**C# Implementation of Skating Events**

**REGISTRATION SUBSYSTEM**

In the registration class, there is going to be a List of type Team. That way all of the information per team will be housed in this list and will be therefore sent to the database. With thing being done, our arrow diagram has changed from athlete being a part of registration (aggregation) to athlete being a part of team (composition). **<<USES>> TEAM**

In the team class, there is a function to get and set the team's name, which will ultimately be their country name. **YOU TWO CHOOSE 12 COUNTRIES. I DON'T CARE IF YOU CHOOSE DJIBOUTI, AZERBAIJAN, BOSNIA AND HERZEGOVINA, ETC or USA, CANADA, JAPAN.** There will be an integer that will have the team's size to ensure that there will be no excess athletes per team. There is a function to add an athlete at our desire as well as to delete an athlete**. We will implement other restrictions down the road.** **<<USES>> ATHLETE**

In the athlete class, there are functions to get and set:

* First Name
* Last Name
* Gender

There will also be a function to get a schedule (which will house the events they partake in). There is a Tuple to represent the athlete's medal count. Now, the medals are NOT A CLASS OF THEIR OWN! This is merely the Skating portion of the Winter Olympics. The Olympics handles the medals and Guercio won't change my mind. So, we won't mention it until presenting. **<<USES>> SCHEDULE**

**REGISTRATION SUBSYSTEM**

**DEPRECATED X X But still on the class diagram because I want to show**

**O what connects to it.**

**<<USES>> Registration**

**<<USES>> Score**

**<<USES>> Event**

**JUDGE SUBSYSTEM**

The judge class has a function to get and set their unique ID. This takes place of a specific name for the judge. As in the Olympics, I believe you see scores from Judge 1, Judge 2, etc. So, that will be their identifier. There is a judge's schedule function as well as a function to obtain their unique score.

**<<USES>> SCHEDULE <<USES>> SCORING**

Scoring class is now apropos of the mindset of Guercio.

**SCHEDULE SUBSYSTEM**

The schedule class will have something cryptic as of now. It is a list of type Tuple of type int,int where the ints are unix time. This is how we will assure everything gets assigned in an orderly fashion without overlap. Again, my friend is going to help me with this implementation.

Rink has a function of type List of type Event. This will house every Event that will partake on that particular rink. There will be six rinks (I believe). I am sure that can change. We can discuss later.

**<<USES>> EVENT**

**Event SUBSYSTEM**

The event class is a busy class. I put in a unique event number function to playaround with abstractness and inheritance (pure virtual). There are two main functions here. One function to run an event and one to schedule a future event. Both utilize a list of judges as well as a list of teams. **<<USES>> JUDGE <<USES>> TEAM**