Club Management System Program

```
package ClubManagementSystem;

import java.util.*;
import java.time.*;
import java.io.*;
import java.time.format.DateTimeFormatter;

abstract class User implements Serializable {
        private static final long serialVersionUID = 1L;
    String name, email, contact;
    abstract boolean login(String password);
}

class Participant extends User {

        private List<Event> registeredEvents = new ArrayList<>();

    private String password;
    String studentId;

    void setPassword(Scanner sc) {
        System.out.print("Set Strong Password: ");
        password = sc.nextLine();
    }

    Participant(String name, String contact, String email, String studentId, Scanner sc) {
        this.name = name;
        this.contact = contact;
        this.email = email;
        this.studentId = studentId;
        setPassword(sc);
    }

    boolean login(String password) {
        return this.password.equals(password);
    }

   public void addRegisteredEvent(Event e) {
            try {
                if (e == null) {
                    throw new IllegalArgumentException("Event cannot be null");
                }
```

```
                if (!registeredEvents.contains(e)) {
                    registeredEvents.add(e);
                    System.out.println("Event added successfully.");
                } else {
                    System.out.println("Event already registered.");
                }
            } catch (IllegalArgumentException ex) {
                System.out.println("Error: " + ex.getMessage());
            } catch (Exception ex) {
                System.out.println("An unexpected error occurred while adding the event: " +
ex.getMessage());
            }
        }

        public void listRegisteredEvents() {
            try {
                if (registeredEvents == null) {
                    throw new IllegalStateException("Registered events list is uninitialized.");
                }

                if (registeredEvents.isEmpty()) {
                    System.out.println("No events registered.");
                } else {
                    for (Event e : registeredEvents) {
                        System.out.println(e.eventName + " - Deadline: " + e.deadline);
                    }
                }
            } catch (IllegalStateException ex) {
                System.out.println("Error: " + ex.getMessage());
            } catch (Exception ex) {
                System.out.println("An unexpected error occurred while listing the events: " +
ex.getMessage());
            }
        }
}

class Member extends User {
    private String password;
    private String memberId;

    void setPassword(Scanner sc) {
        System.out.print("Set Strong Password: ");
        password = sc.nextLine();
```

```
    }

    Member(String name, String contact, String email, String memberId, Scanner sc) {
        this.name = name;
        this.contact = contact;
        this.email = email;
        this.memberId = memberId;
        setPassword(sc);
    }

    String getMemberId() {
        return memberId;
    }

    boolean login(String password) {
        return this.password.equals(password);
    }
}

class Event implements Serializable {
        private static final long serialVersionUID = 1L;
    String eventName;
    LocalDate deadline;
    public ArrayList<Participant> participants = new ArrayList<>();
    public ArrayList<String> achievements = new ArrayList<>();

    public Event(String eventName, LocalDate deadline) {
        this.eventName = eventName;
        this.deadline = deadline;
        this.achievements = new ArrayList<>();
    }

    void addAchievement(String achievement) {
        achievements.add(achievement);
    }

    void displayAchievements() {
        if (achievements.isEmpty()) {
            System.out.println("No achievements yet.");
        } else {
            for (String ach : achievements) {
                System.out.println("- " + ach);
            }
        }
```

```
    }

    void displayEventInfo() {
        System.out.println("Event: " + eventName);
        System.out.println("Deadline: " +
deadline.format(DateTimeFormatter.ofPattern("yyyy-MM-dd")));
    }

    public void registerParticipant(Participant p) {
        try {
            if (p == null) {
                throw new IllegalArgumentException("Participant cannot be null.");
            }
            if (!participants.contains(p)) {
                participants.add(p);
                p.addRegisteredEvent(this); // Track it in participant too
                System.out.println("Participant registered successfully.");
            } else {
                System.out.println("Participant already registered for this event.");
            }
        } catch (IllegalArgumentException ex) {
            System.out.println("Error: " + ex.getMessage());
        }
    }

    public boolean hasParticipant(String studentId) {
        try {
            if (studentId == null || studentId.isEmpty()) {
                throw new IllegalArgumentException("Student ID cannot be null or empty.");
            }
            for (Participant p : participants) {
                if (p.studentId.equals(studentId)) {
                    return true;
                }
            }
            return false;
        } catch (IllegalArgumentException ex) {
            System.out.println("Error: " + ex.getMessage());
            return false;
        }
    }

    public boolean isUpcoming() {
        return deadline.isAfter(LocalDate.now()) || deadline.isEqual(LocalDate.now());
```

```
    }

    public boolean isPast() {
        return deadline.isBefore(LocalDate.now());
    }

}


class Club implements Serializable {
        private static final long serialVersionUID = 1L;
    String clubName, clubCreatorName, clubCreatorPost,clubMotive;
    ArrayList<Member> members = new ArrayList<>();
    ArrayList<Event> events = new ArrayList<>();

    Club(String clubName, String clubCreatorName, String clubCreatorPost,String clubMotive) {
        this.clubName = clubName;
        this.clubCreatorName = clubCreatorName;
        this.clubCreatorPost = clubCreatorPost;
        this.clubMotive = clubMotive;
    }

    void addMember(Member m) {
        try {
            if (m == null) {
                throw new IllegalArgumentException("Member cannot be null.");
            }
            members.add(m);
            System.out.println("Member added successfully.");
        } catch (IllegalArgumentException ex) {
            System.out.println("Error: " + ex.getMessage());
        }
    }

    void addEvent(Event e) {
        try {
            if (e == null) {
                throw new IllegalArgumentException("Event cannot be null.");
            }
            events.add(e);
            System.out.println("Event added successfully.");
        } catch (IllegalArgumentException ex) {
            System.out.println("Error: " + ex.getMessage());
        }
```

```
}

void displayClubInfo() {
    System.out.println("Club: " + clubName);
    System.out.println("Created by: " + clubCreatorName + " (" + clubCreatorPost + ")");
    System.out.println("Motive: " + clubMotive);
}

void displayEvents() {
    try {
        if (events == null) {
            throw new IllegalStateException("Events list is uninitialized.");
        }

        if (events.isEmpty()) {
            System.out.println("No events yet.");
            return;
        }
        int i = 1;
        for (Event e : events) {
            System.out.print(i++ + ". ");
            e.displayEventInfo();
        }
    } catch (IllegalStateException ex) {
        System.out.println("Error: " + ex.getMessage());
    }
}

void displayPastEvents() {
    try {
        if (events == null || events.isEmpty()) {
            System.out.println("No events available.");
            return;
        }

        boolean found = false;
        for (Event e : events) {
            if (e.deadline.isBefore(LocalDate.now())) {
                System.out.println(" 🔹 " + e.eventName + " | Deadline: " + e.deadline);
                found = true;
            }
        }

        if (!found) {
```

```
            System.out.println("No past events found.");
        }
    } catch (Exception ex) {
        System.out.println("Error while displaying past events: " + ex.getMessage());
    }
  }


}


public class ClubManagementSystem {

    private static final String CLUB_FILE = "clubs.ser";
    private static final String PARTICIPANT_FILE = "participants.ser";
    private static final String MEMBER_FILE = "members.ser";

    @SuppressWarnings("unchecked")
    private static void loadData(ArrayList<Club> clubList, HashMap<String, Participant>
participantMap, HashMap<String, Member> memberMap) {
        try (
            ObjectInputStream ois1 = new ObjectInputStream(new
FileInputStream(CLUB_FILE));
            ObjectInputStream ois2 = new ObjectInputStream(new
FileInputStream(PARTICIPANT_FILE));
            ObjectInputStream ois3 = new ObjectInputStream(new
FileInputStream(MEMBER_FILE))
        ) {
            clubList.addAll((ArrayList<Club>) ois1.readObject());
            participantMap.putAll((HashMap<String, Participant>) ois2.readObject());
            memberMap.putAll((HashMap<String, Member>) ois3.readObject());
            System.out.println("Welcome!!!");
        } catch (Exception e) {
            System.out.println("No previous data found or failed to load.");
        }
    }

    private static void saveData(ArrayList<Club> clubList, HashMap<String, Participant>
participantMap, HashMap<String, Member> memberMap) {
        try (
            ObjectOutputStream oos1 = new ObjectOutputStream(new
FileOutputStream(CLUB_FILE));
            ObjectOutputStream oos2 = new ObjectOutputStream(new
FileOutputStream(PARTICIPANT_FILE));
```

```
            ObjectOutputStream oos3 = new ObjectOutputStream(new
FileOutputStream(MEMBER_FILE))
        ) {
            oos1.writeObject(clubList);
            oos2.writeObject(participantMap);
            oos3.writeObject(memberMap);
            System.out.println("Data saved successfully.");
        } catch (IOException e) {
            System.out.println("Error saving data: " + e.getMessage());
        }
    }

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    ArrayList<Club> clubList = new ArrayList<>();
    HashMap<String, Participant> participantMap = new HashMap<>();
    HashMap<String, Member> memberMap = new HashMap<>();
    loadData(clubList, participantMap, memberMap);

    while (true) {
        System.out.println("\n----- CLUB MANAGEMENT SYSTEM -----");
        System.out.println("1. Explore Clubs");
        System.out.println("2. Register New Club");
        System.out.println("3. Login as Participant");
        System.out.println("4. Create Student ID");
        System.out.println("5. Login as Member");
        System.out.println("6. Create Member ID");
        System.out.println("7. Exit");
        System.out.print("Enter your choice: ");
        int choice = -1;
        try {
            choice = Integer.parseInt(sc.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter a number.");
            continue;
        }

        switch (choice) {
        case 1:
            if (clubList.isEmpty()) {
                System.out.println("No clubs registered yet.");
                break;
            }
            // Display list of clubs
```

```
for (int i = 0; i < clubList.size(); i++) {
    System.out.println((i + 1) + ". " + clubList.get(i).clubName);
}
int selected = -1;
boolean validSelection = false;
while (!validSelection) {
    System.out.print("Select club number to explore: ");
    try {
        selected = Integer.parseInt(sc.nextLine());
        if (selected < 1 || selected > clubList.size()) {
            System.out.println("Invalid club selection. Please select a valid number.");
        } else {
            validSelection = true; // Exit the loop if the selection is valid
        }
    } catch (NumberFormatException e) {
        System.out.println("Invalid input! Please enter a valid number.");
    }
}

Club club = clubList.get(selected - 1); // Get the selected club
boolean back = false;
do {
    System.out.println("\n--- Exploring " + club.clubName + " ---");
    System.out.println("1. Club Info");
    System.out.println("2. Events");
    System.out.println("3. Achievements");
    System.out.println("4. Back");

    int sub = -1;
    boolean validSubChoice = false;
    while (!validSubChoice) {
        System.out.print("Choice: ");
        try {
            sub = Integer.parseInt(sc.nextLine());
            if (sub < 1 || sub > 4) {
                System.out.println("Invalid choice. Please select a valid option (1-4).");
            } else {
                validSubChoice = true; // Exit the loop if the sub-choice is valid
            }
        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter a valid number.");
        }
    }
```

```
        switch (sub) {
            case 1:
                club.displayClubInfo();
                break;
            case 2:
                club.displayEvents();
                break;
            case 3:
                if (club.events.isEmpty()) {
                    System.out.println("No events available in this club.");
                } else {
                    for (Event e : club.events) {
                        System.out.println("Achievements in " + e.eventName + ":");
                        e.displayAchievements();
                    }
                }
                break;
            case 4:
                back = true;
                break;
            default:
                System.out.println("Invalid choice.");
        }
    } while (!back);
    break;


case 2:
    String clubName = "", creatorName = "", creatorPost = "",clubMotive ="";
    boolean validInput = false;

    // Loop until all inputs are valid
    while (!validInput) {
        try {
            System.out.print("Enter Club Name: ");
            clubName = sc.nextLine().trim();
            if (clubName.isEmpty()) {
                throw new IllegalArgumentException("Club name cannot be empty.");
            }

            System.out.print("Enter Creator Name: ");
            creatorName = sc.nextLine().trim();
            if (creatorName.isEmpty()) {
                throw new IllegalArgumentException("Creator name cannot be empty.");
```

```
            }

            System.out.print("Enter Your Designation/Post: ");
            creatorPost = sc.nextLine().trim();
            if (creatorPost.isEmpty()) {
                throw new IllegalArgumentException("Creator designation cannot be empty.");
            }

            System.out.print("Enter Your Motive: ");
            clubMotive = sc.nextLine().trim();
            if (clubMotive.isEmpty()) {
                throw new IllegalArgumentException("Club Motive cannot be empty.");
            }

            // If all inputs are valid, proceed
            validInput = true;
            clubList.add(new Club(clubName, creatorName, creatorPost,clubMotive));
            System.out.println("Club '" + clubName + "' registered successfully.");

        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
    break;

case 3:
    System.out.print("Enter Student ID: ");
    String sid = sc.nextLine();
    if (!participantMap.containsKey(sid)) {
        System.out.println("Student ID not found.");
        break;
    }

    Participant p = participantMap.get(sid);
    System.out.print("Enter Password: ");
    if (!p.login(sc.nextLine())) {
        System.out.println("Incorrect password.");
        break;
    }

    boolean logoutP = false;
    do {
        System.out.println("\n--- Welcome " + p.name + " ---");
        System.out.println("1. View Registered Events");
```

```
                System.out.println("2. View Upcoming Events");
                System.out.println("3. Register for Event");
                System.out.println("4. View Past Events");
                System.out.println("5. Logout");
                System.out.print("Choice: ");
                int op = -1;
                try {
                    op = Integer.parseInt(sc.nextLine());
                } catch (NumberFormatException e) {
                    System.out.println("Please enter a valid number.");
                    continue;
                }

                switch (op) {
                    case 1:
                        System.out.println("Your Registered Events:");
                        p.listRegisteredEvents();
                        break;

                    case 2:
                        for (Club c : clubList) {
                            for (Event e : c.events) {
                                if (e.isUpcoming()) {
                                    System.out.println("- " + e.eventName + " (" + c.clubName + ") -
Deadline: " + e.deadline);
                                }
                            }
                        }
                        break;

                    case 3:
                        if (clubList.isEmpty()) {
                            System.out.println("No clubs available.");
                            break;
                        }

                        for (int i = 0; i < clubList.size(); i++) {
                            System.out.println((i + 1) + ". " + clubList.get(i).clubName);
                        }

                        int cid = -1;
                        try {
                            System.out.print("Select club number: ");
                            cid = Integer.parseInt(sc.nextLine());
```

```
} catch (NumberFormatException e) {
    System.out.println("Invalid input. Enter a number.");
    break;
}

if (cid < 1 || cid > clubList.size()) {
    System.out.println("Invalid club.");
    break;
}

Club regClub = clubList.get(cid - 1);
regClub.displayEvents();

int eid = -1;
try {
    System.out.print("Select event number: ");
    eid = Integer.parseInt(sc.nextLine());
} catch (NumberFormatException e) {
    System.out.println("Invalid input. Enter a number.");
    break;
}

if (eid < 1 || eid > regClub.events.size()) {
    System.out.println("Invalid event.");
    break;
}

Event selectedEvent = regClub.events.get(eid - 1);

// Check if the participant is already registered for the event
boolean alreadyRegistered = false;
for (Participant participant : selectedEvent.participants) {
    if (participant.equals(p)) {
        alreadyRegistered = true;
        break;
    }
}

if (alreadyRegistered) {
    System.out.println("You are already registered for this event.");
    break;
}

// Register the participant
```

```
                    selectedEvent.registerParticipant(p);
                    System.out.println("Registered successfully.");
                    break;


                case 4:
                    for (Club c : clubList) {
                        for (Event e : c.events) {
                            if (e.isPast()) {
                                System.out.println("- " + e.eventName + " (" + c.clubName + ") -
Deadline: " + e.deadline);
                            }
                        }
                    }
                    break;

                case 5:
                    logoutP = true;
                    break;

                default:
                    System.out.println("Invalid option.");
                }
            } while (!logoutP);
            break;


        case 4:
            System.out.print("Enter Name: ");
            String sname = sc.nextLine().trim();

            System.out.print("Enter Contact (10 digits): ");
            String scontact = sc.nextLine().trim();

            System.out.print("Enter Email: ");
            String semail = sc.nextLine().trim();

            System.out.print("Create Student ID: ");
            String studentId = sc.nextLine().trim();

            // Basic validation
            if (sname.isEmpty() || scontact.isEmpty() || semail.isEmpty() || studentId.isEmpty()) {
                System.out.println("⚠ Please ensure all fields are filled.");
                break;
```

```
            }

            if (!scontact.matches("\\d{10}")) {
                System.out.println("⚠️ Invalid contact number. Must be 10 digits.");
                break;
            }

            if (!semail.matches("^\\S+@\\S+\\.\\S+$")) {
                System.out.println("⚠️ Invalid email format.");
                break;
            }

            if (participantMap.containsKey(studentId)) {
                System.out.println("⚠️ Student ID already exists.");
                break;
            }

            Participant newP = new Participant(sname, scontact, semail, studentId, sc);
            participantMap.put(studentId, newP);
            System.out.println(" Student ID created successfully! You can now log in as a
participant.");
            break;
        case 5:
            System.out.print("Enter Member ID: ");

            String mid = sc.nextLine().trim();
            if (!memberMap.containsKey(mid)) {
                System.out.println("Member ID not found.");
                break;
            }
            Member m = memberMap.get(mid);
            System.out.print("Enter Password: ");
            if (!m.login(sc.nextLine())) {
                System.out.println("Incorrect password.");
                break;
            }

            // Identify the club where this member exists
            Club myClub = null;
            for (Club c : clubList) {
                if (c.members.contains(m)) {
                    myClub = c;
                    break;
                }
```

```
}

if (myClub == null) {
    System.out.println("⚠️ You are not added to any club.");
    break;
}

boolean logoutM = false;
do {
    System.out.println("\n--- Welcome " + m.name + " ---");
    System.out.println("1. Add Event");
    System.out.println("2. Add Achievement");
    System.out.println("3. View Club All Events");
    System.out.println("4. Logout");
    System.out.print("Choice: ");
    int opt = Integer.parseInt(sc.nextLine().trim());

    switch (opt) {
        case 1:
            // Only club creator can add events

            System.out.print("Enter Event Name: ");
            String ename = sc.nextLine();
            System.out.print("Enter Deadline (yyyy-mm-dd): ");
            LocalDate deadline = LocalDate.parse(sc.nextLine().trim());
            myClub.addEvent(new Event(ename, deadline));
            System.out.println("✅ Event added successfully.");
            break;

        case 2:
            if (myClub.events.isEmpty()) {
                System.out.println("⚠️ No events found. Add events first.");
                break;
            }
            System.out.println("Your Club Events:");
            for (int i = 0; i < myClub.events.size(); i++) {
                System.out.println((i + 1) + ". " + myClub.events.get(i).eventName);
            }
            System.out.print("Select event number: ");
            int eIdx = Integer.parseInt(sc.nextLine());
            if (eIdx < 1 || eIdx > myClub.events.size()) {
                System.out.println("❌ Invalid choice.");
                break;
            }
```

```
                Event chosen = myClub.events.get(eIdx - 1);
                System.out.print("Enter Achievement: ");
                String ach = sc.nextLine();
                chosen.addAchievement(ach);
                System.out.println("🏆 Achievement added.");
                break;

            case 3:
                if (myClub.events.isEmpty()) {
                    System.out.println("No events added yet.");
                } else {
                    myClub.displayEvents();
                }
                break;

            case 4:
                logoutM = true;
                break;

            default:
                System.out.println("Invalid option.");
        }
    } while (!logoutM);
    break;

case 6:
{
    try {
        if (clubList.isEmpty()) {
            System.out.println("No clubs available.");
            break;
        }

        // Display available clubs
        System.out.println("Available Clubs:");
        for (int i = 0; i < clubList.size(); i++) {
            System.out.println((i + 1) + ". " + clubList.get(i).clubName);
        }

        // Get user's club choice
        int joinChoice = -1;
        while (joinChoice < 1 || joinChoice > clubList.size()) {
            System.out.print("Enter club number to join: ");
            try {
```

```
                joinChoice = Integer.parseInt(sc.nextLine().trim());
                if (joinChoice < 1 || joinChoice > clubList.size()) {
                    System.out.println("Invalid club number. Please try again.");
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a valid number.");
            }
        }

        // Get the selected club
        Club selectedClub = clubList.get(joinChoice - 1);

        // Ask for Member details (Student ID, name, etc.)
        System.out.print("Enter your Member ID: ");
        String memberId = sc.nextLine().trim();
        System.out.print("Enter your Name: ");
        String memberName = sc.nextLine().trim();
        System.out.print("Enter your Contact: ");
        String memberContact = sc.nextLine().trim();
        System.out.print("Enter your Email: ");
        String memberEmail = sc.nextLine().trim();

        // Check if Member with the given memberId already exists in the Club
        boolean isAlreadyMember = false;
        for (Member m1 : selectedClub.members) {
            if (m1.getMemberId().equals(memberId)) {
                isAlreadyMember = true;
                break;
            }
        }

        if (isAlreadyMember) {
            System.out.println("You are already a member of this club.");
        } else {
            // Create new Member and add to Club
            Member newMember = new Member(memberName, memberContact,
memberEmail, memberId, sc);
            selectedClub.addMember(newMember);
            memberMap.put(memberId, newMember);
            System.out.println("You've successfully joined the club: " +
selectedClub.clubName);
        }

    } catch (Exception e) {
```

```
                System.out.println("An error occurred while trying to join the club: " +
e.getMessage());
            }
            break;
        }
  case 7:
            System.out.println("Exiting system. Goodbye!");
            saveData(clubList, participantMap, memberMap);
            return;

        default:
            System.out.println("Invalid choice.");
        }
    }
  }
}
```