Building Java Programs

Chapter 5
Lecture 5-2: Random Numbers

reading: 5.1 - 5.2

self-check: #8 - 17

exercises: #3 - 6, 10, 12

videos: Ch. 5 #1-2

The `Random` class

- A `Random` object generates pseudo-random* numbers.
  - Class `Random` is found in the `java.util` package.

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

Method name	**Description**
`nextInt()`	returns a random integer
`nextInt(`**max**`)`	returns a random integer in the range [0, *max*) in other words, 0 to *max*-1 inclusive
`nextDouble()`	returns a random real number in the range [0.0, 1.0)

- Example:

```
Random rand = new Random();
int randomNumber = rand.nextInt(10);    // 0-9
```

Generating random numbers

- Common usage: to get a random number from 1 to *N*

```
int n = rand.nextInt(20) + 1;    // 1-20 inclusive
```

- To get a number in arbitrary range [*min*, *max*] inclusive:

  `nextInt(`**size of range**`) + `**min**

  - where (**size of range**) is (**max** `-` **min** `+ 1`)

- Example: A random integer between 4 and 10 inclusive:

```
int n = rand.nextInt(7) + 4;
```

Random questions

- Given the following declaration, how would you get:

```
Random rand = new Random();
```

  - A random number between 1 and 100 inclusive?

```
int random1 = rand.nextInt(100) + 1;
```

  - A random number between 2 and 4 inclusive?

```
int random2 = rand.nextInt(3) + 2;
```

  - A random number between 50 and 100 inclusive?

```
int random3 = rand.nextInt(51) + 50;
```

`Random` and other types

- `nextDouble` method returns a `double` between 0.0 - 1.0
  - Example: Get a random value between 2.0 and 6.0:

    ```
    double r = rand.nextDouble() * 4.0 + 2.0;
    ```

- Any finite set of possible values can be mapped to integers
  - code to randomly play Rock-Paper-Scissors:

    ```
    int r = rand.nextInt(3);
    if (r == 0) {
        System.out.println("Rock");
    } else if (r == 1) {
        System.out.println("Paper");
    } else {
        System.out.println("Scissors");
    }
    ```

`Random` question

- Write a program that simulates rolling of two 6-sided dice until their combined result comes up as 7.

```
2 + 4 = 6
3 + 5 = 8
5 + 6 = 11
1 + 1 = 2
4 + 3 = 7
You won after 5 tries!
```

Random answer

```
// Rolls two dice until a sum of 7 is reached.
import java.util.*;

public class Dice {
    public static void main(String[] args) {
        Random rand = new Random();
        int tries = 0;

        int sum = 0;
        while (sum != 7) {
            // roll the dice once
            int roll1 = rand.nextInt(6) + 1;
            int roll2 = rand.nextInt(6) + 1;
            sum = roll1 + roll2;
            System.out.println(roll1 + " + " + roll2 + " = " + sum);
            tries++;
        }

        System.out.println("You won after " + tries + " tries!");
    }
}
```

Random question

- Write a multiplication tutor program.
  - Ask user to solve problems with random numbers from 1-20.
  - The program stops after an incorrect answer.

```
14 * 8 = 112
Correct!
5 * 12 = 60
Correct!
8 * 3 = 24
Correct!
5 * 5 = 25
Correct!
20 * 14 = 280
Correct!
19 * 14 = 256
Incorrect; the answer was 266

You solved 5 correctly
Last correct answer was 280
```

  - The last line should not appear if the user solves 0 correctly.

Random answer

```
import java.util.*;

// Asks the user to do multiplication problems and scores them.
public class MultiplicationTutor {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        Random rand = new Random();

        // fencepost solution - pull first question outside of loop
        int correct = 0;
        int last = askQuestion(console, rand);
        int lastCorrect = 0;

        // loop until user gets one wrong
        while (last > 0) {
            lastCorrect = last;
            correct++;
            last = askQuestion(console, rand);
        }

        System.out.println("You solved " + correct + " correctly");
        if (correct > 0) {
            System.out.println("Last correct answer was " + lastCorrect);
        }
    }
    ...
```

Random answer 2

```
...

// Asks the user one multiplication problem,
// returning the answer if they get it right and 0 if not.
public static int askQuestion(Scanner console, Random rand) {
    // pick two random numbers between 1 and 20 inclusive
    int num1 = rand.nextInt(20) + 1;
    int num2 = rand.nextInt(20) + 1;

    System.out.print(num1 + " * " + num2 + " = ");
    int guess = console.nextInt();
    if (guess == num1 * num2) {
        System.out.println("Correct!");
        return num1 * num2;
    } else {
        System.out.println("Incorrect; the correct answer was " +
                           (num1 * num2));
        return 0;
    }
}
}
```

A Big Deal

- Some reasons why computers have changed all of science, engineering, sociology, politics, economics, …
  - They can *process* tons of data quickly
  - They can *generate* tons of data quickly
    - Example: Roll dice 10 million times

- Data generation often requires simulating a process with randomness
  - Because some things (e.g., dice rolls) are random
  - Because some things (e.g., disease causes) may not be random, but it’s the best guess we have
    - X% probability of cancer if you smoke

Known vs. unknown solutions

- Sometimes mathematicians have discovered a formula that gives an exact answer to a probability problem
  - Example: Probability two dice sum to 7

- But for more complicated problems sometimes no human knows!
  - "Next best thing": Try it a lot of times and measure the result
    - Use a computer because it's faster
  - Can be easier and more convincing than the math even when a formula is known

Two Examples

1. Playing roulette with a particular betting strategy
   - It turns out a formula exists (it's a random walk), but programming a simulation is easy
   - And simulation handles "can't bet more than you have"

2. UrbanSim
   - Simulating the inter-related effects of land use and transportation decisions, and their environmental impact
   - Much more complicated than gambling!

Roulette conclusions

- Bet small to play longer
- Bet big to increase your chances of winning
  - Best is all at once: 48.3%
- “Can’t bet more than you have” rule leads to surprising results:
  - Given $1000, better off betting $500 than $990
- But more importantly, we learned all this from simulation!
  - But always make sure your code is right!