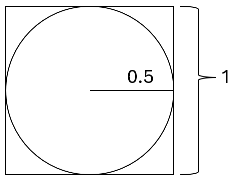
Tim Jackman
July 8, 2025

Activity - Drawing in Java

In this activity comes in two parts. In the first part, we will use Java to simulate an experiment to estimate the value of π. In the second part, we will use a *library* developed for the textbook Introduction to Programming in Java, to visualize the experiment.

1 Estimating π

In this section we will estimate π using randomness in Java. Take a circle centered at $(0.5, 0.5)$ with radius 0.5, and circumscribe a square of side length 1 around it:



The area of the square is $1 \times 1 = 1$ and the area of the circle is $\pi \times (0.5)^2 = \frac{\pi}{4}$. If we take a *uniformly random* point in the square (i.e. every point is equally likely), then the probability the point is within the circle is given by $\frac{\text{area of Circle}}{\text{area of Square}} = \frac{\frac{\pi}{4}}{1} = \frac{\pi}{4}$. Rearranging terms, we get that $\pi = 4 \times (\text{probability a random point is in Circle})$ We can estimate this probability experimentally in Java by picking many points at random and counting how many of our points end up in the circle out of the total number of points.

Setup. Create a folder called EstimatingPi in your Documents folder. Open this folder in VS Code and create a EstimatingPi.java file. Define the class and a *stub* main method (i.e. nothing within it).

1. Write a method `inCircle` that takes in two doubles x and y returns whether or not the point (x, y) is within the circle. Recall that a circle of radius r centered at (a, b) contains all points such that $(x-a)^2 + (y-b)^2 \leq r^2$. You can use $Math.pow(x, k)$ to compute x^k.

2. Write a method `getPoints` that takes in an integer n and returns a $2 \times n$ multi-dimensional double array consisting of n randomly selected points. For instance, `getPoints(10)` generates 10 random points, `getPoints(10)[0][3]` is the x coordinate of the fourth point, and `getPoints(10)[1][3]` is the y coordinate of the fourth point. You will need to import and use the `java.util.Random` class:

```java
// Instantiates a source of randomness we can reuse to generate random numbers
Random random = new Random();

// Generates a random double between 0 and 1
double randomDouble = random.nextDouble();
```

3. Write a method `estimatePi` that takes in a $2 \times n$ multi-dimensional double array and returns an estimate of π as a double. You should use any methods we defined above that are appropriate for this task.

4. Using the methods defined above, write a main method that takes an int from the command line arguments and prints out an estimate of π using that number of points.

2 Visualizing the Experiment

In this section we are going to use StdDraw.java to draw a picture of the experiment. You can read StdDraw's documentation here.

Setup. Download StdDraw.java from here, and place it in your EstimatingPi folder (ctrl + S to save the file).

1. In your main method, draw the unit square and circle. To make the square visible, set your pen size to 0.005 and make the square slightly smaller than 1 unit across (0.999 works well). Use the same thickness for the circle.

2. Set your pen size to 0.001 and change the color to green. Have your main method plot every randomly generated point.

3. Challenge: Use a different pen color for points that fall within the circle.

4. Challenge: Include the estimate of π on the sketch. To make it really look nice, try making the window taller than wide so you can put the estimate above the square. You may need to adjust the scale of the canvas to keep the ratios correct.