CS 1301 - Homework 13

Due before Thursday, November 20, 1:59 *pm* on Vista (WebCT). You can resubmit your work up until the deadline.

1. Consider the game of Yahtzee. We will write a system that allows a user to play one round of Yahtzee (a real game is 13 rounds). A round consists of rolling the dice up to three times. This is a sample round:

Roll all five dice:

```
First Roll    : [1, 3, 4, 5, 5]
Kept          : []
Still Rolling : [1, 3, 4, 5, 5]
```

Specify faces to keep:

```
Which face do you want to keep(-1 to stop adding)? 5
Kept          : [5]
Still Rolling : [1, 3, 4, 5]
Which face do you want to keep(-1 to stop adding)? 5
Kept          : [5, 5]
Still Rolling : [1, 3, 4]
Which face do you want to keep(-1 to stop adding)? -1
```

2nd Roll, roll the dice that are not kept:

```
Second Roll   : [4, 5, 6]
Kept          : [5, 5]
Still Rolling : [4, 5, 6]
```

Specify face to remove from kept:

```
Which face do you want to remove from the keepers(-1 to stop removing)? -1
Kept          : [5, 5]
Still Rolling : [4, 5, 6]
```

Specify faces to keep:

```
Which face do you want to keep(-1 to stop adding)? 5
Kept          : [5, 5, 5]
Still Rolling : [4, 6]
Which face do you want to keep(-1 to stop adding)? -1
```

3rd Roll, roll the dice that are not kept:

```
Third Roll    : [2, 6]
```

Final *hand:*

```
Kept          : [2, 5, 5, 5, 6]
```

Evaluation:

```
Evaluation    : 3-OF-A-KIND, Score=23
```

When the round is over, the kept dice will be evaluated for one of these values:

Evaluation	Score
3-of-a-kind	sum of dice
4-of-a-kind	sum of dice
Full House	25
Small Straight	30
Large Straight	40
Yahtzee	50
Chance	sum of dice

See http://en.wikipedia.org/wiki/Yahtzee for a definition of these if necessary.

The only part of this system that you will write will be the methods to determine if any of the values above (3-of-a-kind, *etc.*) are true as explained in item 5 below.

You have been provided a zip file with six files:

1. yahtzee.gpj – Double-click to open the project file. You will notice the four java files in the Project pane on the left of the screen.
2. Dice.java – Do not edit this file; it is complete.
3. Referee.java – Do not edit this file; it is complete.
4. Yahtzee.java – This is the driver. It is complete. Your final system should run with this driver. You can run this right now and see that it runs. Currently, any round is evaluated as a Chance. There are two methods in main: *test* and *playGame*. You can use either to test your system.
5. YahtzeeRules.java – This file has a number of methods you will write shown below. Currently, each method simply returns *false*. You will write the code to return the correct value, depending on the *dice* input.

Method
boolean is3ofaKind(Dice dice)
boolean is4ofaKind(Dice dice)
boolean isFull House(Dice dice)
boolean isSmall Straight(Dice dice)
boolean isLarge Straight(Dice dice)
boolean isYahtzee(Dice dice)
boolean isChance(Dice dice)

Notes:

a. The *isChance* method should always return true because any hand can be considered a Chance.
b. A 4-of-a-kind is also a 3-of-a-kind.
c. A Full House is also a 3-of-a-kind
d. A Large Straight is also a Small Straight
e. A Yahtzee is a 4-of-a-kind and a 3-of-a-kind.
f. You are not concerned with picking the *best* evaluation (the Referee class does that for you). You just want to write the method above all of which return true or false. In other, if the dice faces are: 2,1,2,2,2 then both *is3ofaKind* and *is4ofaKind* will both return true, as will *isChance*.

The Dice class has a number of public methods; however you should only need these:

Evaluation	Score
int[] counts()	Returns an array of the counts of each face that has been kept (you can assume all five faces have been kept). For instance, if the faces are: 2,2,4,5,6 then counts() will return: 0,2,0,1,1,1. Note: you don't have to use this method to solve the problem, but it could be useful.
int[] getFaces()	Returns a sorted array of the faces that have been kept, *e.g.* 2,2,4,5,6.

6. DGYahtzee.class – This is a working example of how the program should play a round of Yahtzee. You must run it from the command line (java DGYahtzee).

Suggested steps to complete problem:

1. Reread this document carefully. Make sure you understand what all the evaluations are (3-of-a-kind, *etc*.).
2. Unzip the files to a folder.
3. Open a command prompt, navigate to the folder, and run DGYahtzee. Play with the game until you understand how it works.
4. Open the yahtzee.gpj project, and then open the YahtzeeRules.java class and write a method and compile.
5. Open the Yahtzee.java file and run it. Play the game until you generate a hand that your method should identify.
6. Goto Step 4.

2. You will write a GUI application that will create a circle and a rectangle and see which one has the greater area.

   a. Write a Circle class (Circle.java) that has a radius (inches) and a method to get its area.
   b. Write a Rectangle class (Rectangle.java) that has a length and width (inches) and a method to get its area.
   c. Write a GUI class that (GUI.java) allows the user to enter a radius, a length and a width. When a button is pressed, a message should be displayed which shows the dimensions and area of each shape, and which shape has greater area and by how much. You can use the template from the Ch. 7D notes.

Continued next page

3. You will write three classes:

   a. **Product.java –** This class has a constructor that takes a price and a quantity. It also has an *add* method which accepts an integer value to increase the quantity by.

   b. **Receipt.java –** This class has three constructors. One constructor takes a *Product* and a boolean value which represents if the product is on sale or not. If a product is on sale, then the total price is reduced by 10%. A second constructor just takes a *Product*. In this case, the product is not on sale. The third constructor takes no arguments. There is a method, *addProduct* which takes a *Product* as an argument and adds this product to the receipt. Note, there is only one *Product* in a *Receipt*. The class also has a method *increaseProductQuantity* which takes an integer argument that will increase the product quantity by the argument's value. Hint: use *delegation*, *e.g.* have this method call the products, *add* method. Finally, there is a *toString* method which returns the receipt. If there is no product, then the method returns:

      RECEIPT:
      There is no product for this receipt

      If the product is not on sale, the method will return, for instance:

      RECEIPT:
      Total Cost: $200.00

      If the product is on sale, the method will return, for instance:

      RECEIPT:
      Total Cost: $200.00
      Discounted Cost: $180.00

   c. **ReceiptDriver.java –** This class will test your other classes. You should use this main:

```
// Test 1
Receipt r1 = new Receipt();
System.out.println( r1 );
Product p1 = new Product( 5.00, 4 );
r1.addProduct( p1 );
System.out.println( r1 );
r1.increaseProductQuantity(3);
System.out.println( r1 );

// Test 2
Product p2 = new Product( 10.00, 6 );
Receipt r2 = new Receipt( p2 );
System.out.println( r2 );

// Test 3
Product p3 = new Product( 50.00, 5 );
Receipt r3 = new Receipt( p3, true );
System.out.println( r3 );
```