University of Maryland College Park
Department of Computer Science
CMSC131 Spring 2023
Exam #2

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE): KEY

STUDENT ID (e.g. 123456789):

Instructions

- Please print your answers and use a pencil.
- Do not remove the staple from the exam. Removing it will interfere with the Gradescope scanning process.
- To make sure Gradescope can recognize your exam, print your name, write your directory id at the bottom of pages with the text DirectoryId, provide answers in the rectangular areas provided, and do not remove any exam pages. Even if you use the provided extra pages for scratch work, they must be returned with the rest of the exam.
- This exam is a closed-book, closed-notes exam, with a duration of 50 minutes and 100 total points.
- Your code must be efficient.
- Multiple choice questions have only one answer unless indicated otherwise.
- You don't need to use meaningful variable names; however, we expect good indentation.

Grader Use Only

#1	Problem #1 (Short Answers - 2pts each)	16
#2	Write a method	14
#3	Write a class	70
Total	Total	100

Problem #1 (Short Answers – 2 pts each)

1. (2 pts) A default constructor of a class, has no parameters.

2. (2 pts) A static field is shared among all instances of a class.

3. (2 pts) The continue keyword can be used to skip the remainder of the current iteration in a loop and start a new one.

4. (2 pts) Which statement is true?
   a. All fields in a class have to be reference types.
   b. The control expression of a `switch` statement can be a `double`.
   c. The default value for an `int` field is null.
   d. A local `int` variable will not have a default value if not initialized with one.

5. (2 pts) Which statement is false?
   a. A class can have more than one copy constructor
   b. The keyword `this` can be used to call another constructor in the same class.
   c. The parameter of the `equals` method for a class, should be of type `Object`.
   d. If the last line in a method of the class `Person` is `return this;` the return type of the method can be `Person`.

6. (2 pts) Assume the following code fragment in a `main` method, what is the output?

```
String s =null;
try {
      s.length();
      System.out.println("Line 1");
}
catch(NullPointerException e){
      System.out.println("Line 2");
}
catch(Exception e){
      System.out.println("Line 3");
}
finally {
      System.out.println("Line 4");
}
```

Line 2
Line 4

7. (2 pts) Which of the three types of copying that we discussed in class is demonstrated in the code below?

```
StringBuffer s1 = new StringBuffer("ABC");
StringBuffer s2 = s1;
```

Reference copy

8. (2 pts) Assume the following code fragment in a `main` method, what is the output?

```
int [] x = {2,3};
int [] y = x;
y[0]=  2 > 3 ? --x[0]: x[1]++;
System.out.println(x[0]);
```

3

Problem #2 (Write a method – 14 pts)

1. Complete the code for the method below . This method takes in an array of `int` called `input` and an `int` called `num`. If the array has less than 2 elements or if the first element of the array is `num`, simply `return`. You can assume `input` will not be `null`. Otherwise, starting at the beginning of the array, replace every occurrence of `num` in the array with the value to its immediate left.

```
int a[] = {5,7,3,6,14,7, 31,7,7,63,7,8};
int b[] = {5,7,7,7,7,7};
replace (a, 7);
replace (b, 7);
System.out.println(Arrays.toString(a)); //prints [5, 5, 3, 6, 14, 14, 31, 31, 31, 63, 63, 8]
System.out.println(Arrays.toString(b)); //prints  [5, 5, 5, 5, 5, 5]
```

```
public static void replace (int [] input, int num)
{

           if (input.length <=1 || input[0] == num)
                 return;
           for (int i =1; i <input.length;i++)
           {
                 if (input[i]==num)
                       input[i]=input[i-1];

           }



 }
```

Directory id:

Problem #3 (Write a class)

Given the immutable class below (that you are not to modify in any way):

```
public class Player {
      private final int id;
      private final int skillLevel;

      public Player(int id, int skillLevel) {
            this.id = id;
            this.skillLevel = skillLevel;
      }
      public int getSkillLevel() {
            return skillLevel;
      }
      @Override
      public String toString() {
            return "[id=" + id + ", sL=" + skillLevel + "]";
      }
}
```

Complete the implementation of a class called `Team` that represents a team of up to 5 players . You MAY NOT add any methods other than the ones specified below (not even private). Also you MAY NOT add any additional instance or static variables. Local variables in your methods are fine.

```
import java.util.Arrays;

public class Team {

      private Player[] players;
      private int numOfPlayers;

      public Team() {
            //answer to P1
      }

      public Team(Team t) {
            //answer to P2
      }

      public void addPlayer(int id, int skillLevel) {
            //answer to P3
      }

      public Player[] getPlayers(){
            //answer to P4
      }

      public boolean equals(Object obj) {
           //answer to P5
      }

      @Override
      public String toString() {
            return Arrays.toString(players) +  ", numOfPlayers=" + numOfPlayers;
      }

      public static Team mergeTeams(Team t1, Team t2) {
              //answer to P6
      }
}
```

Driver

```
import java.util.Arrays;

public class Driver {

      public static void main(String[] args) {

            Team t1 = new Team();
            t1.addPlayer(123, 5);
            t1.addPlayer(124, 7);
            t1.addPlayer(125, 7);
            System.out.println(t1);

            Team t2 = new Team();
            t2.addPlayer(223, 10);
            t2.addPlayer(224, 9);
            System.out.println(t2);

            System.out.println(t1.equals(t2));

            Team t3 = Team.mergeTeams(t1, t2);
            System.out.println(t3);

            try {
                  t3.addPlayer(321, 8);
            }
            catch (UnsupportedOperationException e){
                  System.out.println(e.getMessage());
            }

            Team t4 = new Team(t2);
            t2.addPlayer(225, 3);
            System.out.println(t2);
            System.out.println(t4);

            Player temp [] =t2.getPlayers();
            System.out.println(Arrays.toString(temp));
            temp[0]=temp[1]=null;
            System.out.println(Arrays.toString(temp));
            System.out.println(t2);
      }
}
```

Output

```
[[id=123, sL=5], [id=124, sL=7], [id=125, sL=7], null, null], numOfPlayers=3
[[id=223, sL=10], [id=224, sL=9], null, null, null], numOfPlayers=2
true
[[id=123, sL=5], [id=124, sL=7], [id=125, sL=7], [id=223, sL=10], [id=224, sL=9]], numOfPlayers=5
already at 5 players
[[id=223, sL=10], [id=224, sL=9], [id=225, sL=3], null, null], numOfPlayers=3
[[id=223, sL=10], [id=224, sL=9], null, null, null], numOfPlayers=2
[[id=223, sL=10], [id=224, sL=9], [id=225, sL=3]]
[null, null, [id=225, sL=3]]
[[id=223, sL=10], [id=224, sL=9], [id=225, sL=3], null, null], numOfPlayers=3
```

Directory id:

1. This **constructor** should assign a `Player` array with 5 elements to the `players` field. Since `Player` objects are not added in this method, the elements will default to `null`. Set the `numOfPlayers` field to zero.

```
public Team() {

      players = new Player[5];  //all 5 are null

      numOfPlayers = 0;

      }
```

2. Define a **copy constructor** method for the class. Changes to the copy should not affect the original object. Remember that the `Player` class is immutable.

```
public Team(Team t) {

            players = new Player[5];

            for (int i =0; i < 5; i++)
            {
                  players[i]=t.players[i];  //copy all over, even the null
            }

            numOfPlayers = t.numOfPlayers;

      }
```

3. If the numOfPlayers field is 5, throw the UnsupportedOperationException with the message already at 5 players. Otherwise, make a new Player object, constructing it with the values passed into addPlayer. Assign the reference of this Player object to next available element of the array referenced by the field players. Increment the numOfPlayers field to indicate the number of players on the team has increased by one.

```
public void addPlayer(int id, int skillLevel) {

             if (numOfPlayers ==5)

                  throw new UnsupportedOperationException("already at 5 players");

             players[numOfPlayers++]=new Player(id, skillLevel);

      }
```

4. This method returns a new Player array with the team's players. The length should match the actual number of players (i.e. it will not necessarily be 5). Remember that the Player class is immutable.

```
public Player[] getPlayers(){

            Player[] retVal = new Player[numOfPlayers];

            for (int i =0; i < numOfPlayers; i++)

            {

                  retVal[i]= players[i];

            }

            return retVal;


      }
```

Directory id:

5. Finish the `equals` method below. Two teams are equal if they have the same number of players **OR** if the sum of the skill levels of the players in both teams are the same. Be careful to not get a `NullPointerException`!

```
public boolean equals(Object obj) {
            if (obj == this)
                  return true;
            if (obj == null || getClass() != obj.getClass())
                  return false;



            Team t = (Team) obj;

            int total1 =0, total2 =0;

            for (int i =0; i <5; i++)
            {
                  if (players[i]!=null)
                        total1 +=players[i].getSkillLevel();
                  if (t.players[i]!=null)
                        total2 +=t.players[i].getSkillLevel();
            }

            if (numOfPlayers ==t.numOfPlayers || total1==total2 )
                  return true;
            else
                  return false;
      }
```

6. If the sum of the players in `t1` and `t2` is greater than 5, simply return `null`. Otherwise, return a new `Team` object that has the players from t1 followed by the players in t2 (with no `null` elements between the players). Do not forget to assign the correct value to the `numOfPlayers` of the new object.

```
public static Team mergeTeams(Team t1, Team t2) {

            if (t1.numOfPlayers+t2.numOfPlayers>5)
                  return null;

            Team retVal = new Team();
            int index =0;

            for (int i =0; i <t1.numOfPlayers; i++){

                  retVal.players[index]=t1.players[i];
                  index++;
            }

            for (int i =0; i <t2.numOfPlayers; i++){

                  retVal.players[index]=t2.players[i];
                  index++;
            }
            retVal.numOfPlayers =index;


            return retVal;

      }
```

Directory id:

EXTRA PAGE IN CASE YOU NEED IT

LAST PAGE