Name: _____

USC NetID (e.g., `ttrojan`): _____

CS 455 Midterm Exam 1
Spring 2017 [Bono]

Thursday, Feb. 16, 2017

There are 5 problems on the exam, with 58 points total available. There are 12 pages to the exam (6 pages **double-sided**), including this one; make sure you have all of them. If you need additional space to write any answers or scratch work, pages 10, 11 and 12 are left blank for that purpose. If you use those pages for answers you just need to direct us to look there. *Do not detach any pages from this exam.*

Note: if you give multiple answers for a problem, we will only grade the first one. Avoid this issue by labeling and circling your final answers and crossing out any other answers you changed your mind about (though it's fine if you show your work).

Put your name and USC username (a.k.a., NetID) at the top of the exam. Also put your NetID at the top right of the front side of each page of the exam. Please read over the whole test before beginning. Good luck!

Selected methods of Java `Rectangle` class:

`Rectangle(x, y, width, height)`

Constructs rectangle object whose upper-left corner is (x, y) and whose width and height are specified by the arguments of the same name.

`void translate(int dx, int dy)`

Changes `x` and `y` values of this rectangle by `dx` and `dy`, respectively. I.e., if this rectangle had upper-left coordinates `(x, y)`, its value after the call is a rectangle with upper-left coordinates `(x+dx, y+dy)` [this is a mutator]

Selected methods of Java `Random` class:

`Random()`

Creates a random number generator.

`int nextInt(int max)`

Generates and returns a random integer in the range `[0,max)`, (i.e., 0 <= num < max)

Problem 1 [10 pts. total]

Consider the following program. Note: it uses the Java `Rectangle` class – more information about `Rectangle` on the cover of the exam.

```
public class Prob1 {
  public static void foo(Rectangle r) {
    r = new Rectangle(60, 70, 15 20);
    r.translate(5, 10);
  }

  public static void main(String[] args) {
    Rectangle box = new Rectangle(6, 8, 20, 30);
    foo(box);
    Rectangle house = null;
    foo(house);
    System.out.println(box.getX() + " " + box.getY());
    System.out.println(house.getX() + " " + house.getY());
  }
}
```

Part A [9]. In the space below, draw a box-and-pointer diagram (a.k.a., memory diagram) showing all object variables, objects, and their state as they have changed during the code sequence. To distinguish between the two calls to `foo`, you can refer to the parameter as r_1 in the first call and r_2 in the second call.

Part B [1]. How many Rectangle objects are created by the code?

Problem 2 [6 pts.]

Consider the following static method that is *supposed* to return `true` if the array `vals` has more `1`'s than `2`'s, and `false` otherwise (there are no restrictions on what numbers might be in the array). For example, if `vals` contained `[1, 2, 3, 2, 7, 1]`, the method should return `false`, because `vals` has the same number of `1`'s as `2`'s.

However, the code below doesn't always do the right thing.

```
public static boolean more1sThan2s(int[] vals) {
  int count1 = 0;
  int count2 = 0;
  for (inti  = 0; i < vals.length; i++) {
    if (vals[i] == 1){
      count1++;
      count2 = 0;
    }
    else if (vals[i] == 2) {
      count2++;
      count1 = 0;
    }
  }
  return count1 > count2;
}
```

Do not modify the code. **Show two example parameter values and the result of calling the method above on each of them: the first one must be one where the code above returns an incorrect answer, and a second one such that the code above returns a correct answer:**

`vals`	return value of **`more1sThan2s(vals)`**

1. (incorrect)

2. (correct)

Problem 3 [11 pts total]

Refactor the main class below so it is shorter overall: your new version will involve a new method you will write and use. You may not use static variables. This program is to test a class called `CTS`, that has methods similar to the `CoinTossSimulator` from pa1, but simulates tossing only *one* coin repeatedly. Your new version of `CTSTester` must have the same functionality and output as the version here (except the results of the `CTS` method calls might be different since it simulates random events). To help decrease the amount writing you have to do, we have used short names here; additionally, **for any statements that are *exactly* the same in the new program as the old one, you may list its statement number from the old program in your new code instead of writing out that statement again.**

Note: you should not implement the `CTS` class, just reimplement the `CTSTester` class.

```
1   public class CTSTester {
2      public static void main(String[] args) {
3        CTS cs = new CTS();
4        System.out.println("After run(1000):");
5        cs.run(1000);
6        System.out.println("Number of trials [exp: 1000]: " + cs.getNumTrials());
7        System.out.println("Num heads: " + cs.getHeads());
8        System.out.println("Num tails: " + cs.getTails());
9        System.out.println("Tosses add up correctly? " +
10             (cs.getNumTrials() == (cs.getHeads() + cs.getTails());
11       System.out.println();
12       System.out.println("After run(5):");
13       cs.run(5);
14       System.out.println("Number of trials [exp: 1005]: " + cs.getNumTrials());
15       System.out.println("Num heads: " + cs.getHeads());
16       System.out.println("Num tails: " + cs.getTails());
17       System.out.println("Tosses add up correctly? " +
18             (cs.getNumTrials() == (cs.getHeads() + cs.getTails());
19       System.out.println();
20       System.out.println("After run(12):");
21       cs.run(12);
22       System.out.println("Number of trials [exp: 1017]: " + cs.getNumTrials());
23       System.out.println("Num heads: " + cs.getHeads());
24       System.out.println("Num tails: " + cs.getTails());
25       System.out.println("Tosses add up correctly? " +
26             (cs.getNumTrials() == (cs.getHeads() + cs.getTails());
27       System.out.println();
28
```

[space for your answer on the next page]

Problem 3 (cont.)

Complete your refactored `CTSTester` below:

```
public class CTSTester {
   public static void main(String[] args) {
```

Problem 4 [16 points]

Implement the class `SlotMachine`, which simulates a simple slot machine whose behavior is described here:

The slot machine starts with some number of tokens inside it, that can be used to pay out when a player wins. When a player puts in a token and pulls a lever, the slot machine has three reels that spin independently until they stop. As a result of the spin, sometimes the user gets a prize (the payout) of some tokens based on what is showing at the front of the reels when they stop spinning. Each reel has 6 symbols, each of which can come up with equal probability. The conditions when the player can get a prize, and the payout for that condition are listed in the table below. (Note: The symbols are represented in the class as distinct integer values.)

Symbols showing at end of spin:	**Player payout:**	
Three 0's	1	
Three 1's	2	
Three 2's	4	
Three 3's	6	
Three 4's	8	
Three 5's	10	
any other combination	0	[no tokens given out]

A spin can fail if the number of tokens currently in the machine is less than the number needed for the payout. In that case, the spin method returns the special value `SHORT_PAY`, and no tokens are paid out. The player can keep playing after a `SHORT_PAY`, and they could still get a later payout if there is enough in the machine; or the operator can `addTokens`, so that later plays will be able to pay out sufficient tokens.

Notes: (1) This class has no I/O. (2) Info about the Java `Random` class appears on the cover of the exam.

See the class and method comments and headers for the exact interface. Space for your answer starts here and continues onto the next two pages.

```
// SlotMachine simulates a simple slot machine.  Payout rules are given in
// the problem description
public class SlotMachine {
   public static final int SHORT_PAY = -1;
   public static final int NUM_SYMBOLS = 6;
   // space for instance variables here:
```

[class definition continued next page]

Problem 4 (cont.)

Complete the implementation of `SlotMachine` – details of this problem are on the previous page.

```
// Creates slot machine with some number of tokens in it.
// It initially displays three zeros (see "get" methods below).
// @param startTokens the number of tokens in the machine at the start
// PRECONDITION: startTokens >= 0
public SlotMachine(int startTokens) {









}

// Simulate one play at the slot machine. A spin with no payout increases the
// number of tokens in the machine by one.  But if the play results in a payout,
// the number of tokens could stay the same or decrease from what it was before
// the spin.  The rules for payouts are given in the problem description.
// @return the payout from the result of this spin or SHORT_PAY if there aren't
//         enough tokens in the machine to give the correct payout.
 public int spin() {









}
```

[class definition continued next page]

Problem 4 (cont.)

[SlotMachine class definition, continued]

```
    // Add some tokens to the machine.  (This does not correspond to playing
    // the machine, but would be done by the machine operator.)
    // @param numTokens the number of tokens to add to the machine
    // PRECONDITION: numTokens >= 0
    public void addTokens(int numTokens) {









    }


    // Accessor to get the number of tokens currently in the machine.
    //(This is a machine operator method.)
    public int getTokensLeft()









    }


    // the following three accessors, getReel1Symbol, etc., allow client to
    // access what's currently being displayed on the reels.
    // @return a number in the range 0 through NUM_SYMBOLS - 1
    public int getReel1Symbol() {









    }

    public int getReel2Symbol() {









    }

    public int getReel3Symbol() {









    }
}
```

Problem 5 [15 points]

Write the static Java `boolean` method, `hasDouble5or8`, which takes an array of numbers and returns `true` if the array contains two 5's in a row, or it contains two 8's in a row, but not both.

Examples

`vals`	**`hasDouble5or8(vals)`**
[5, 5]	*true*
[5, 7, 5, 8, 5]	*false*
[1, 8, 8, 2, 5, 5, 3, 3, 2]	*false*
[2, 17, 5, 8, 8, 8]	*true*
[3, 7]	*false*
[]	*false*

```
public static boolean hasDouble5or8(int[] vals) {
```

Extra space for answers or scratch work.

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.