CSCI 1101 – Winter 2013
Computer Science II
Lab No. 4

This lab is a continuation of class aggregation.

Exercise 1. Extension of the ClockDisplay

Download the NumberDisplay.java, ClockDisplay.java and ClockDemo.java programs that we discussed in the lectures. (These programs are given along side the Lab link).

Modify the `ClockDisplay` class to include a seconds display (No applets, just a text based version).

08:35:57

This can be done by include an `incrementSecond` method that increments the seconds and an `incrementHour` method that increments the hours. This is in addition to the `incrementMinute` method which was already implemented. Use a 24-hour clock format.

Note that incrementing seconds past 59 should automatically increment the number of minutes, and if in turn the number of minutes increments past 59, it should automatically increment the number of hours. For example, if the time is 11:59:59, incrementing seconds should give 12:00:00.

Test the class by incrementing hours, minutes and seconds in separate for-loops and printing the time every iteration.

Be sure to test the following cases and include the output in your submission:

a. Incrementing into the next minute
b. Incrementing into the next hour
c. Incrementing into the next day by adding minutes, hours, seconds (i.e., 23:59:00 + 1 minute)

Exercise 2. Another extension to the ClockDisplay

Modify the above program so that it displays the time in a 12-hour clock format. For example, if it is 13 hours, 14 minutes and 27 seconds in the 24-hour clock format, it should display

01:14:27 PM

If it is 1 hour, 14 minutes, and 27 seconds in the 24-hour clock format, it should display

01:14:27 AM

(No applets, just a text-based version).

Be sure to test the following cases and include the output in your submission:

a. Incrementing into the next minute
b. Incrementing into the next hour (including 12:00:00 AM to 1:00:00 AM)
c. Incrementing into the next day period, from AM to PM and from PM to AM (i.e., 11:59:59 PM to 12:00:00 AM, 11:59:59 AM to 12:00:00 PM).

Exercise 3: Create a class called Point that has the following:

- Integers x and y that represent the x and y coordinates of the point.
- Constructors and other basic methods
- Method distanceFrom(Point another) that returns the distance of this point from another point.
  (Note: Given the two points (x_1, y_1) and (x_2, y_2), the distance between these points is given by the formula:
  $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$)

  Test your methods using a client program with a main method.

  Next create a class called Circle that has the following:

- Attributes: Center of the circle (a Point) and radius (double)
- A no-arg constructor that creates a default circle with (0,0) as center and 1.0 as radius.
- A constructor that creates a circle with the specified center and radius.
- A method getArea() that returns the area of the circle.
- A method getPerimeter() that returns the perimeter of the circle.
- A method contains(Point p) that returns true if this circle contains a point p (see Figure 1).
- A method touches(Point p) that returns true if this circle touches a point p (see Figure 2).

Test all methods using a client program with a main method.

Figure 1

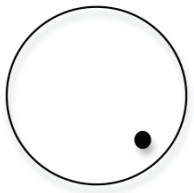

Figure 2

Hint: You can determine if a point is inside a circle by the following procedure:
Measure the distance of that point from the center. If it is more than the radius, then the point is outside; if it is less than the radius, then it is inside; if it is equal to the radius, then it touches the circle.

Note: Two interesting bonus challenges to this exercise will be revealed during the lab time. Attend the lab!