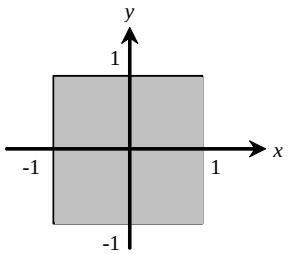
Case Study: Monte Carlo Simulation

Monte Carlo simulation uses random numbers and probability to solve problems. This method has a wide range of applications in computational mathematics, physics, chemistry, and finance. This section gives an example of using Monte Carlo simulation for estimating π.

To estimate π using the Monte Carlo method, draw a circle with its bounding square as shown below.



Assume the radius of the circle is `1`. Therefore, the circle area is π and the square area is `4`. Randomly generate a point in the square. The probability for the point to fall in the circle is `circleArea / squareArea = π / 4`.

Write a program that randomly generates 1,000,000 points in the square and let `numberOfHits` denote the number of points that fall in the circle. Thus, `numberOfHits` is approximately `1000000 * (π / 4)`. π can be approximated as `4 * numberOfHits / 1000000`. The complete program is shown in Listing 1.

Listing 4.1 MonteCarloSimulation.java

```
1  public class MonteCarloSimulation {
2    public static void main(String[] args) {
3      final int NUMBER_OF_TRIALS = 10000000;
4      int numberOfHits = 0;
5
6      for (int i = 0; i < NUMBER_OF_TRIALS; i++) {
7        double x = Math.random() * 2.0 - 1;
8        double y = Math.random() * 2.0 - 1;
9        if (x * x + y * y <= 1)
10         numberOfHits++;
11     }
```

```
12
13      double pi = 4.0 * numberOfHits / NUMBER_OF_TRIALS;
14      System.out.println("PI is " + pi);
15    }
16  }
```

<output>

```
PI is 3.14124
```

<end output>

<margin note (line 7)>generate random points

<margin note (line 9)>check inside circle

<margin note (line 13)>estimate pi

<end listing 1>

The program repeatedly generates a random point (**x**, **y**) in the square in lines 7–8:

```
double x = Math.random() * 2.0 - 1;
double y = Math.random() * 2.0 - 1;
```

If $x^2 + y^2 \leq 1$, the point is inside the circle and **numberOfHits** is incremented by **1**. π is approximately **4 * numberOfHits / NUMBER_OF_TRIALS** (line 13).