COMP 1020
Lab 2

MATERIAL COVERED

- Basic objects – instance variables, instance methods, constructors
- Class variables and class methods (i.e. static)
- Sorting objects using compareTo (Gold exercise only)

Notes:

- In the lab, all exercises are to be done using Dr. Java. (At home, use anything, as usual.)
- Make sure your TA has recorded your mark before leaving.
- The three questions are sequential – each builds on the previous one.
- Always try to complete as many as you can. The Gold exercise uses some material that has not yet been fully covered, but it is explained briefly here.

A simple object class

1. Create a file **`HockeyTeam.java`** which will implement a class of objects that will store some information about a hockey team. In your file, implement the following variables and methods.

    a. **private** instance variables that will store the team's name (a **`String`**), and the number of wins, losses, and overtime losses (three **`int`** values).

    b. a constructor that will accept those four data values, in the order listed above. A new **`HockeyTeam`** object will always be created by using an expression such as

    ```
    new HockeyTeam("Winnipeg",22,14,8)
    ```

    c. three **public** methods **`void won()`**, **`void lost()`**, and **`void lostOvertime()`**, each of which will add one to the corresponding integer value.

    d. a **public** method **`int points()`** which will calculate and return the number of points that the team has. A team receives 2 points for each win, 0 points for each loss, and 1 point for each overtime loss.

    e. a **public** method **`String toString()`** which will return a **`String`** giving all of the information stored about a team, and the number of points that the team has, in the format **`"Winnipeg(22,14,8=52)"`** where 52 is the number of points (22*2+8). This method should call the **`points()`** method.

2. Run the supplied file **`TestLab1Bronze.java`** to test your **`HockeyTeam`** class. Look at the contents of that file to see how it is creating, updating, and printing team data. If your class is working correctly the output should be:

```
Initial teams:
Winnipeg(22,14,8=52)
Chicago(28,13,2=58)
Colorado(18,17,8=44)
St. Louis(27,13,3=57)
Dallas(19,16,7=45)
Minnesota(18,19,5=41)
Nashville(29,9,4=62)

Final teams:
Winnipeg(28,14,8=64)
Chicago(28,15,2=58)
Colorado(19,17,9=47)
St. Louis(28,13,4=60)
Dallas(19,18,7=45)
Minnesota(18,20,5=41)
Nashville(29,9,5=63)
```

Adding class variables and a class method (static variables and method)

---

1. Modify your **`HockeyTeam`** class by adding the following class (static) variables and methods:
    a. A few **private** class variables to keep a list of all of the **`HockeyTeam`** objects that have been created, as a partially-full array with a maximum capacity of 64 teams. You will need an array of **`HockeyTeam`** objects, an **`int`** to keep track of the current number of teams, and a constant for the maximum capacity (64). (A separate **`HockeyTeamList`** class would be another way to do this – don't do that this time. If you know how to use an **`ArrayList`**, go ahead and use one of those instead of an array if you want to.)
    b. Modify the constructor, so that every time a new **`HockeyTeam`** object is created it will be added to the list of **`HockeyTeam`** objects.
    c. Add a **public** class method **`void listTeams()`** which will print all of the teams in the list, one per line. You can simply use **`System.out.println`** to print a **`HockeyTeam`**, and the **`toString`** method you wrote in the Bronze exercise should take care of the rest automatically.
2. Modify the supplied file **`TestLab1Silver.java`** to test your new **`HockeyTeam`** class. In two places, marked with **`****ADD ONE LINE HERE****`**, add a line that will call your **`listTeams`** class method.
3. Run the **`TestLab1Silver.java`** file to test your modified class. The output should be identical to the Bronze exercise output, except that now it is your **`listTeams`** method that is doing the printing, not the **`main`** method.

Sorting the standings

When the list of teams is printed, it should be sorted into descending order according to the number of points each team has, as standings always are.

The built-in method **`Arrays.sort(a, 0, n)`** will sort elements **`a[0]`** to **`a[n-1]`** of *any* array of objects **`a`**, provided that a **`compareTo`** method has been supplied that will compare those objects. You will need the import statement **`import java.util.Arrays;`** to access this method.

Modify your **`HockeyTeam`** class to use this method to sort the teams.

1. Add a **public** instance method **`int compareTo(Object other)`** which will compare a **`HockeyTeam`** object to some **`other`** object. The type **`Object`** is a generic type that can refer to any object. Use the ordinary cast **`(HockeyTeam)other`** to convert it into a **`HockeyTeam`** object before using it. This method should return a negative number if the **`other`** team is "larger" and should appear later in the list, a positive number if the **`other`** team is "smaller" and should appear earlier in the list, and a zero if the two teams are equal and can appear in either order.
2. Add the words **`implements Comparable`** to your class header, just before the **`{`**. This means "I hereby promise to implement a **`compareTo`** method." and allows **`Arrays.sort`** to be used.
3. Write a **public** class method **`void sortTeams()`** which uses the **`Arrays.sort`** method to sort the teams in your list of teams into the proper order.
4. Modify the supplied file **`TestLab1Gold.java`** to test your new **`HockeyTeam`** class. In the places marked with **`****ADD ONE LINE HERE****`**, add lines that will first call your **`sortTeams`** class method, and then call your **`listTeams`** class method.
5. Run the **`TestLab1Gold.java`** file to test your modified class. The output should be identical to the Bronze exercise output, except that now the teams should be listed in descending order according to points.
