CS 106A, Lecture 6
More Loops;
Scope and Constants;
Random Numbers

reading:

Art & Science of Java, 4.5, 3.2

Cumulative loops

```
int sum = 0;
for (int i = 1; i <= 1000; i++) {
    sum = sum + i;
}
println("The sum is " + sum);
```

- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
  - The `sum` in the above code is a cumulative sum.
  - Cumulative sum variables must be declared *outside* the loops that update them, so that they will still exist after the loop.

Sentinel loops

SentinelSum

- **sentinel**: A value that signals the end of user input.
  - **sentinel loop**: Repeats until a sentinel value is seen.

- Example: Write a program that prompts the user for numbers until the user types 0, then output the sum of the numbers.
  - In this case, 0 is the sentinel value.

```
Type a number: 10
Type a number: 20
Type a number: 30
Type a number: 0
Sum is 60
```

Sentinel solution?

- This solution *seems to* work just fine ...

```
int sum = 0;
int n = 1;          // "dummy" value, anything but 0
while (n != 0) {
    n = readInt("Type a number: ");
    sum += n;
}
println("Sum is " + sum);
```

- Example output:

```
Type a number: 10
Type a number: 20
Type a number: 30
Type a number: 0
Sum is 60
```

Incorrect solution

- Change the sentinel to -1. The solution now fails. Why?

```
int sum = 0;
int n = 1;          // "dummy" value, anything but -1
while (n != -1) {
    n = readInt("Type a number: ");
    sum += n;
}
println("Sum is " + sum);
```

- Example output:

```
Type a number: 10
Type a number: 20
Type a number: 30
Type a number: -1
Sum is 59
```

Sentinel fix #1

- Prompt for the first number outside the `while` loop.
  - This is really a fencepost problem. Move a "post" (prompt) out.
  - Reverse the order of the two statements in the `while` loop.

```
int sum = 0;
int n = readInt("Type a number: ");
while (n != -1) {
    sum += n;
    n = readInt("Type a number: ");
}
println("Sum is " + sum);
```

Sentinel fix #2

- For this particular problem, simply initializing the sum to 0 will work because the 0 gets added to the sum and doesn't affect it.
  - Would not work for some other problems, e.g. finding max/min value

```
int sum = 0;
int n = 0;    // must be 0 to avoid corrupting sum
while (n != -1) {
    sum += n;
    n = readInt("Type a number: ");
}
println("Sum is " + sum);
```

Sentinel fix #3

- A `while (true)` loop continues until it is manually stopped using a command called `break`.
  - Sometimes called an *infinite loop*, *forever loop*, or *loop-and-a-half*

```
int sum = 0;
while (true) {
    int n = readInt("Type a number: ");
    if (n == -1) {
        break;          // exit the loop
    }
    sum += n;
}
println("Sum is " + sum);
```

Nested Loops

Nested loops

starsPrint

- **nested loop**: A loop placed inside another loop.

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; j++) {
        print("*");
    }
    println();    // to end the line
}
```

- Output:

```
**********
**********
**********
**********
**********
```

- The outer loop repeats 5 times; the inner one 10 times.

Nested loop question

numberLoops

- **Q:** What output is produced by the following code?

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        print("*");
    }
    println();
}
```

```
A.            B.            C.            D.            E.
  *****         *****         *             1             12345
  *****         ****          **            22
  *****         ***           ***           333
  *****         **            ****          4444
  *****         *             *****         55555
```

(How would you modify the code to produce each output above?)

Nested loop question 2

numberLoops2

- How would we produce the following output?

```
....1
...22
..333
.4444
55555
```

- Answer:

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 5 - i; j++) {
        print(".");
    }
    for (int j = 1; j <= i; j++) {
        print(i);
    }
    println();
}
```

Nested loop question 3

numberLoops3

- How would we produce the following output?

```
....1
...2.
..3..
.4...
5....
```

- Answer:

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 5 - i; j++) {
        print(".");
    }
    print(i);
    for (int j = 1; j <= i - 1; j++) {
        print(".");
    }
    println();
}
```

Variable Scope and Constants

Limitations of variables

- Idea: Make a variable to represent the size.
  - Use the variable's value in the methods.

- Problem: A variable in one method can't be seen in others.

```
public void run() {
    int size = 4;
    topHalf();
    bottomHalf();
}

public void topHalf() {
    for (int i = 1; i <= size; i++) {    // ERROR: size not found
        ...
    }
}

public void bottomHalf() {
    for (int i = size; i >= 1; i--) {    // ERROR: size not found
        ...
    }
}
```

Scope

- **scope**: The part of a program where a variable exists.
  - From its declaration to the end of the `{ }` braces
    - A variable declared in a `for` loop exists only in that loop.
    - A variable declared in a method exists only in that method.

```
public void example() {
    int x = 3;
    for (int i = 1; i <= 10; i++) {
        println(x);
    }
    // i no longer exists here
} // x ceases to exist here
```

i's scope

x's scope

Scope implications

- Variables without overlapping scope can have same name.

```
for (int i = 1; i <= 100; i++) {
    print("/");
}
for (int i = 1; i <= 100; i++) {   // OK
    print("\\");
}
int i = 5;                         // OK: outside of loop's scope
```

- Can't declare a variable twice in same scope, or use it out of scope.

```
for (int i = 1; i <= 100 * line; i++) {
    int i = 2;                     // ERROR: overlapping scope
    print("/");
}
i = 4;                             // ERROR: outside scope
```

Class constants

- **class constant**: A fixed value visible to the whole program.
  - value can be set only at declaration; cannot be reassigned

- Syntax:

  `private static final` ***`type name`*** `=` ***`value`***`;`

  - name is usually in ALL_UPPER_CASE

  - Examples:

```
private static final int DAYS_IN_WEEK = 7;
private static final double INTEREST_RATE = 3.5;
private static final int SSN = 658234569;
```

Nested loop w/ constant

- Make our nested-loop code use a constant for the output's size:

```
....1  (size 5)         ..1  (size 3)         ...1  (size 4)
...2.                   .2.                   ..2.
..3..                   3..                   .3..
.4...                                         4...
5....
```

- Answer:

```
for (int i = 1; i <= SIZE; i++) {
    for (int j = 1; j <= SIZE - i; j++) {
        print(".");
    }
    print(i);
    for (int j = 1; j <= i - 1; j++) {
        print(".");
    }
    println();
}
```

Random Numbers (in brief)

RandomGenerator

- `import acm.util.*;`

Method	Description
`RandomGenerator.getInstance()` `.nextInt(`***`min`***`,` ***`max`***`)`	a random integer in the given range, inclusive

```
// random number from 0-9 inclusive
int rdigit = RandomGenerator.getInstance().nextInt(0, 9);
println(rigit);

// print "hello! between 3-6 times
int times = RandomGenerator.getInstance().nextInt(3, 6);
for (int i = 0; i < times; i++) {
    print("hello!");
}
```

Dice exercise

RollTwoDice

- Write a console program **RollTwoDice** that repeatedly rolls two 6-sided dice until they arrive at a given desired sum.

```
Desired sum? 9
3 and 4 = 7
2 and 1 = 3
5 and 5 = 10
6 and 2 = 8
6 and 5 = 11
4 and 5 = 9
```

Dice solution

```
import acm.program.*;
import acm.util.*;

public class RollTwoDice extends ConsoleProgram {
    public void run() {
        int desiredSum = readInt("Desired sum? ");

        int die1 = 0;
        int die2 = 0;
        while (die1 + die2 != desiredSum) {
            die1 =
 RandomGenerator.getInstance().nextInt(1, 6);
            die2 =
 RandomGenerator.getInstance().nextInt(1, 6);
            println(die1 + " and " + die2 + " = " +
 (die1 + die2));
        }
    }
}
```

Random walk exercise

- Write a console program **`RandomWalk`** that randomly moves a star left or right by 1 character for a given number of steps.
  - Start the star 20 characters from the left edge.
  - Pause the program briefly after each step to produce animation.

```
Number of steps? 10
                     *
                      *
                       *
                      *
                     *
             ←        *    →
                     *
                    *
                   *
                    *
```

Random walk solution

```
import acm.program.*;
import acm.util.*;

public class RandomWalk extends ConsoleProgram {
    public void run() {
        int position = 20;
        int steps = readInt("Number of steps? ");

        for (int i = 0; i < steps; i++) {
            // randomly move left or right
            int flip = RandomGenerator.getInstance().nextInt(1, 2);
            if (flip == 1) {
                position++;
            } else {
                position--;
            }

            // draw the walker star on the screen at its position
            for (int j = 0; j < position; j++) {
                print(" ");
            }
            println("*");
            pause(50);
        }
    }
}
```