# Tutorial 6

## Objectives

Find partners for the class project. Practice implementing the **Comparable** interface, implementing abstract methods and extending abstract classes.

## Attendance Quiz

Please log on to cuLearn using one of the computer in the tutorial room and complete the attendance quiz. You can only access the quiz if you log in using one of the computers in the lab. You cannot use a laptop for this. This is a time limited quiz. Be sure to do this as soon as you arrive.

At the end of the tutorial a TA will assign you a grade, call it G, which is 0, 1 or 2, depending on the progress you make during the tutorial. If you spend your time reading mail and chatting with your friends you will receive 0. If you have completed the attendance quiz on time then G will be your tutorial grade. If you have not completed the attendance quiz on time, then your tutorial grade will be max(0, G - 1/2). Each tutorial grade will therefore be one of 0, 0.5, 1.5 or 2.

## Project Teams

You must form a team of four (4) students from your tutorial section. Some teams of three will be allowed if the number of students in the tutorial is not a multiple of 4.

Starting in tutorial 7, tutorials will be done as a team.

## Comparable Warm-up

Create the **Box** class as defined by

```
public class Box implements Comparable<Box>
```

Each box object should have a String attribute and boxes should be compared by the length of these Strings. Override the **toString** method (inherited from **Object**) to simply display object's string attribute.

A sample testing program is provided. It creates an array of boxes, prints them, sorts them and then prints them again. The testing program should output

```
before sorting ------------------------
[kitten, cat, dog, oh, cow, oxen]
after sorting -------------------------
[oh, cat, dog, cow, oxen, kitten]
```

Change the *strings* array to be sure your code works correctly with different inputs.

---

## Comparable

Complete the provided **Team** class by overriding the **compareTo** method (inherited from the **Comparable** interface) and overriding the **toString()** method (inherited from **Object**) as described below.

Each team will have zero or more *points*. The number of points that a team has is two times the number of wins a team has plus the number of draws. So, if a team has 3 wins and 1 draw then that team has 7 points.

Let A and B be two teams. The ordering of teams (when compared) is as follows:

1. A < B when A has less points than B.
2. A > B when A has more points than B.
3. When A and B have the same amount of points, then
   a. A < B when A has played more games than B
   b. A > B when A has played less games than B
4. When A and B have the same amount of points and have played the same amount of games, then
   a. A < B when A has less wins than B
   b. A > B when A has more wins than B
5. A == B when A and B have the same amount of points, played the same number of games, have the same number of wins and same number of draws.

When you print a team object (**toString()**) it should be of the form

```
"name[points]: w/l/d"
```

where w,l,d are the wins, losses and draws of the team. For example, the output might look like

```
Carleton[12]: 5/3/2
```

---

## Extra

Practice implementing the **Comparable** interface with different ways of comparing the objects. Use the Team class or make up your own. Create your own (non-trivial) ways of ordering the teams. For example, add attributes for goals scored and goals against and use these to construct elaborate ways of comparing teams (in addition to wins/losses/draws and points).