5. Deactivate/Activate Member: Toggles membership status with validation.
6. Exit: Cleanly terminates the application.

CODE:

```
import java.util.Scanner;

interface Displayable {
   void displayInfo();
}

abstract class Person {
   protected int id;
   protected String name;

   public Person(int id, String name) {
      this.id = id;
      this.name = name;
   }

   public abstract void greet();
}

class InvalidOperationException extends Exception {
   public InvalidOperationException(String msg) {
      super(msg);
   }
}

class Member extends Person implements Displayable {
   private String membershipType;
   private boolean active;

   public Member(int id, String name, String membershipType) {
      super(id, name);
      this.membershipType = membershipType;
      this.active = true;
   }

   public int getId() {
      return id;
   }

   public boolean isActive() {
      return active;
   }

   public void deactivate() {
      active = false;
   }
```

```
    public void activate() {
        active = true;
    }

    public double getFee() {
        return membershipType.equalsIgnoreCase("Premium") ? 1000.0 : 500.0;
    }

    public void displayInfo() {
        System.out.println("Member ID: " + id + " | Name: " + name + " | Type: " +
membershipType + " | Active: " + active);
    }

    public void greet() {
        System.out.println("Hi " + name + ", welcome to the gym!");
    }
}

class Trainer extends Person implements Displayable {
    private String specialization;

    public Trainer(int id, String name, String specialization) {
        super(id, name);
        this.specialization = specialization;
    }

    public void displayInfo() {
        System.out.println("Trainer ID: " + id + " | Name: " + name + " | Specialization: "
+ specialization);
    }

    public void greet() {
        System.out.println("Trainer " + name + " here, ready to help you!");
    }
}

class Payment {
    private Member member;
    private double amount;
    private String method;

    public Payment(Member member, double amount, String method) {
        this.member = member;
        this.amount = amount;
        this.method = method;
    }
```

```
    public void process() {
        System.out.println("Processing payment...");
        System.out.println("Payment of Rs" + amount + " via " + method + " completed
for Member ID: " + member.getId());
    }
}

public class GymManagementSystem {
    static final int MAX_MEMBERS = 100;
    static final int MAX_TRAINERS = 50;
    static Member[] members = new Member[MAX_MEMBERS];
    static Trainer[] trainers = new Trainer[MAX_TRAINERS];
    static int memberCount = 0;
    static int trainerCount = 0;
    static int memberIdGen = 1001;
    static int trainerIdGen = 2001;
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n--- Gym Management Menu ---");
            System.out.println("1. Register Member");
            System.out.println("2. View Members");
            System.out.println("3. Add Trainer");
            System.out.println("4. View Trainers");
            System.out.println("5. Make Payment");
            System.out.println("6. Deactivate Member");
            System.out.println("7. Activate Member");
            System.out.println("8. Exit");
            System.out.print("Choice: ");

            try {
                int choice = Integer.parseInt(sc.nextLine());

                switch (choice) {
                    case 1:
                        registerMember();
                        break;
                    case 2:
                        viewMembers();
                        break;
                    case 3:
                        addTrainer();
                        break;
                    case 4:
                        viewTrainers();
                        break;
```

```
            case 5:
                makePayment();
                break;
            case 6:
                deactivateMember();
                break;
            case 7:
                activateMember();
                break;
            case 8:
                System.out.println("Thanks for using the system!");
                return;
            default:
                System.out.println("Invalid option. Try again.");
          }
        } catch (NumberFormatException e) {
            System.out.println("Enter a number only!");
        } catch (InvalidOperationException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

private static void registerMember() {
    if (memberCount >= MAX_MEMBERS) {
        System.out.println("Can't register more members.");
        return;
    }

    System.out.print("Enter member name: ");
    String name = sc.nextLine();

    System.out.print("Membership Type (Basic/Premium): ");
    String type = sc.nextLine();

    Member m = new Member(memberIdGen++, name, type);
    members[memberCount++] = m;
    m.greet();
    System.out.println("Member added.");
}

private static void viewMembers() {
    if (memberCount == 0) {
        System.out.println("No members yet.");
        return;
    }
```

```
    for (int i = 0; i < memberCount; i++) {
        members[i].displayInfo();
    }
}

private static void addTrainer() {
    if (trainerCount >= MAX_TRAINERS) {
        System.out.println("Can't add more trainers.");
        return;
    }

    System.out.print("Trainer name: ");
    String name = sc.nextLine();

    System.out.print("Specialization: ");
    String spec = sc.nextLine();

    Trainer t = new Trainer(trainerIdGen++, name, spec);
    trainers[trainerCount++] = t;
    t.greet();
    System.out.println("Trainer added.");
}

private static void viewTrainers() {
    if (trainerCount == 0) {
        System.out.println("No trainers added yet.");
        return;
    }

    for (int i = 0; i < trainerCount; i++) {
        trainers[i].displayInfo();
    }
}

private static void makePayment() throws InvalidOperationException {
    if (memberCount == 0) {
        throw new InvalidOperationException("No members registered.");
    }

    System.out.print("Enter Member ID: ");
    int id = Integer.parseInt(sc.nextLine());

    Member m = findMemberById(id);
    if (m == null) {
        throw new InvalidOperationException("Member not found.");
    }
```

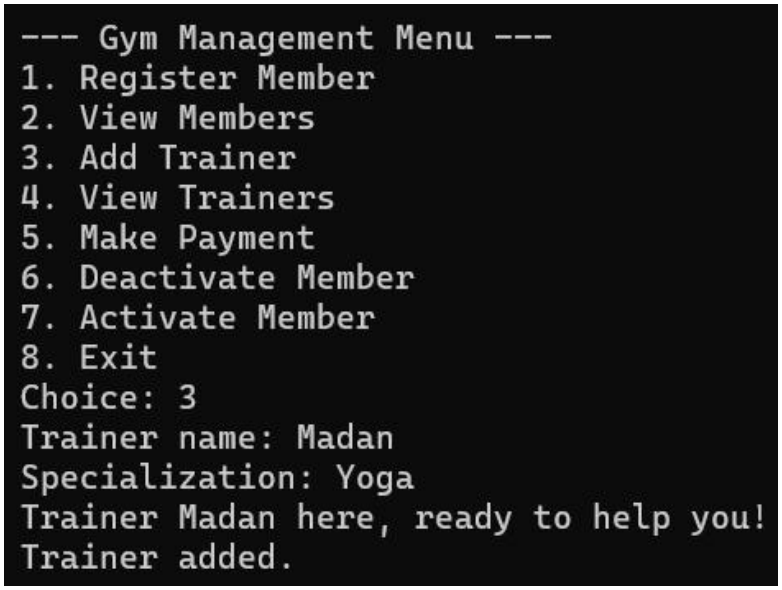
```
        if (!m.isActive()) {
            throw new InvalidOperationException("Member is inactive.");
        }

        double fee = m.getFee();

        System.out.print("Payment Method (Cash/UPI/Card): ");
        String method = sc.nextLine();

        Payment p = new Payment(m, fee, method);
        p.process();
    }

    private static void deactivateMember() {
        System.out.print("Enter Member ID to deactivate: ");
        int id = Integer.parseInt(sc.nextLine());

        Member m = findMemberById(id);
        if (m == null) {
            System.out.println("No such member.");
            return;
        }

        m.deactivate();
        System.out.println("Member deactivated.");
    }

    private static void activateMember() {
        System.out.print("Enter Member ID to activate: ");
        int id = Integer.parseInt(sc.nextLine());

        Member m = findMemberById(id);
        if (m == null) {
            System.out.println("No such member.");
            return;
        }

        if (m.isActive()) {
            System.out.println("Member is already active.");
            return;
        }

        m.activate();
        System.out.println("Member activated successfully.");
    }

    private static Member findMemberById(int id) {
```

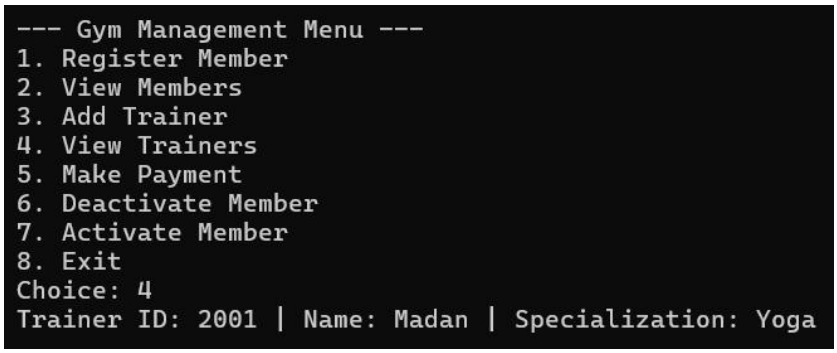
```
        for (int i = 0; i < memberCount; i++) {
            if (members[i].getId() == id) return members[i];
        }
        return null;
    }
}
```

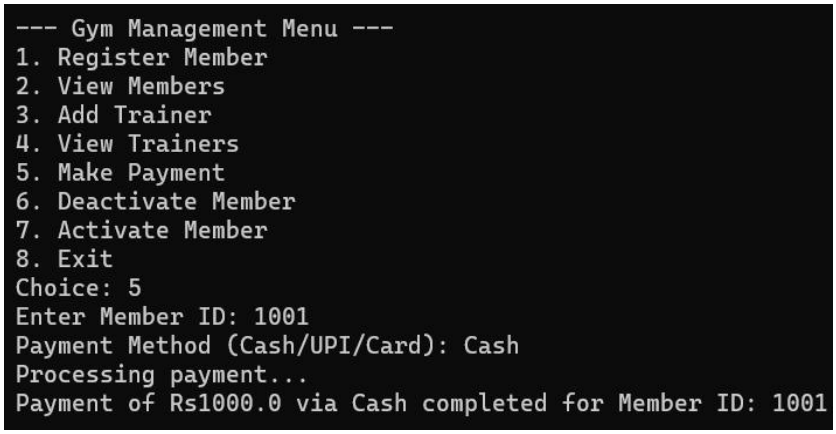
OUTPUT:

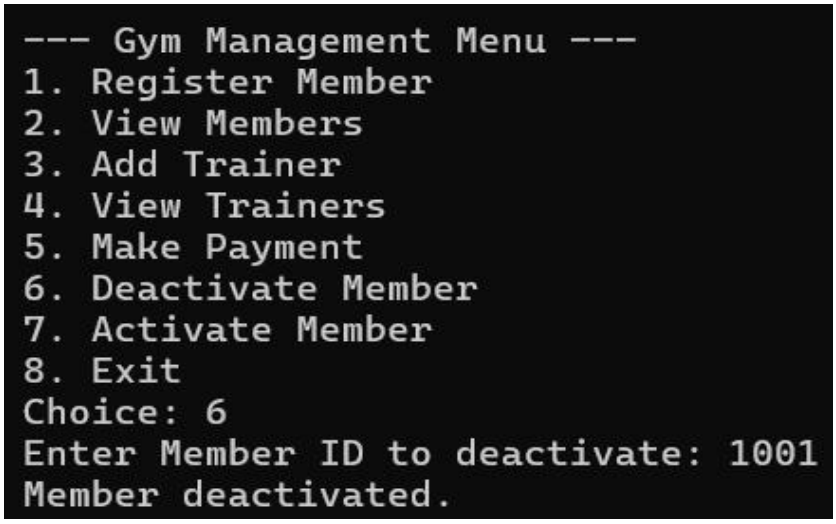
```
B:\java project>javac GymManagementSystem.java

B:\java project>java GymManagementSystem

--- Gym Management Menu ---
1. Register Member
2. View Members
3. Add Trainer
4. View Trainers
5. Make Payment
6. Deactivate Member
7. Activate Member
8. Exit
Choice: 1
Enter member name: Bala
Membership Type (Basic/Premium): Premium
Hi Bala, welcome to the gym!
Member added.
```

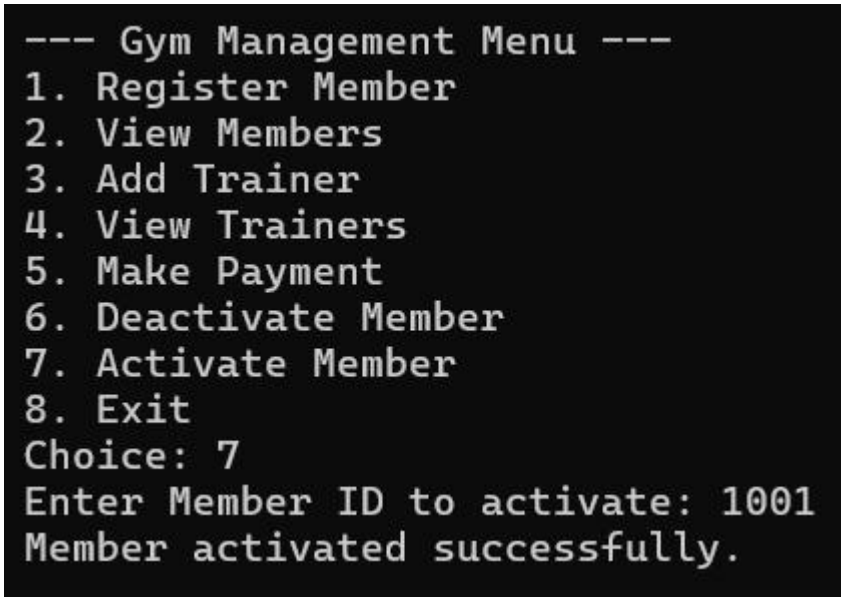
```
--- Gym Management Menu ---
1. Register Member
2. View Members
3. Add Trainer
4. View Trainers
5. Make Payment
6. Deactivate Member
7. Activate Member
8. Exit
Choice: 2
Member ID: 1001 | Name: Bala | Type: Premium | Active: true
```


--- Gym Management Menu ---
1. Register Member
2. View Members
3. Add Trainer
4. View Trainers
5. Make Payment
6. Deactivate Member
7. Activate Member
8. Exit
Choice: 3
Trainer name: Madan
Specialization: Yoga
Trainer Madan here, ready to help you!
Trainer added.


--- Gym Management Menu ---
1. Register Member
2. View Members
3. Add Trainer
4. View Trainers
5. Make Payment
6. Deactivate Member
7. Activate Member
8. Exit
Choice: 4
Trainer ID: 2001 | Name: Madan | Specialization: Yoga


--- Gym Management Menu ---
1. Register Member
2. View Members
3. Add Trainer
4. View Trainers
5. Make Payment
6. Deactivate Member
7. Activate Member
8. Exit
Choice: 5
Enter Member ID: 1001
Payment Method (Cash/UPI/Card): Cash
Processing payment...
Payment of Rs1000.0 via Cash completed for Member ID: 1001





INFERENCE:

The given code implements a basic gym management system using Object-Oriented Programming (OOP) principles in Java, along with exception handling, interfaces, and abstract classes. It allows the management of gym members and trainers, including their registration, payments, activation/deactivation, and information display.