Name:________________________________

CS162 Computer Science I Fall 2018

Practice Exam 1
DRAFT (9 Oct.)

The real test will look much like this one, but it will be shorter. I suggest taking this practice test under "real" conditions (closed book, closed note) first, then going back and using your book and notes to correct your answers. I will grade parts of this test for homework points, then give it back in time for you to study from it. I strongly suggest that you limit your collaboration.

The real test will be **Monday**, 15 October.

- Sample short answer essay questions: (you need not turn these in):
    - Briefly explain the difference between classes and objects.
    - Explain the difference between an actual parameter and a formal parameter
    - What does "void" mean.
    - What are "scope" and "lifetime" (with respect to Java).
    - What is "shadowing" and how does it affect a Java program?
- Review the problems on CodingBat.

1a). Which of the following expressions evaluate to true?

- !(4<5)
- !false
- (2>2) || ((4 == 4) && (1 < 0))
- (2>2) || (4 == 4) && (1 < 0)
- (34 != 33) && !false

1b). Evaluate the following expressions:

______ 3 + 4 * 5
______ 4 * 5 + 6
______ 35 % 7
______ 35 % 8
______ 5 + 4 % 2
______ 5 + 4 % 2 + 3
______ "C" + 3 + "PO"
______ 7 + 6 + " Trombones"
______ 27 / 4
______ 3 + 4 / 5
______ 3 + 4 / 5 + 6

2). Complete the method below that calculates how many packages of hot dogs you will use (i.e., open) and how many hot dogs will be left in an open package. Assume that half the people at your party (rounded down, if odd) will eat two hot dogs and the rest will eat only one. You may not use any of the methods in the `Math` package (e.g., `Math.round`). Remember how Java handles integer division.

For example, if the input is 50, your output should look like this:

```
You will use 10 packages and have 5 hot dogs left over.
```

```
class HotDogs
{
  public void dogsAndBuns(int people_at_party)
  {
     final int DOGS_PER_PACKAGE = 8;
     // Your code here










  } // end method
} // end class
```

3). On the back of the previous page, write a `Student` class that contains the following:

a) Four fields: `String`s named `firstName` and `lastName`, an `int` named `creditHours`, `and a double` named `qualityPoints`
b) A constructor that takes the `Student`'s names as parameters, but sets `creditHours` and `qualityPoints` to `0`.
c) A method named `getGPA` that takes no parameters and returns the student's GPA (quality points divided by credit hours).
d) A method named `addClass` that takes as parameters the number of hours of the class and the student's grade in the class. Update `creditHours` and `qualityPoints` accordingly.
e) A method named `isOnDeansList` that takes no parameters and returns `true` if the student has a GPA of 3.0 or more, and returns `false` otherwise. Do not write code to compute the GPA; use the existing `getGPA()` method instead.

Now, follow the directions in the comments below to complete the method `testSudentClass`.
Note: This method is **not** part of the `Student` class.

```
public void testStudentClass() {

    // Declare and create two Student objects using appropriate literal values (i.e., make up data).









    // Call the addClass method once for each student using appropriate literal values.









    // Print the average of the two students' GPAs. (Use getGPA, don't calculate GPA by hand.)









} // end testStudentClass
```

4). Consider the `Date` class described below. Then, complete the `isWeekend` and `testDateClass` methods described below. Do not implement the constructor, `getDayOfWeek`, `daysUntilChristmas`, `increment`, or `toString`.

```
class Date {

  public Date(int year, int month, int day) {…}
  public String getDayOfWeek()  { … }
  public int daysUntilChristmas() { … }
  public void increment(int numDays) { … };
  public String toString() { … }
}
```

Write a method named `isWeekend` that takes no parameters and returns `true` if the current date is a Saturday or Sunday.

Complete the method below according to the comments. This method is **not** part of the `Date` class.

```
public void testDateClass()
{
    // Create a Date object using appropriate literal values.

    // Call the Date object's toString method and print the String returned.

    // If it's Christmas print out "It's Christmas! If it's not Christmas, print "I can't wait" and
    // use the daysUntilChristmas and increment methods together to set the date object
    // to Christmas.

}
```

5). For each of the provided terms at the right, circle an example in the code and label it with the appropriate number. If the term does not apply anywhere in the code on this place, write “none”. Use pencil, plan ahead, and label neatly! Don’t obscure the code, you’ll need it later.

1 – string concatenation
2 – internal method call
3 – field declaration
4 - constructor
5 – conditional stmt
6 – equality operator
7 – formal parameter
8 – external method call
9 – return type
10 – actual parameter
11 – return value
12 – local variable
13 – comment

```
class MysteryGamePlayer
{
     private int redPoints;
     private int bluePoints;
     private int greenPoints;

     public MysteryGamePlayer(int a, int b)
     {
       redPoints = a;
       bluePoints = b;
       greenPoints = 0;
     }

     public void setGreenPoints(int c)
     {
       greenPoints = c;
     }

     public int getGreenPoints()
     {
       return greenPoints;
     }

     public int method1()
     {
       return  redPoints - bluePoints + greenPoints;
     }

     public int method2(int p)
     {
       int m1val = method1();
       return m1val + (p – 1)*greenPoints;
     }

     public void modify(int d, int e)
     {
       redPoints = redPoints*d;
       bluePoints = bluePoints*4;
       e = greenPoints;
     }

     public String toString()
     {
       return “rgb points: “ + redPoints + “ “ + bluePoints +
              “ “ + greenPoints;
     }
} // end class MysteryGamePlayer
```

6). Predict the output of the method `playMysteryGame`. This code uses the `MysteryGamePlayer` class on the previous page. There are no compile errors, or run-time errors. (If you find one, then I made a mistake when preparing the problem.)

```
class MysteryGame
{

     public void playMysteryGame()
     {
       MysteryGamePlayer object1 = new MysteryGamePlayer(1,2);
       System.out.println("A:  " + object1.method1());

       object1.setGreenPoints(3);
       System.out.println("B: " + object1.method1());
       System.out.println("C: " + object1.method2(2));

       int a = 5;
       int b = 6;
       MysteryGamePlayer object2 = new MysteryGamePlayer(b, a);
       System.out.println("D:  " + object2.method1());
       object1.setGreenPoints(7);
       System.out.println("E: " + object2.method1());
       System.out.println("F: " +
                          object2.method2(object1.getGreenPoints()));



       int d = 3;
       int e = 4;
       object1.modify(d, d);
       System.out.println("H:  " + object1.toString());

       object2.modify(object1.method2(e) + d, 0);
       System.out.println("I: " + object2.toString());

}
```

7). Complete the method below that takes all of `m1`'s green points and gives them to `m2`.

```
void giveAndTake(MysteryGamePlayer m1, MysteryGamePlayer m2)
{









}
```

8). In the state of Georgia, you must use a car (or booster) seat until you are either (1) at least 7 years old, or (2) weigh at least 70 pounds. Complete the method `carSeatRequired` below that returns `true` if, and only if, the child described still needs a car or booster seat. For example:

- `carSeatRequired(8, 65)` should return `false`
- `carSeatRequired(8, 75)` should return `false`
- `carSeatRequired(6, 65)` should return `true`
- `carSeatRequired(6, 75)` should return `false`

```
public static boolean carSeatRequired(int age, double weight)
{









} // end carSeatRequired
```

9). Write a line of code that would give different answers if the associativity were left to right instead of right to left.

10). Write a line of code that would give different answers if the precedence were different.

11). What is "short-circuit evaluation"?

12). Write a line of code that relies on short-circuit evaluation for correct operation.

13). Explain what the following code does in plain English. (Specifically, describe what the method does as a whole. Don't describe it line-by-line.)

```
public static boolean mystery(String str)
{
   int lastIndex = str.length() – 1;
   for (int i = 0; i < str.length() / 2; i++) {
     if (str.charAt(i) != str.charAt(lastIndex -1) {
           return false;
     }
   }
   return true;
}
```