Swimming Competition

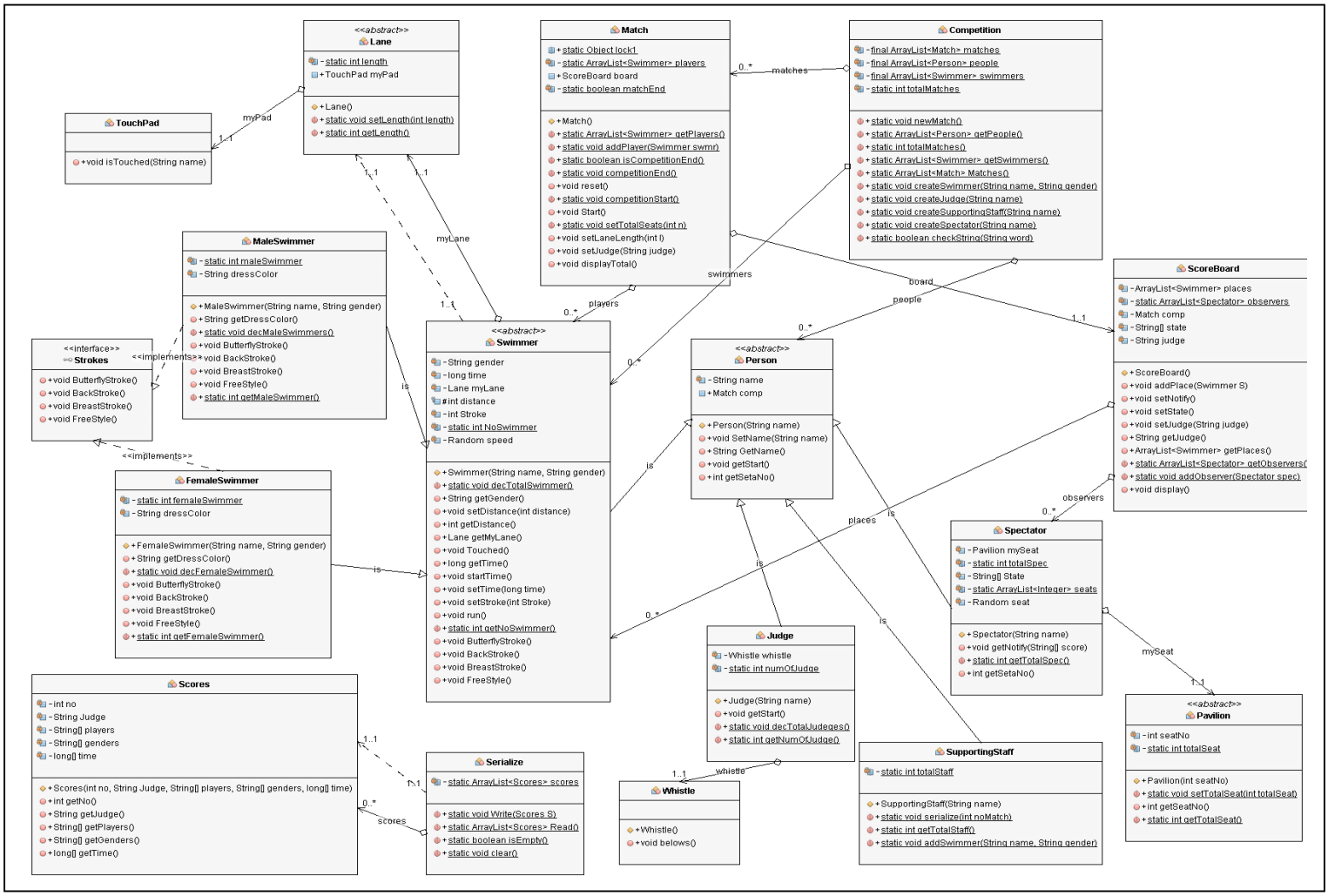
Semester project report (2015)

CS2012 – Principal of Object Oriented Programming

Name: M.J.M Anees

Index Number: 140033P

Class diagram:



Implementation of OOP concepts:

1. **Inheritance:**

Generally Swimmers, judges, supporting staffs, spectators all of them are humans. All of them are implemented using separate classes. To implement the Human, all the classes are extended from the class person. Attribute name is declared only inside the person class but all the subclasses have the attribute names since they are inherited from person.

Classes:

- SuportingStaff
- Swimmer
- Judge
- Spectator

If we consider the Swimmers there are male swimmers and female swimmers. To implement this there is separate classes created for male swimmer and female swimmer called “MaleSwimmer” & “FemaleSwimmer”. But as a swimmer almost all attributes and methods between them are equal. So to imply that in the simulation a parent class is created for male simmer ad female swimmer call “Swimmer”.

Now Person, Judge, SupportinStaff, Spectator, Swimmer, MaleSwimmer, FemaleSwimmer are main classes. Also when saving the score in persistence storage I’m saving the Score object. But u can only save a serializable object. Score class implies serializable interface.

2. **Abstraction:**

In this simulation we should conceder the gender difference between swimmers only. Other factors like age, country... etc. Are not relevant. Gender of the other people also not relevant. So I ignored those factors in my simulation.

In this simulation it is obvious that if there is a person exists he should be one of Swimmer, supporting staff, spectator, and judge. So a person object shouldn’t be created from person class. To make sure I have made person class an abstract class.

Like that I also made Swimmer class an abstract class so there cannot be any swimmers neither Male nor Female created.

When implementing the swimming strokes male swimmer and female swimmer will implement it little differently but the stroke they do same. So to implement the scenario I made Stroke interface with 4 abstract methods. Those are over ride in MaleSwimmer and FemaleSwimmer classes. It can be seen in my program code.

3. **Encapsulation:**

Encapsulation is all about hiding a data. User will never know how the data is stored inside an object. They will only get the necessary data. To implement this I have made all the fields private and protected as much as possible.

(Competition class: line 8 - 11)

(Swimmer class: line **9** - 16)

(Spectator class: line 5, 6, 7)

(Scoreboard class: 8 - 12)

Some methods that I used in the same class those are also made private. Also some of the fields are made protected in order to make only the inherited classes can access.

(Swimmer class: line 12)

(Scoreboard class: line 36)

When storing the score in scores it scores the names, times, gender in the sorted order of place in separate three Arrays to serialize. With no modifier, so no one accept in the package cannot access.

(Scores class all variables are private and final)

4. **Polymorphism:**

All the people created in the competition is store in an Array List type of Person, since every supporting staff, spectators, swimmers and judges are person. Also all the Swimmers are stored in another Array list. When storing in the array list they will act as a person or swimmer. Even in Swimmer Array List doesn't matter whether it is a male swimmer or female swimmer, when storing in the Array List the will act as swimmer.

(Competition class: line 8, 9, 10)

Since all the objects created in java is inherited from the class Object, when storing the Array List of Score to a persistence storage I store it as an object and when reading I downcast it to Array List of Score.

When asking swimmers to do deferent Strokes since they are implemented from the abstract interface Stroke, they both have the same method name for all 4 stroke, but by override the methods implies the strokes different from each other's.

(MaleSwimmer &Femaleswimmer classes: line 21 – 43)

Implementation of Concurrency control:

In the swimming pool simulation I have to use threads in order to make multiple swimmers to swim in a given period of time. Whenever I want to start swimming competition I call the override run method to process. But by starting the thread there will be starvation problems such as one swimmer gets chance to execute many of steps in run method while another one is still haven't a chance to execute even a single step. So to avoid this problem I have set the threads to sleep for a short period so other swimmer thread get chances to run.

(Swimmer class: line 67 - 79)

By changing the sleeping time of the thread we can control the speed of the thread so by Random number selection I change the speed of the swimmer. Also when it comes to implement deferent stroke there must be a different in the speed should be shown. So I control it by changing the random number selection limit.

(Swimmer class: line 62)

I start a new thread by passing the runnable object as a parameter every time I start a new event.

When each swimmer finishes they touch a touch pad and set their place. By changing a static filed place is been given to swimmer. When it comes to multiple threads changing a single variable there will be thread interferes error. So to prevent I have defined a lock and synchronized method.

Match class: line 10,34

To update JSlider according to a given swimmer's traveled distance we should continuously request for the distance traveled. So to do that I have used Swing timer provided by java. It creates an event for a given period of time (1ms) and inside the event we can increase the position of the jSlider.

In edition most of the processes in visualizing of this application are based on graphical user interface. I have used event dispatching thread as it is default thread in java.