Random Numbers

Cody Lewis

Assignment Definition And General Feedback By Michael Panitz
at Cascadia Community College (http://www.cascadia.edu)

Table of contents:

Quick Summary:

While programming with Java, you will be required form time to time to work around or build a certain level of mystery in your programs. Sometimes you won't always want the city and its inhabitants in the exact same place every single time. A random number generator is a simple tool to give your program a degree of uncertainty, and can be used to test your programs to the fullest.

When To Use This / Avoid This:

You will want to use random numbers whenever you want a variance or even level of mystery in your programs.

If you want your program to be exactly the same every time you run it, definitely want to avoid using random numbers.

Example Of Usage:

The first step to implementing random numbers is to import the Java utility file:

```
    import java.util.Random;
```

or

```
    import java.util.*;
```

This will define what the generator should do and allow you to construct it. The simplest constructor to use is:

```
    Random generator = new Random();
```

The name “generator” can be replaced with any name so long as it is consistently used throughout the whole program. In BIT115 we will be using the name “numberGenerator” as a more descriptive and memorable name.

Line	**Code**
1.	`import becker.robots.*;`
2.	`import java.util.Random;`
3.	`public class ICE_04_Demo_01 extends Object`
4.	`{`
5.	`public static void main(String[] args)`
6.	`{`
7.	`City bothell = new City();`
8.	`Random numberGenerator = new Random();`
9.	`if( numberGenerator.nextInt(100) < 50    )`
10.	`{`
11.	`new Thing(bothell, 2,1);`
12.	`}`
13.	`if (numberGenerator.nextInt(100) < 20)`
14.	`{`
15.	`new Thing(bothell, 2,2);`
16.	`}`
17.	`}`
18.	`}`

In this program, random numbers are combined with if statements to decide if something will or will not happen. In this case, the random number generated will decide if a thing is placed or not. The line, `if( numberGenerator.nextInt(100) < 50),` means that a random number between 1 and 100 is generated, and if that number is less than 50, the statement will return true. Essentially, this creates a 50% chance of the thing being placed.

Example Trace:

In a nutshell, this technique allows you to add a bit of unpredictability to your programs. In this run of the program, the first number randomly selected was less than 50, and the second number generated was greater than 20. This causes line 9 to return true and line 13 to return false, causing the program to place a thing at (2,1) but not (2,2). Whenever you perform a trace using random numbers, always trace whatever happens when you run the program one time.

In order to go over these details more thoroughly, here is a (partial) trace of the above program.

Line#	Program Statement	Thing 1	Thing 2
1	`import becker.robots.*;`	-	-
2	`import java.util.Random;`	-	-
3	`public class Random_Numbers extends Object`	-	-
5	`public static void main(String[] args)`	-	-
7	`City bothell = new City();`	-	-
8	`Random numberGenerator = new Random();`	-	-
9	`if( numberGenerator.nextInt(100) < 50    )`	-	-
11	`new Thing(bothell, 2,1);`	(2,1)	-
13	`if (numberGenerator.nextInt(100) < 20)`	-	-
15	`new Thing(bothell, 2,2);`	-	-

Syntax Explanation:

The syntax for using random number generators is actually pretty simple. To create a RNG, `Random numberGenerator = new Random();` is used. numberGenerator is simply the name chosen for this program's RNG and can be changed to any name so long as it is used throughout consistently. `Random` is a premade class that is set up inside of the `import java.util.Random;`. To use the RNG, you call the name just like you would any other command followed by `.nextInt`, meaning next integer, telling it to come up with an integer between the set parameters. The parameters used here is (100), meaning any number between 1 and 100.

Important Details:

In Java, the random number class isn't truly random. It is actually based off of an algorithm which in turn is based off a "seed." The seed itself is a number that is ran through the algorithm, so if you know the seed that the numbers are based off of, you will know exactly what numbers will be produced, and in what order. However, for our intents and purposes the random class will work just fine.

Licensing

Plagiarism