CS 1331 Homework 3

Due Thursay September 13, 2012 8:00PM

Introduction

This homework will cover more on applets, writing a class, the Random class, the Math class, iteration, and more javadocing.

More Javadoc

Javadoc allows you to add additional information that the javadoc generator can use when it creates the javadoc HTML pages:

- `@author` For indicating who wrote the class
- `@version` Shows the version number for the class, and possibly the date
- `@param paramName` For explaining what a parameter means in a method or constructor. Note the parameter name is repeated in the javadoc
- `@return` For explaining what a method returns

For example, (see next page)

```
import java.lang.Math;
//Note: You do not need to import any class from java.lang -
// I am only doing it here for demonstrative purposes

/**
 * This class represents a right trapezoid
 * (Collaboration statement can go here)
 *
 * @author Bill Panfel
 * @version 1.0 9/5/2010
 */
public class RightTrapezoid
{
     private int height;
     private int base1;
     private int base2;

     /**
      * Initializes the height and two bases to the
      * inputted parameters.
      *
      * @param height The height of the trapezoid
      * @param base1 The first base of the trapezoid
      * @param base2 The second base of the trapezoid
      *
      */

     public RightTrapezoid(int height, int base1, int base2) {
          this.height = height;
          this.base1 = base1;
          this.base2 = base2;
     }
     /**
      * Calculates the area of the trapezoid by taking the
      * average of the bases and multiplying the result with the
      * height.
      *
      * @return The area of the trapezoid
      *
      */
     public double area() {
          return (double)(base1 + base2) / 2 * height;
     }
     /**
      * Calculates the perimeter of the trapezoid by taking the
      * sum of the height, the two bases, and the length of the
      * fourth side.
```

```
     *
     * @return The perimeter of the trapezoid
     *
     */

    public double perimeter() {
         return base1 + base2 + height +
              Math.sqrt(height * height + (base1 – base2) *
              (base1 – base2));
    }
    /**
     * Gets the height of the trapezoid
     *
     * @return The height of the trapezoid
     *
     */
    public int getHeight() {
         return height;
    }
    /**
     * Sets the height of the trapezoid
     *
     * @param height The height to set of the trapezoid
     *
     */
    public void setHeight(int height) {
         this.height = height;
    }
}
```

Note these features about javadocs:

- They always begin with /**
- The class javadoc goes between the imports and public class line
- The method javadocs go above the method they are javadocing
- The @param tag is followed by the name of the variable it describes
- The @return tag is not followed by a variable
- There are no @param tags when the method had no parameters
- There is no @return tag when the method does not return
- The description of the method / class is followed by the tags (that is, no “body” text appears below any tag)

To get your javadoc to include the @author and @version tags, execute:

```
javadoc *.java –author –version
```

The @return and @param tags are automatically included in the javadoc. If you wish to create the javadoc html files in a separate folder (there are several html files) add a "`-d <directory>`" tag to the command.

3.1 Dart Class

Create a class called `Dart` that represents a dart. It should know its location, how to draw itself, and how to calculate distance between darts.

1. Make instance variables for the dart. It should have an int x and an int y location.
2. Create the following methods:
    a. A constructor that sets the x and y location of the dart:

    ```
    public Dart(int x, int y) { … }
    ```

    b. A distance method that calculates the distance between this dart and some other coordinate (look up the formula for distance if you do not remember it:

    ```
    public double distance(int x, int y) { … }
    ```

    c. A draw method that draws the dart:

    ```
    public void draw(Graphics g) { … }
    ```

3. Javadoc the class and methods using the tags described in the introduction and homework 2. In the future we will not explicitly tell you to javadoc your files, but it will always be expected.

3.2 Pi Applet

Create a java applet called `PiApplet` that will approximate pi by throwing darts at a circle. To accomplish this, follow these steps:

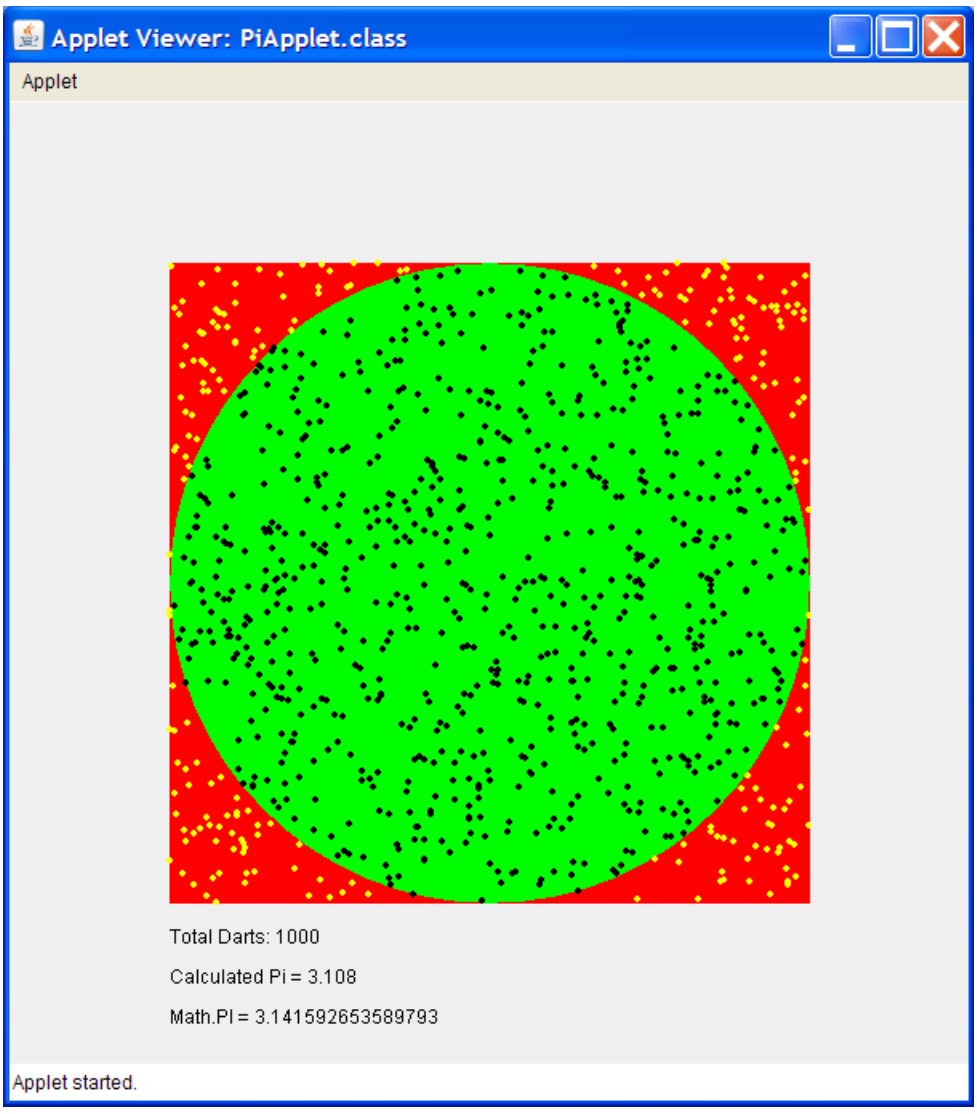
1. Draw a circle inscribed in a square (there is a picture later in the instructions)
2. Create a "center dart" to keep track of the center of your dart board
3. Create a constant that holds the total number of darts to be thrown
4. Create a random number generator
5. Create a variable to keep track of how many darts landed inside the circle
6. Start a for loop. On each iteration:
    a. Create a new dart at a random x and y location inside the square
    b. Calculate the distance to the center. If the distance is less than or equal to the radius of the circle, the dart landed inside the circle.
    c. If the dart landed inside the circle, increment the variable that holds how many darts landed inside. Change the color depending on whether the dart landed inside or outside the circle.
    d. Draw the dart
7. Calculate an approximate value of pi using this formula:

    Hit Percentage = number of hits / total number of darts thrown
    Calculated Pi = 4 * Hit Percentage

Derived from Area of Square = d^2 = 4*r^2
Area of Circle = pi*r^2

8. Display the total number of darts, the calculated value of pi, and the actual value of pi
9. Also, create the necessary HTML page to view the applet. Name this file `PiApplet.html`. Remember, you should write this yourself using a plaintext editor and test it using `appletviewer`.
10. `Experiment with the values to get a better approximation,`

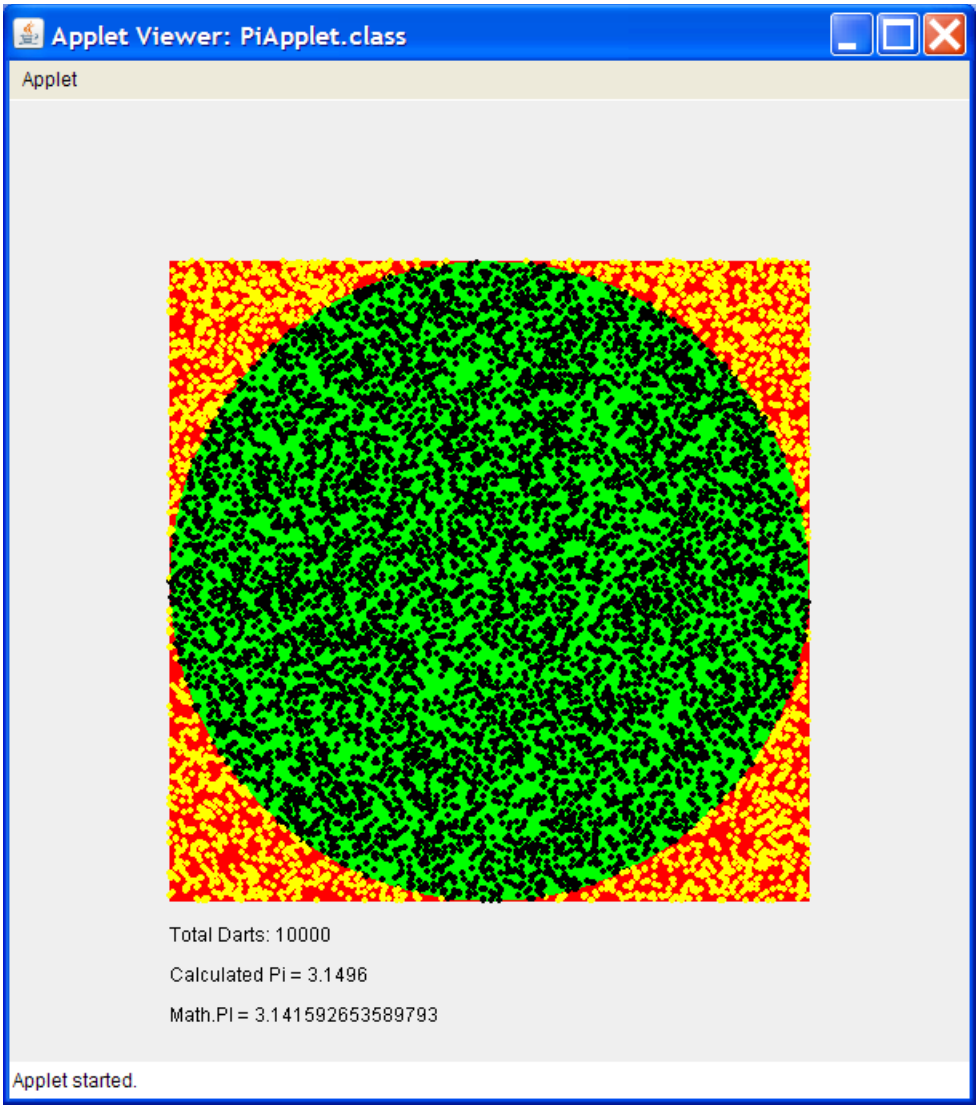
Your applet might look something like this:



Dimensions used in above example:

- Window size: 600 x 600 pixels
- Square size: 400 x 400 pixels
- Offset of square from each edge: 100 pixels
- Circle radius: 200 pixels (should always be half the square side)
- Dart radius: 2 pixels
- Color of square: red
- Color of circle: green
- Color of darts inside circle: black
- Color of darts outside circle: yellow

Another example using more darts:



Turn-in Procedure

Turn in the following files on T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft.

- Dart.java
- PiApplet.java
- PiApplet.html

All .java files should have a descriptive javadoc comment.

Verify the Success of Your HW Turn-In

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**
    (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework (past the grace period of 2 am) will not be accepted regardless of excuse. Treat the due date with respect. The real due date and time is 8 pm Thursday. Do not wait until the last minute!