CS 1302A – HW 8 (Spring 2016)
Total: 100 pts
Due: Monday, April 25, 2016 11:59 pm

This homework has 3 problems which deal with sets and maps (Ch 21).

Eclipse Reminder

- You will create a Java Project in Eclipse with the name: *hw8_FLastname*
- You will have three packages: *prob1, prob2, prob3*

Problem 1

Do Programming Exercise 21-1 on page 818 from the text. Create a *prob1* package and add a new class named *Exercise21_1*. Note, the problem says to use *LinkedHashSet,* you can use *HashSet* if you like. However, there is absolutely no difference between the two in terms of this problem. The output should look like this:

```
Original Set 1: [George, Jim, John, Blake, Kevin, Michael]
Original Set 2: [George, Katie, Kevin, Michelle, Ryan]
Union of the two sets: [George, Jim, John, Blake, Kevin, Michael, Katie, Michelle, Ryan]
Difference of the two sets: [Jim, John, Blake, Michael]
Intersection of the two sets: [George, Kevin]
```

Hint: 1. To clone a Set *set1*, you may use the following code:

```
LinkedHashSet<String> set1Clone1 = (LinkedHashSet<String>)set1.clone();
```

2. Use the Collection methods: addAll, removeAll and retainAll

Problem 2

A *HashMap* holds the points scored (value) for each player (key) in game 1. A similar *HashMap* holds the points scored for each player in game 2. For example:

```
HashMap<String,Integer>  game1 = new HashMap<>();
game1.put("Chen", 4);
game1.put("Allie", 9);
game1.put("Reece", 4);
game1.put("Skuye", 6);
game1.put("Meshell", 6);

HashMap<String,Integer>  game2 = new HashMap<>();
game2.put("Chen", 8);
game2.put("Zyrene", 4);
game2.put("Skuye", 2);
game2.put("Meshell", 3);
```

Do the following:

1. Create a package named, “prob2” and a class named, “Games”.

2. Write a method named *getTotals* that accepts two *HashMaps* as described above and returns a *TreeMap* of all the players that played in both games and the total number of points they got. For the example above, the result is:

```
Key=Chen, Value=12
Key=Meshell, Value=9
Key=Skuye, Value=8
```

3. Write a *main* with the maps above, call *getTotals*, and print the resulting *TreeMap* as shown immediately above.

Problem 3

Do the following:

1. Create a package named, “prob3” and a class named, “Games2”.
2. Create a *Player* class and add this code:

```
public class Player {
    private String name;
    private int points;

    public Player(String name, int points) {
           this.name = name;
           this.points = points;
    }

    public String getName() {
           return name;
    }
    public int getPoints() {
           return points;
    }

    @Override
    public String toString() {
           return "name=" + name + ", points=" + points;
    }
}
```

3. Define a *TreeSet* in *main* to hold *Player* objects sorted by name, then points. Test with this code:

```
Player p1 = new Player("Benito", 33);
Player p2 = new Player("Quincy", 14);
Player p3 = new Player("Lean", 22);
Player p4 = new Player("Carly", 41);
Player p5 = new Player("Pepper", 18);
```

```
// Add your code to define TreeSet named "team" here

team.addAll(Arrays.asList(p1,p2,p3,p4,p5));
System.out.println("\nPlayers sorted by name:");
for(Player p : team) System.out.println(p);
```

Hint: You need to implement and use a Comparator, *NameComparator*, to sort players by name, then points, in a *TreeSet*

4. Write a method named *getPlayersAbove* that accepts a *TreeSet* of *Player* objects and an integer, *val*. The method should return a *set* of *Players* containing only the players with points at or above *val.*

   Hints:
   (1) You need to implement and use a Comparator, *PointsComparator*, to sort players by points, then name, in a *TreeSet*
   (2) Create a "dummy" player with points *val* and use the method *tailSet on TreeSet* to return a set of all players that are larger than or equal to the "dummy" player in terms of points.

5. Add the code below to the bottom of *main* to test:

```
Set<Player> big = getPlayersAbove(team,20);
System.out.println("\nPlayers with a lot of points:");
for(Player p : big) System.out.println(p);
```

The excepted outcome is as follows:

```
Players sorted by name:
name=Benito, points=33
name=Carly, points=41
name=Lean, points=22
name=Pepper, points=18
name=Quincy, points=14

Players sorted by points:
name=Quincy, points=14
name=Pepper, points=18
name=Lean, points=22
name=Benito, points=33
name=Carly, points=41

Players with a lot of points:
name=Lean, points=22
name=Benito, points=33
name=Carly, points=41
```

Submission

Make sure you zip the ENTIRE folder *hw8_FLastname*!! Submit your zip file to Blazeview by the due date. The name of your file should be: *hw8_FLastname.zip*.

Grading

I will RANDOMLY select one problem for grading. For example, if problem 1 is selected, I will only go to the package/folder *prob1* for grading. So make sure all your Java codes are in the correct packages/folders.

Additional Requirements:

1) No late submission will be accepted.
2) Please exactly follow the naming rules described above. You will be deducted 10 points for incorrect naming.
3) Write comments at the beginning of your Java source file(s) to indicate your name, student ID, "CS 1302-A Homework 8", and the due date.
4) Make sure that your programs are well-documented, readable, and user friendly. Make sure you document your programs by adding comments to your statements/blocks so that I can understand your code. You will get 0 to 10 points based on the helpfulness of your comments. You will be deducted 10 points for no comments.
5) It is your responsibility to make sure your programs can compile and run correctly. Please be aware that programs that do not compile may receive ZERO credit.
6) When grades are returned to you via BlazeView, you have 7 days to meet with the instructor to discuss on grade changes. After 7 days, the grades are written in stone and can't be changed after that point.