Lab Partner 1 Name

Lab Partner 2 Name

CS 115 Fall 2019 Lab #7

Due: **Monday, October 17th, midnight**

Points: **20**

Instructions:

1. Use this document template to report your answers. Enter all lab partner names at the top of first page.

2. You don't need to finish your lab work during the corresponding lab session.

3. ZIP your lab report and java files (if any) into a single ZIP file. Name the ZIP file as follows:

`LastName_FirstName_CS115_Lab7_Report.zip`

4. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.

5. ALL lab partners need to submit a report, even if it is the same document.

Objectives:

1. (10 points) Write and test a user-defined class.

2. (10 points) Write and test a user-defined class and use iterations

Problem 1 [10 points]:

A) **[7 points] Design and implement a class** `BobsLife` that simulates a simple virtual world. In this world there is a person called Bob that is at a location (use a String type to store location) and there are three possible locations:

- at home,
- at work,
- or at the gym.

Bob has three integer characteristics: his hunger, his fitness, and the dollars he owns. The class **should have an instance method** called `move` that moves Bob from his current location (home, work, or gym) **to a new location specified by the method argument**.

The simulation starts at time 0 (another **attribute** of Bob). Time moves in steps. The simulation is **moved forward by one step manually by calling an instance method** `nextTime` after each `move` call.

Depending on Bob's location when method `nextTime` is called, the following happens:

- If Bob is **at home**, then his **hunger is decreased by 3, because he eats a meal**. **However, the hunger cannot drop below 0**. Furthermore, **his dollars are decreased by 1 because food costs money**,
- If Bob is **at work**, then his **hunger is increased by 2, because working makes him hungry**. Furthermore, **his dollars are increased by 3 (he earns money at his job)**. Also: he has a desk job and so **his fitness decreases by 1**,
- If Bob is **at the gym**, then his **hunger is increased by 3 because a workout makes him hungry**. His **dollars are decrease by 2 since the gym costs money**. On the positive side **his fitness increases by 2**.

In addition, this virtual world is quite harsh:

- If Bob's hunger goes above 6 then the poor guy starves to death.
- If his dollars drop below zero than he is broke and is thrown into jail for a life time sentence.
- If his fitness is 0 then he dies of a heart attack.

Your class should keep track of whether Bob is dead or in jail (use Boolean data type to keep track of that). If either of these things happens, then it should no longer be possible to move Bob to another location and his characteristics no longer change if the `nextTime` method is called.

The parametrized constructor of the class `BobsLife` should:

- initialize the parameters of the simulation (`time`, `isDead`, `inJail`),
- and accept arguments for:
  - Bob's initial location,
  - and Bob's initial characteristics (`hunger`, `fitness`, `dollars`).

You only need a **private** mutator method for the location to verify the argument is correct and set some default location if incorrect. **No accessor methods are needed**.

Finally, write a `toString()` method for your class that will generate a String type value describing Bob's "state" at current time (see sample output below).

B) [3 points] Write a class `LongLiveBob` which runs a BobsLife simulation in its `main` method.

The task is to instantiate a new `BobsLife` object with:

- 0 hunger,
- 5 dollar,
- and 4 fitness.

Then execute a series of three statements:

- `move`(you determine the location),
- `nextTime()`,
- and `toString()` - again and again

without causing Bob to die or go to jail. In each step you have the free choice to move Bob to any of the locations. Using trial-and-error, try to come up with a sequence of 15 or 20 moves that do not lead to Bob's demise or imprisonment. You may see a pattern that you can repeat. For example, here is some sample output where Bob goes to work and stays at work without moving:

```
Time: 0 - location:home, hunger:0, dollars:5, fitness:4 (alive and well)
Time: 1 - location:work, hunger:2, dollars:8, fitness:3 (alive and well)
Time: 2 - location:work, hunger:4, dollars:11, fitness:2 (alive and well)
Time: 3 - location:work, hunger:6, dollars:14, fitness:1 (alive and well)
Time: 4 - location:work, hunger:8, dollars:17, fitness:0 (deceased)
```

`BobsLife` **class:**

```
public class BobsLife {
    private String location;
    private int hunger;
    private int fitness;
    private int dollars;
    private int time;
    private boolean isDead;
    private boolean inJail;
    private final String DEFAULT_LOCATION="home";

    public BobsLife (String l, int h, int f, int d) {
        time = 0;
        isDead = false;
        inJail = false;
        setLocation(l);
        hunger=h;
```

```
        fitness=f;
        dollars=d;
    }

    private void setLocation(String l) {
        if(!isDead && !inJail) {
            if (l!=null && (l.equals("home") || l.equals("work") || l.equals("gym")))
                location = l;
            else location=DEFAULT_LOCATION;
        }
    }

    public void move(String to) {
        setLocation(to);
    }

    public void nextTime() {
        time++;
        if (inJail || isDead) return;

        if (location.equals("home")) atHome();
        else if (location.equals("gym")) atGym();
        else if (location.equals("work")) atWork();
    }

    private void atHome() {
        hunger = Math.max(0, hunger - 3);
        dollars -= 1;
        checkState();
    }

    private void atWork() {
        hunger+= 2;
        dollars += 3;
        fitness -= 1;
        checkState();
    }

    private void atGym() {
```

```
        hunger += 3;
        dollars -= 2;
        fitness += 2;
        checkState();
    }

    private void checkState() {
        if (dollars < 0) inJail = true;
        if (hunger > 6) isDead = true;
        if (fitness == 0) isDead = true;
    }

    public String toString() {
        return "Time: "+time+" - location:"+location+", hunger:"+hunger+",
dollars:"+dollars+", fitness:"+fitness+" ("+getStatus()+")";
    }

    public String getStatus() {
        if (isDead)
            return ("deceased");
        if (inJail)
            return ("in jail");
        return "alive and well";
    }
}
```

`LongLiveBob` **class:**

```
public class LongLiveBob {
    public static void main(String[] args) {
        BobsLife b = new BobsLife("home", 0, 4, 5);
        System.out.println(b.toString());

        // repeated sequence that lasts 22 moves
        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
```

```
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("work");
        b.nextTime();
        System.out.println(b.toString());

        b.move("gym");
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("work");
        b.nextTime();
        System.out.println(b.toString());


        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
        b.nextTime();
        System.out.println(b.toString());
```

```
b.move("work");
b.nextTime();
System.out.println(b.toString());

b.move("gym");
b.nextTime();
System.out.println(b.toString());

b.move("home");
b.nextTime();
System.out.println(b.toString());

b.move("home");
b.nextTime();
System.out.println(b.toString());

b.move("work");
b.nextTime();
System.out.println(b.toString());

b.move("home");
b.nextTime();
System.out.println(b.toString());

b.move("home");
b.nextTime();
System.out.println(b.toString());

b.move("home");
b.nextTime();
System.out.println(b.toString());

b.move("work");
b.nextTime();
System.out.println(b.toString());

b.move("gym");
b.nextTime();
```

```
        System.out.println(b.toString());


        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("home");
        b.nextTime();
        System.out.println(b.toString());

        b.move("work");
        b.nextTime();
        System.out.println(b.toString());



        b.move("home");
        b.nextTime();
        System.out.println(b);

        b.move("home");
        b.nextTime();
        System.out.println(b);

        b.move("home");
        b.nextTime();
        System.out.println(b);

        b.move("work");
        b.nextTime();
        System.out.println(b);

        b.move("gym");
        b.nextTime();
        System.out.println(b);

        b.move("home");
        b.nextTime();
        System.out.println(b);
```

```
        b.move("home");
        b.nextTime();
        System.out.println(b);

        b.move("work");
        b.nextTime();
        System.out.println(b);

    }
}
```

Problem 2 [10 points]:

A) **[4 points] Design and implement TWO classes:**

- `SixSidedDie` that represents a six-sided fair (all outcomes are equally likely to happen) die,
- `TwelveSidedDie` that represents a twelve-sided fair die,

Both classes should have:

- a **private** field called `value` that is INITIALLY set to 1,
- a **public** method called roll() that will generate a random integer (from the set {1,2,3,4,5,6} for `SixSidedDie` and from the set {1,2,3,4,5,6,7,8,9,10,11,12} for `TwelveSidedDie` that will get stored in `value` **AND return that integer**,
- a **public** `toString()` method that will return a String saying "rolled <rolled_value>", where <rolled_value> is the current number assigned to the `value` field.

You will need to import `java.util.Random` package.

B) **[6 points]** Implement a class `RollDice` that in its `main` method will:

- Instantiate one object, called A of class `SixSidedDie`,
- Instantiate one object, called B of class `TwelveSidedDie`,
- Use iterations to play:
  - Game 1: Roll both dice 10 times and sum and display all the rolled values (A and B rolls for each iteration),
  - Game 2: Keep rolling both dice until the sum of rolled numbers between A and B is 8.

Your `RollDice` program output should look similar to:

```
Game 1
Iteration 1: A rolled 2 and B rolled 1
Iteration 2: A rolled 3 and B rolled 2
Iteration 3: A rolled 5 and B rolled 3
```

```
Iteration 4: A rolled 1 and B rolled 10
Iteration 5: A rolled 1 and B rolled 6
Iteration 6: A rolled 5 and B rolled 8
Iteration 7: A rolled 2 and B rolled 6
Iteration 8: A rolled 1 and B rolled 4
Iteration 9: A rolled 4 and B rolled 1
Iteration 10: A rolled 1 and B rolled 10

Final sum of rolls: 76

Game 2
Iteration 1: A rolled 3 and B rolled 1
Iteration 2: A rolled 4 and B rolled 2
Iteration 3: A rolled 3 and B rolled 5

it took 3 iterations to roll a sum of 8.
```

SixSidedDie **class:**

```
// Import a random generator package
import java.util.Random;


class SixSidedDie {
    // Declare a private integer variable called value and set it initially to 1
    private int value = 1;


    // Public roll method. It has to return a "rolled" integer
    public int roll(){
        // Instantiate a random generator object
        Random randomGenerator = new Random();


        // Generate a random integer from the set {1,2,3,4,5,6}
        this.value = randomGenerator.nextInt(5) + 1;


        // Return value of this integer
        return this.value;
    }


    // toString method
    public String toString(){
        // Notice the integer to String conversion
        return "rolled " + Integer.toString(this.value);
    }
}
```

`TwelveSidedDie` **class:**

```
// Import a random generator package
import java.util.Random;

class TwelveSidedDie {
    // Declare a private integer variable called value and set it initially to 1
    private int value = 1;

    // Public roll method. It has to return a "rolled" integer
    public int roll(){
        // Instantiate a random generator object
        Random randomGenerator = new Random();

        // Generate a random integer from the set {1,2,3,4,5,6,7,8,9,10,11,12}
        this.value = randomGenerator.nextInt(11) + 1;

        // Return value of this integer
        return this.value;
    }

    // toString method
    public String toString(){
        // Notice the integer to String conversion
        return "rolled " + Integer.toString(this.value);
    }
}
```

`RollDice` **class:**

```
class RollDice {
    public static void main( String [] args ) {
        // Program variables
        int sum = 0;
        int partialSum = 0;
        boolean rolled8 = false;

        // Instantiate dice objects
        SixSidedDie A = new SixSidedDie();
        TwelveSidedDie B = new TwelveSidedDie();
```

```
        // Start Game 1
        System.out.println("Game 1");

        for (int i = 1; i < 11; i++){
            // Use roll class methods to calculate partial sum and update total sum
            sum += A.roll() + B.roll();

            // Display roll results
            System.out.println("Iteration " + i + ": A " + A.toString() + " and B " +
B.toString());
        }

        // Display the final sum for Game 1
        System.out.println();
        System.out.println("Final sum of rolls: " + sum);

        // Start Game 2
        System.out.println();
        System.out.println("Game 2");

        // Reset i counter (you can use other variable for that variable)
        int i = 1;

        // rolled8 boolean is already set to false.
        // While loop will continue looping until rolled8 stops being false / becomes
true
        while(rolled8 != true){
            // Use roll class methods to calculate partial sum
            partialSum = A.roll() + B.roll();

            // Display roll results
            System.out.println("Iteration " + i + ": A " + A.toString() + " and B " +
B.toString());

            // Increment iteration counter. It's not a for loop, we have to do it
ourselves
            i++;

            // Check if partial sum is 8, if it is set the rolled8 boolean switch/flag
to true so we can exit the loop
```

```
            if (partialSum == 8){
                rolled8 = true;
            }
        }

        // OR ALTERNATIVELY (uncomment code below)
        //rolled8 = false;
        //i = 1;
        //do {
        //  partialSum = A.roll() + B.roll();
        //  System.out.println("Iteration " + i + ": A " + A.toString() + " and B " +
B.toString());
        //  i++;
        //  if (partialSum == 8){
        //      rolled8 = true;
        //  }
        //} while (rolled8 == false);
        // END OF ALTERNATIVELY

        // Display the final number of iterations for Game 2
        System.out.println();
        System.out.println("it took " + (i-1) + " iterations to roll a sum of 8.");
    }
}
```