CSCI 135 Programming Exam #2
Fundamentals of Computer Science I
Fall 2012

This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #2 dropbox. Please ***double check you have submitted all the required files***.

You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.

Overview. You are painting all the walls in your new house. You need to know how many gallons of paint and how much blue painter's tape to buy at the store. You decide to write a Java program to calculate this for you. Your program will read in data about each wall in your house as well as about any windows on each wall. From this, it will calculate the area to be painted (the wall area minus the windows) and the total length of tape you need to mask the outside of the windows. Here is a sample run of the final program:

```
% java PaintCalc
Area per gallon? 400
Width of wall (<=0 if no more walls)? 24
Height of wall? 9
Width of window (<=0 if no more windows)? 3
Height of window? 4
Width of window (<=0 if no more windows)? 5.5
Height of window? 4
Width of window (<=0 if no more windows)? -1

Wall [24.00 x 9.00], paintable area = 182.00, tape length = 33.00, windows = 2
Window [3.00 x 4.00], area = 12.00, perimeter = 14.00
Window [5.50 x 4.00], area = 22.00, perimeter = 19.00
*** Total paintable area = 182.00, gallons needed = 1, tape needed = 33.00
Width of wall (negative if no more walls)? -1
```

To get started, create an empty Eclipse project and extract the contents of this zip file into your project directory: http://katie.mtech.edu/classes/csci135/paintcalc.zip

Part 1: Window. This class represents a single rectangular window. It knows things like its width and height. Width and height are floating-point values. You can assume all dimensions in this program are in the same unit of length (e.g. feet, meters, rods, it doesn't really matter). A `Window` object can do things like calculate its area, calculate its perimeter (how much tape is required to mask it off), and print out a `String` representation of itself. You need to need to implement the following public API:

```
public class Window
```

```
       Window(double w, double h)        // Create a new window of width w, height h
 double getArea()                        // Calculate the area of this window
 double getPerimeter()                   // Calculate the perimeter of this window
 String toString()                       // Return a String describing this window
```

The constructor for `Window` should throw a `RuntimeException` "Invalid window dimensions" if anybody tries to construct a `Window` with non-positive width or height. The `toString()` method should return a String formatted as shown in our example output. All floating-point output in this program should be rounded to two decimal places. We have provided a test `main()` method in the stub version of `Window.java`. Here is our test output (line numbers in `Window.java` may differ in your output):

```
% java Window
Window [4.00 x 6.00], area = 24.00, perimeter = 20.00
Window [10.50 x 5.00], area = 52.50, perimeter = 31.00
Exception in thread "main" java.lang.RuntimeException: Invalid window dimensions!
      at Window.<init>(Window.java:13)
      at Window.main(Window.java:45)
```

Part 2: Wall. This class represents a single rectangular wall. It knows things like its width and height. It also knows a list of zero or more windows that are on the wall. We assume all windows on a wall are non-overlapping . A `Wall` can do things like add a window to itself, calculate the total paintable area (the area of the wall minus the area of all windows on the wall), calculate the total length of painter's tape needed (the sum of the perimeters of all windows on the wall), and print out a `String` representation of itself.

You need to need to implement the following public API:

```
public class Wall
```

```
       Wall(double w, double h)          // Create a new wall of width w, height h
  void addWindow(double w, double h)     // Add a window of width w, height h to wall
double getPaintableArea()                // Area of the wall that requires paint
double getTapeLength()                   // Length of tape needed to mask all windows
String toString()                        // Return a String describing this wall
```

The constructor for `Wall` should throw a `RuntimeException` "Invalid wall dimensions" if anybody tries to construct a `Wall` with non-positive width or height.

The `toString()` method should return a String formatted as shown in our example output. We have provided a test `main()` method in the stub version of `Wall.java`. Here is our test output (line numbers in `Wall.java` may differ in your output):

```
% java Wall
Wall [12.00 x 9.00], paintable area = 108.00, tape length = 0.00, windows = 0

Wall [12.00 x 9.00], paintable area = 92.00, tape length = 16.00, windows = 1
Window [4.00 x 4.00], area = 16.00, perimeter = 16.00

Wall [12.00 x 9.00], paintable area = 72.60, tape length = 50.40, windows = 4
Window [4.00 x 4.00], area = 16.00, perimeter = 16.00
Window [2.00 x 1.00], area = 2.00, perimeter = 6.00
Window [3.20 x 4.50], area = 14.40, perimeter = 15.40
Window [6.00 x 0.50], area = 3.00, perimeter = 13.00

Exception in thread "main" java.lang.RuntimeException: Invalid wall dimensions!
      at Wall.<init>(Wall.java:16)
      at Wall.main(Wall.java:70)
```

Part 3: PaintCalc. This program makes use of the Wall data type to compute the house's total paintable area, the minimum number of gallons of paint needed given that area, and the total length of painter's tape required. This input comes from standard input (either interactively from the user or redirected from a file).

The program first prompts for a floating-point value representing how many square units of area a gallon of paint covers (this varies depending on what type of paint you buy). You can assume users provide a positive value for the area per gallon. Next, the program lets the user specify the details of the first wall in the house. First the user enters the width of the wall. If the width is less than or equal to zero, the program ends. Otherwise, the user enters the height of the wall:

```
Area per gallon? 300.0
Width of wall (<= 0 if no more walls)? 12
Height of wall? 9
```

After specifying a wall's width and height, the user then enters the width and height of zero or more windows. As with entry of walls, a non-positive width indicates the end of window entry for that wall. Once all windows have been entered, the program prints out the details of the wall just entered as well as the current total paintable area of all walls (rounded to two decimal places), the minimum number of gallons needed (an integer), and the amount of tape needed (rounded to two decimal places):

```
Width of window (<=0 if no more windows)? 3
Height of window? 4
Width of window (<=0 if no more windows)? -1

Wall [12.00 x 9.00], paintable area = 96.00, tape length = 14.00, windows = 1
Window [3.00 x 4.00], area = 12.00, perimeter = 14.00
*** Total paintable area = 96.00, gallons needed = 1, tape needed = 14.00
```

After the first wall, more walls and their associated window can be added. After each wall and its windows are specified, the program prints out a new total for the paintable area, gallons needs, and tape needed based on all walls entered thus far.

We have provided a number of test files. Here is the output we obtained on the command-line (note the input prompts and program output gets a bit jumbled):

```
% java PaintCalc < room1.txt
```

```
Area per gallon? Width of wall (<= 0 if no more walls)? Height of wall? Width of
window (<=0 if no more windows)?
Wall [12.00 x 9.00], paintable area = 108.00, tape length = 0.00, windows = 0
*** Total paintable area = 108.00, gallons needed = 1, tape needed = 0.00
Width of wall (<= 0 if no more walls)? Height of wall? Width of window (<=0 if no more
windows)? Height of window? Width of window (<=0 if no more windows)?
Wall [12.00 x 9.00], paintable area = 99.00, tape length = 12.00, windows = 1
Window [3.00 x 3.00], area = 9.00, perimeter = 12.00
*** Total paintable area = 207.00, gallons needed = 1, tape needed = 12.00
```

```
Width of wall (<= 0 if no more walls)? Height of wall? Width of window (<=0 if no more
windows)? Height of window? Width of window (<=0 if no more windows)? Height of
window? Width of window (<=0 if no more windows)?
Wall [24.00 x 9.00], paintable area = 156.00, tape length = 44.00, windows = 2
Window [4.00 x 5.00], area = 20.00, perimeter = 18.00
Window [8.00 x 5.00], area = 40.00, perimeter = 26.00
*** Total paintable area = 363.00, gallons needed = 2, tape needed = 56.00
Width of wall (<= 0 if no more walls)? Height of wall? Width of window (<=0 if no more
windows)? Height of window? Width of window (<=0 if no more windows)? Height of
window? Width of window (<=0 if no more windows)? Height of window? Width of window
(<=0 if no more windows)?
Wall [24.00 x 9.00], paintable area = 198.00, tape length = 30.00, windows = 3
Window [2.00 x 3.00], area = 6.00, perimeter = 10.00
Window [2.00 x 3.00], area = 6.00, perimeter = 10.00
Window [2.00 x 3.00], area = 6.00, perimeter = 10.00
*** Total paintable area = 561.00, gallons needed = 2, tape needed = 86.00
```

% java PaintCalc < room2.txt

```
Area per gallon? Width of wall (<= 0 if no more walls)? Height of wall? Width of
window (<=0 if no more windows)? Height of window? Width of window (<=0 if no more
windows)?
Wall [24.00 x 10.50], paintable area = 243.00, tape length = 12.00, windows = 1
Window [3.00 x 3.00], area = 9.00, perimeter = 12.00
*** Total paintable area = 243.00, gallons needed = 1, tape needed = 12.00
Width of wall (<= 0 if no more walls)? Height of wall? Width of window (<=0 if no more
windows)? Height of window? Width of window (<=0 if no more windows)? Height of
window? Width of window (<=0 if no more windows)?
Wall [30.00 x 10.50], paintable area = 255.00, tape length = 44.00, windows = 2
Window [4.00 x 5.00], area = 20.00, perimeter = 18.00
Window [8.00 x 5.00], area = 40.00, perimeter = 26.00
*** Total paintable area = 498.00, gallons needed = 2, tape needed = 56.00
Width of wall (<= 0 if no more walls)? Height of wall? Width of window (<=0 if no more
windows)? Height of window? Width of window (<=0 if no more windows)? Height of
window? Width of window (<=0 if no more windows)? Height of window? Width of window
(<=0 if no more windows)? Height of window? Width of window (<=0 if no more windows)?
Wall [30.00 x 10.50], paintable area = 272.06, tape length = 51.00, windows = 4
Window [2.50 x 3.50], area = 8.75, perimeter = 12.00
Window [3.25 x 3.75], area = 12.19, perimeter = 14.00
Window [4.50 x 4.00], area = 18.00, perimeter = 17.00
Window [2.00 x 2.00], area = 4.00, perimeter = 8.00
*** Total paintable area = 770.06, gallons needed = 2, tape needed = 107.00
Width of wall (<= 0 if no more walls)? Height of wall? Width of window (<=0 if no more
windows)?
Wall [24.00 x 10.50], paintable area = 252.00, tape length = 0.00, windows = 0
*** Total paintable area = 1022.06, gallons needed = 3, tape needed = 107.00
```

% java PaintCalc < room3.txt

```
Area per gallon? Width of wall (<= 0 if no more walls)?
```