Faculty of Arts & Sciences
Department of Computer Science
CMPS 200—Introduction to Programming
Assignment 11 – Due Monday Dec 10, 2012

Notes and Announcements

Reading material:

- Chapter 8 of the text
- In addition, p. 286-290 show an example describing the use of the `throw` construct for raising exceptions. P. 184, 287, 351, and 463 show examples of Exceptions thrown in various contexts.

Exercises

1. Line Segment Data Type

Exercises 9-12 on page 575 of your textbook describe the API of a data type that represents line segments in the plane Implement the data type as a Java class, and write a client program to test your implementation. Note: You will be graded on the client test program you write, and not just on your implementation of the `Line` data type.

2. Rectangle Data Type

Exercises 13-17 on page 576 of your textbook describe the API of a data type that represents rectangles. Implement the data type as a Java class, and write a client program to test your implementation. Note: You will be graded on the client test program you write, and not just on your implementation of the `Rectangle` data type.

3. Card Data Type

A card from a standard 52-card deck of playing cards has two elements:

- a rank ("2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"); and
- a suit ("Clubs", "Diamonds", "Hearts", "Spades")

Write a class `Card` that implements the methods whose signatures are shown below. The class should have 2 private data members: `rank` and `suit` (both of type `String`).

```
public         Card (String r, String s)    // constructor
public String  rank()
public String  suit()
public boolean isOfSuit(String s)           // checks if card is of given suit
public boolean stronger(Card c)             // true if card is stronger than c
public String  toString()                   // returns a printed representation
                                            // in the form "8S", "10D", "KC",...
```

A card is stronger than another if its rank is higher. In case of equal rank, the suit determines the relative strength: Spades beat Hearts which beat Diamonds which beat Clubs.

Your code should be in a file `Card.java`. The file should also include a `main()` method to test the methods of the `Card` class. Use the `main()` below and augment it with a few additional tests.

```
public static void main(String[] args) {
    Card c1 = new Card("10", "Hearts");
    Card c2 = new Card("Q", "Spades");
    System.out.println(c1);
    System.out.println(c2);
    System.out.println(c1.isOfSuit("Hearts"));       // should print true
    System.out.println(c2.isOfSuit("Hearts"));       // should print false
    System.out.ptintln(c1.stronger(c2));             // should print false
}
```

4. Fraction (Revisited)

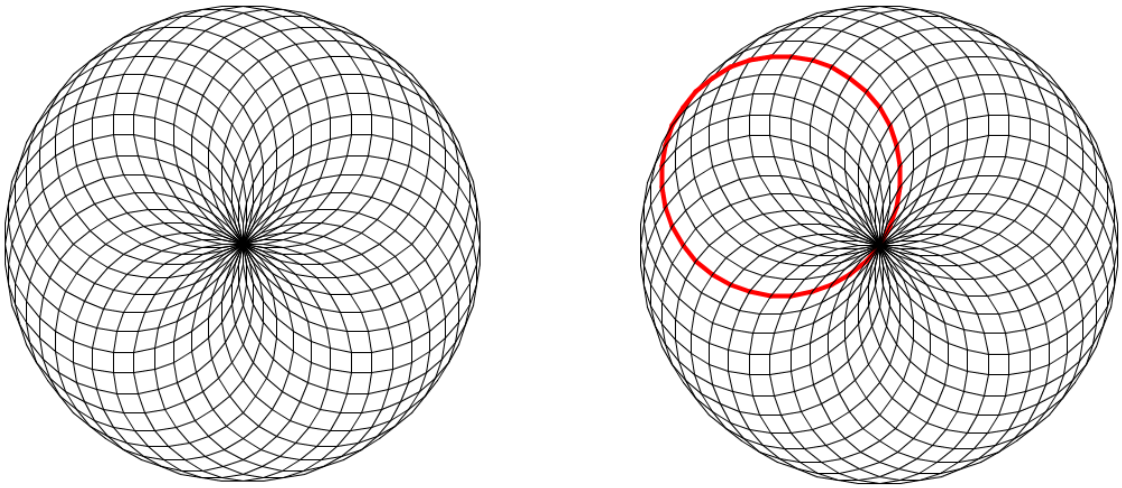
The data type `Fraction` you wrote in the previous assignment had a deficiency in that it returned fractions that were not simplified (e.g., 24/48, 128/32, etc.). Fix this problem by writing a private instance method `simplify()` that gets called as needed. Hint: the `simplify()` method will call another private method `gcd()` that computes the greatest common divisor.

You can find the greatest common divisor (gcd) of two integers x and y using *Euclid's algorithm*, which is an iterative computation based on the following observation: If x > y, then if y divides x, the gcd of x and y is y; otherwise the gcd of x and y is the same as the gcd of y and x % y.

Test your enhanced implementation of the `Fraction` data type. Create test data to test all the methods of the class. Write a client program that reads a set of fractions (pairs of integers) from standard input and computes their cumulative sum and product.

5. Turtle

Write the `Turtle` class discussed in class and use it in a program that creates the following geometric figure on the left.



The figure consists of n=36 identical circles. A circle is drawn as n turtle steps, each of length d=0.04 followed by a (360/n) degree rotation counterclockwise. Each of the n circles starts with the turtle at (0.5,0.5) initially rotated by (360/n) degrees more than the previous circle. The first circle starts with the turtle rotated 0 degrees. The figure on the right has one of the circles (the 6^{th} one) highlighted in red.

Extend `Turtle` in the following ways:

- Add color so that the path may be drawn in specified colors. Write a client to demonstrate this feature.
- Add error checking. For example, throw a `RuntimeException` with some meaningful information if the turtle goes outside the designated boundary. Write appropriate client code to demonstrate this feature.