Building Java Programs

Chapter 5:
Program Logic and Indefinite Loops

Lecture 5-2: Random Numbers and Boolean Logic

The Big Picture

- Many interesting programs require random behavior
  - dealing out cards for solitaire
  - simulating processes like evolution or the stock market
- We often want methods that answer questions
  - did the user get a new high score?
  - is the password the user has entered correct?
  - did the user/computer get a royal flush?

Random numbers

reading: 5.1

self-check: #8 - 10
exercises: #3 - 6, 10

The Random class

- Random objects generate 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);
// randomNumber has a random value between 0 and 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]:

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

  where ***<size of range>*** is ***<max>*** `-` ***<min>*** `+ 1`

  - Example: A random integer between 5 and 10 inclusive:

```
int n = rand.nextInt(6) + 5;
```

Pseudo-random?!

- The numbers are generated algorithmically
  - there's some formula which takes a **seed** as input
  - a good seed might be the position of the mouse or the time
  - the seed has a finite number of possible values

- Generating truly random numbers
  - point a camera at a lava lamp
  - measure radioactive decay

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 50 and 100 inclusive?

```
int random2 = rand.nextInt(51) + 50;
```

  - A random number between 4 and 17 inclusive?

```
int random3 = rand.nextInt(14) + 4;
```

Random: not only for ints

- Often, the values that need to be generated aren't numeric
  - 5 cards to deal out for poker
  - a series of coin tosses
  - a day of the week to assign a chore

- The 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 double values

- `nextDouble` method returns a `double` between 0.0 - 1.0
  - Example: Get a random GPA value between 1.5 and 4.0:

```
double randomGpa = rand.nextDouble() * 2.5 + 1.5;
```

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 Roll {
    public static void main(String[] args) {
        Random rand = new Random();
        int sum = 0;
        int tries = 0;
        while (sum != 7) {
            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!");
    }
}
```

Boolean logic

reading: 5.2

self-check: #11 - 17
exercises: #12

Type boolean

- **`boolean`**: Represents logical values of `true` or `false`.
  - A ***<condition>*** in an `if`, `for`, `while` is a `boolean` expression.

```
boolean minor = (age < 21);
boolean expensive = (iPhonePrice > 200.00);
boolean iLoveCS = true;

if (minor) {
    System.out.println("Can't purchase alcohol!");
}
if (iLoveCS || !expensive) {
    System.out.println("Buying an iPhone");
}
```

  - You can create `boolean` variables, pass `boolean` parameters, return `boolean` values from methods, ...

Methods that return boolean

- Methods can return `boolean` values.
  - A call to such a method can be a loop or `if`'s ***<test>***.

```
Scanner console = new Scanner(System.in);
System.out.print("Type your name: ");
String line = console.nextLine();

if (line.startsWith("Dr.")) {
    System.out.println("Will you marry me?");
} else if (line.endsWith(", Esq.")) {
    System.out.println("And I am Ted 'Theodore' Logan!");
}
```

Writing boolean methods

```
public static boolean bothOdd(int n1, int n2) {
    if (n1 % 2 != 0 && n2 % 2 != 0) {
        return true;
    } else {
        return false;
    }
}
```

- Calls to this methods can now be used as tests:

```
if (bothOdd(7, 13)) {
    ...
}
```

"Boolean Zen"

- Methods that return a `boolean` result often have an `if/else` statement:

```
public static boolean bothOdd(int n1, int n2) {
    if (n1 % 2 != 0 && n2 % 2 != 0) {
        return true;
    } else {
        return false;
    }
}
```

- ... but the `if/else` is sometimes unnecessary.
  - The `if/else`'s condition is itself a `boolean` expression; its value is exactly what you want to return. So do that!

```
public static boolean bothOdd(int n1, int n2) {
    return (n1 % 2 != 0 && n2 % 2 != 0);
}
```

"Boolean Zen" template

- Replace:

```
public static boolean <name>(<parameters>) {
    if (<condition>) {
        return true;
    } else {
        return false;
    }
}
```

- with:

```
public static boolean <name>(<parameters>) {
    return <condition>;
}
```

Random/while question

- Write a multiplication tutor program.
  - Use a static method that returns a boolean value.
  - Test multiplication of numbers between 1 and 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.
```

Random/while answer

```
import java.util.*;

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

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

        System.out.println("You solved " + correct + " correctly.");
    }

    ...
```

Random/while answer 2

```
    ...

    // Asks the user one multiplication problem,
    // returning true if they get it right and false if not.
    public static boolean 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 true;
        } else {
            System.out.println("Incorrect; the correct answer was " +
                               (num1 * num2));
            return false;
        }
    }
}
```

boolean questions

- Modify our previous *Primes* program to use a `isFactor` method rather than a `countFactors` method.
  - Example output of primes up to 50:

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

- Write methods with return values to tell whether two words rhyme and/or alliterate.
  - Example log of execution:

```
Type two words: car STAR
They rhyme but don't alliterate.
```

boolean answer

```
// Determines whether two words rhyme and/or start with the same letter.
import java.util.*;

public class Rhyme {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Type two words: ");
        String word1 = console.next();
        String word2 = console.next();

        if(rhyme(word1, word2) && alliterate(word1, word2)) {
            System.out.println("They rhyme and alliterate");
        } else if(rhyme(word1, word2)) {
             System.out.println("They rhyme but don't alliterate");
        } else if(alliterate(word1, word2)) {
            System.out.println("They alliterate but don't rhyme");
        } else {
            System.out.println("They don't rhyme or alliterate");
        }
    }
    ...
```

boolean answer, continued

```
    // Returns true if s1 and s2 end with the same two letters.
    public static boolean rhyme(String s1, String s2) {
        return s2.length() >= 2 &&
               s1.endsWith(s2.substring(s2.length() - 2));
    }

    // Returns true if s1 and s2 start with the same letter.
    public static boolean alliterate(String s1, String s2) {
        return s1.startsWith(s2.substring(0, 1));
    }
}
```