Randomness and Returns

Announcements

- Assignment 2 (**Welcome to Java!**) was due today at 3:15PM.
- Assignment 3 (**Problem-Solving in Java**) goes out now and is due next Monday at 3:15PM.
  - Play around with `while` loops, methods, random numbers, and variable assignment!
- CS for Social Good:
  - cs-for-social-good@lists.stanford.edu
  - Facebook: *Stanford CS for Social Impact*.

Outline for Today

- **Random Numbers**
  - Randomness meets computing.
- **The Loop-and-a-Half Idiom**
  - A particularly clever loop structure.
- **Returning Values**
  - Communicating information out of methods.

Randomness and Computing

Random Number Generators

RandomGenerator

- The class `RandomGenerator` acts as a random number generator. To use it, you'll need to `import acm.util.*;`
- To generate random numbers, start by getting a random generator:

```
RandomGenerator rgen = RandomGenerator.getInstance();
```

- Then, use the `next`***X*** functions to get the random values you want:
  - `rgen.nextInt(`***low***`, `***high***`)`
  - `rgen.nextDouble(`***low***`, `***high***`)`
  - `rgen.nextBoolean(`***probability***`)`
  - `rgen.nextColor() // Ooh, shiny!`

Looping Forever

- `while` loops iterate as long as their condition evaluates to `true`.
- A loop of the form `while (true)` will loop forever (unless something stops it).

```
while (true) {
    …

}
```

- You can immediately exit a loop by using the `break` statement.

Looping Forever

- `while` loops iterate as long as their condition evaluates to `true`.
- A loop of the form `while (true)` will loop forever (unless something stops it).

```
while (true) {
    …
    if ( … ) break;
}
```

- You can immediately exit a loop by using the `break` statement.

The “Loop-and-a-Half” Idiom

- Often you will need to
  - read a value from the user,
  - decide whether to continue, and if so
  - process the value.
- Technique: The ***loop-and-a-half idiom***:

```
while (true) {
    /* … get a value from the user … */

    if (condition) {
        break;
    }

    /* … process the value … */
}
```

Methods that Return Values

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

Returning Values

- Methods can return values that can be used elsewhere in the program.
- Examples:
  - The `getWidth` and `getHeight` methods *return* the width and height of the window.
  - The `readInt` and `readDouble` methods *return* values entered by the user.
- You can write your own methods that return values!

Return Syntax

- To make a method that communicates a value to the outside world, you need to do two things.
- First, say what kind of value you want to communicate back by specifying a return type. Declare your method as

```
private returnType methodName(parameters)
```

- Then, include a `return` statement in your method saying what value to hand back.

```
return value;
```

Factorials!

- The number ***n* factorial**, denoted ***n*!**, is

$$1 \times 2 \times 3 \times \ldots \times (n - 1) \times n$$

- For example:
  - $3! = 1 \times 2 \times 3 = 6$.
  - $5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$
  - $0! = 1$ (by definition)
- Factorials arise surprisingly frequently in computer science:
  - Determining how quickly computers can sort a list of values.
  - Analyzing the efficiency of various algorithms.