CSE2DES/CSE5DES – Assignment – Part 2

This part is worth 50% of the total Assignment Mark

Due Date: 9.30 am, Thursday 24th October 2013

Where to Submit: A hardcopy of completed design class diagrams, atomic use case specifications and Java code is to be submitted in the labelled submission box behind BG139. You must also submit an electronic version of your code via latcs6 using the submit command:

```
submit DES <directory or file name>
```

You will be required to demonstrate your program and answer questions in your week 13 lab class.

This is an individual assignment. You are not permitted to work as a group when writing this assignment.

Copying, Plagiarism: Plagiarism is the submission of somebody else's work in a manner that gives the impression that the work is your own. The Department of Computer Science and Computer Engineering treats plagiarism very seriously. When it is detected, penalties are strictly imposed. Students are referred to the Department of Computer Science and Computer Engineering's Handbook and policy documents with regard to plagiarism.

No extensions will be given: Penalties are applied to late assignments (5% of total assignment mark given is deducted per day, accepted up to 4 days after the due date only). If there are circumstances that prevent the assignment being submitted on time, an application for special consideration may be made. See the departmental Student Handbook for details. Note that delays caused by computer downtime cannot be accepted as a valid reason for a late submission without penalty. Students must plan their work to allow for both scheduled and unscheduled downtime.

Return of Assignments: Assignments are to be returned within three weeks from the last day on which the assignment can be submitted.

Objectives: To learn, given a design class diagram, to specify atomic use cases, and implement and test a prototype of the system.

For Part 2 of the assignment, you are to continue with the Gymnastics System described in Part 1. Whereas Part 1 is concerned with the analysis phase, Part 2 will be concerned with design, prototyping and testing.

As the starting point for Part 2, assume that the following design class diagrams have been adopted. Figure 1 shows the classes representing the domain objects, and Figure 2 shows the system class.

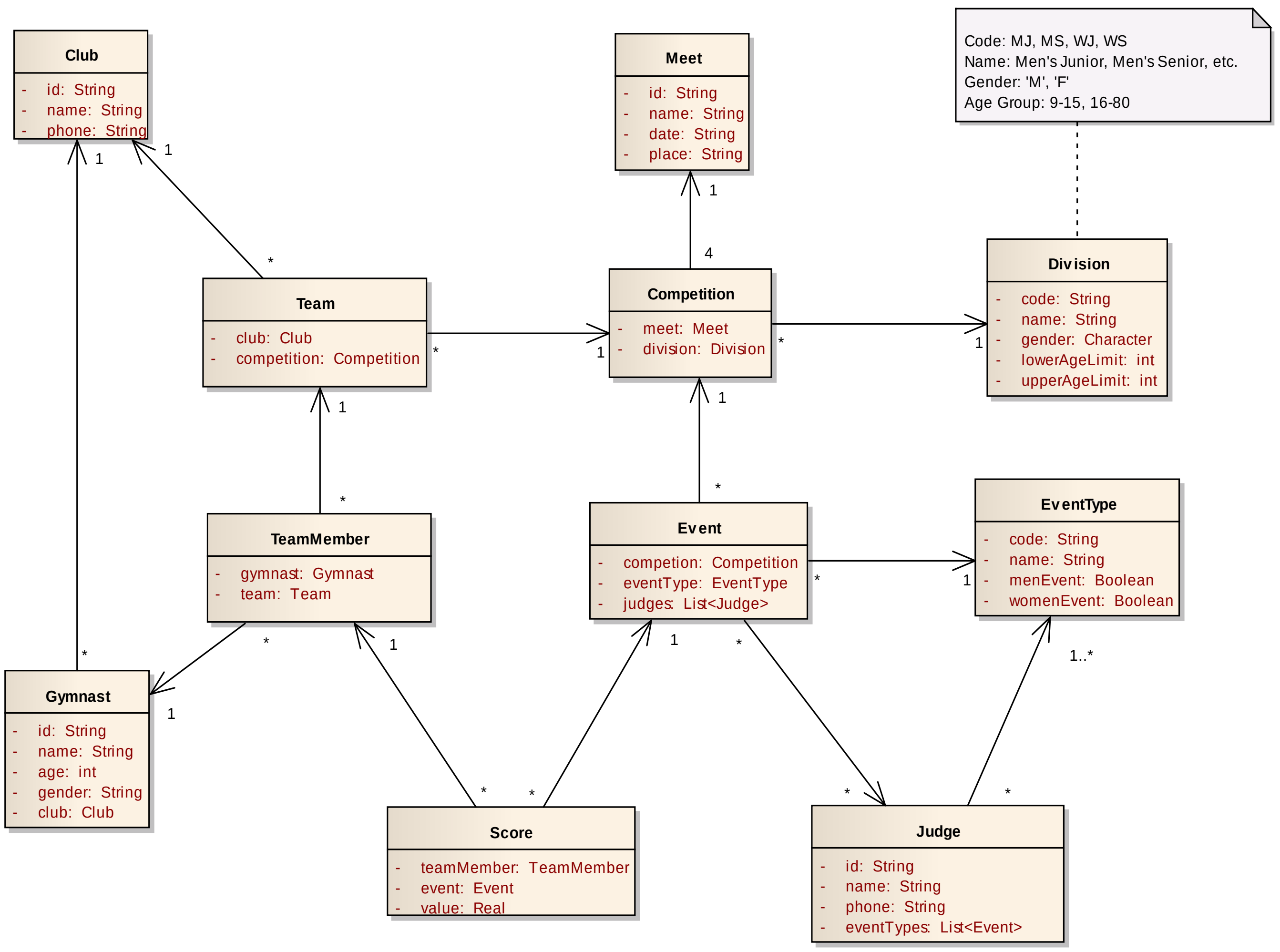

Figure 1 – Design Class Diagram Showing Classes for Domain Objects and Their Attributes

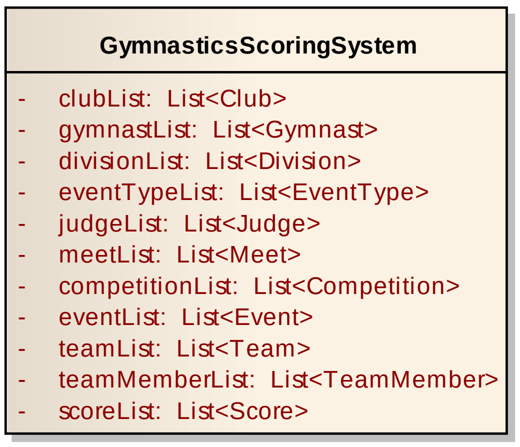

Figure 2 – The System Class – With Attributes

Task 1 – Atomic Use Case Specifications (40%)

Formally specify the following atomic use cases:

1. Add a club
2. Add a gymnast
3. Add a division
4. Add an event type
5. Add a judge
6. Add a meet (and the competitions for the meet)
7. Add an event for a competition
8. Assign a judge to an event (in a competition)
9. Register a team
10. Register a member for a team
11. Enter the score for a gymnast (member of a team) for an event
12. Generate a report on the performance of a gymnast in a meet (**optional**)

Note 1:

– In your answer, **number the use cases (from 1 to 12)** as shown above.

– Your specifications **must be based on the design class diagrams given.**

– If necessary, refer to the problem statement given in Part 1 of the assignment.

Note 2: You are required to specify 11 atomic use cases (not counting use case 12). But about five atomic use cases will be selected for marking.

What to submit for Task 2

1. Hardcopy of the atomic use case specifications.
2. Electronic copy of the atomic use case specification

Task 2 – Prototyping (40%)

Prototype all the atomic use cases listed for Task 1 in Java. (Use case 12 is of course optional.)

Your implementation of the prototype must be done in a systematic manner. In particular, for each use case, the preconditions should be checked first, and the postconditions should then be realized.

Note 1: You should use the Helper class given in the labs to search a collection on the basis of a simple or composite key.

Note 2: The Appendix provides a sample "quick-and-dirty" test program. You can use the tests given in this program as a means to quickly test your in-progress implementation. **Your implementation must be such that we can run this test program without any change.** This means in particular, the method signatures of the system class have to be compatible.

Note 3: Similarly to the previous task, about five atomic use cases will be selected for marking.

What to submit for Task 2

1. Hardcopy of the listing of the code. Arrange your classes as follows.

   The first class in the listing is the GymnasticsScoringSystem class, followed by the "domain" classes in *alphabetical order* of the class names.

2. Electronic copy of all the classes.

Task 3 – Testing the Prototype (20%)

For each atomic use case, design the test cases and include them in a Java program, called GymnasticsScoringSystemTester, to carry out the testing.

Organizing Test Cases

1. Arrange the testing in a number of test methods

2. Each method is to test one atomic use case only. An atomic use case must have at least two test methods: one to test the successful cases; and one (at least) to test the invalid request.

3. The test methods must be **named** as follows: use case number, followed by test number for that use case. For example: **`UC01Test01()`**

4. Each test method must be **commented** to indicate (a) Whether it is for successful cases or unsuccessfully cases; and (b) for unsuccessful cases, the reasons for them to be unsuccessful.

Note: The sample test program given in the Appendix is only for a quick test. Your test program required for this task must be organized into a number of methods as described above.

Note: You should perform the testing for each atomic use case as soon as you finish implement it.

What to submit for Task 3

1. Hardcopy of the code listing of the GymnasticsScoringSystemTester class.
2. Electronic copy of the GymnasticsScoringSystemTester class.
3. Electronic copy of the test result.

Appendix – Quick Test Program

```
import java.util.*;
public class GymnasticsScoringSystemQuickTester
{
  public static void main(String [] args) throws Exception
  {
    int test = 1;

    // new sytem
    System.out.println("...Test " + (test++) + ": Create gymnastic system");

    GymnasticsScoringSystem gss = new GymnasticsScoringSystem();
    System.out.println( gss );

    // 1. add clubs
    System.out.println("...Test " + (test++) + ": Add clubs");

    gss.addClub("CLUB-10", "Acrobats", "1010");
    System.out.println( gss );

    // 2. add gymnasts
    System.out.println("...Test " + (test++) + ": Add gymnasts");

    gss.addGymnast("GYMNAST-10", "Smith", 'M', 20, "CLUB-10");
    System.out.println( gss );

    // 3. add divisions
    System.out.println("...Test " + (test++) + ": Add divisions");
    gss.addDivision("MJ", "Mens Juniors", 'M', 10, 15);
    gss.addDivision("MS", "Mens Seniors", 'M', 16, 80);
    gss.addDivision("WJ", "Womens Juniors", 'F', 10, 15);
    gss.addDivision("WS", "Womens seniors", 'F', 16, 80);
    System.out.println( gss );

    // 4. add 3 event types
    System.out.println("...Test " + (test++) + ": Add event types");
    gss.addEventType("EVENT-TYPE-10", "floor", true, true);
    gss.addEventType("EVENT-TYPE-20", "bar", true, false);
    gss.addEventType("EVENT-TYPE-30", "beam", false, true);
    System.out.println( gss );

    // 5. add judges
    System.out.println("...Test " + (test++) + ": Add judges");

    List<String> eventTypeCodes = new ArrayList<String>();
    eventTypeCodes.add("EVENT-TYPE-10");
    eventTypeCodes.add("EVENT-TYPE-20");

    gss.addJudge("JUDGE-10", "Adams", "1111", eventTypeCodes);
    System.out.println( gss );
    // 6. add meets
    System.out.println("...Test " + (test++) + ": Add meets");

    gss.addMeet("MEET-10", "Town Hall 10", "Town Hall", 10);
    System.out.println( gss );

    // 7. add events
    System.out.println("...Test " + (test++) + ": Add events");

    gss.addEvent("MEET-10", "MS", "EVENT-TYPE-10");
    System.out.println( gss );

    // 8. assign judge to event
     System.out.println("...Test " + (test++) + ": Add events");

     gss.assignJudge("JUDGE-10", "MEET-10", "MS", "EVENT-TYPE-10");

    // 9. add teams
    System.out.println("...Test " + (test++) + ": Add teams");

    gss.addteam("CLUB-10", "MEET-10", "MS");
    System.out.println( gss );

    // 10. add team members
    System.out.println("...Test " + (test++) + ": Add members");

    gss.addTeamMember("GYMNAST-10", "MEET-10");
        // CLUB-10 and MS (division) can be computed
    System.out.println( gss );

    // 11. add scores
    System.out.println("...Test " + (test++) + ": Add scores");

    gss.addScore("GYMNAST-10", "MEET-10", "EVENT-TYPE-10", 100);
        // CLUB-10, MS (division code) can be deduced
    System.out.println( gss );

  }
}
```

■