Introduction to Computer Programming

Lecture 3 – Counting Loops

Repetition with for Loops

• So far, repeating a statement is redundant:
  System.out.println("Homer says:");
  System.out.println("I am so smart");
  System.out.println("I am so smart");
  System.out.println("I am so smart");
  System.out.println("S-M-R-T... I mean S-M-A-R-T");

• Java's for loop statement performs a task many times.
  System.out.println("Homer says:");
  for (int i = 1; i <= 4; i++) {
    // repeat 4 times
    System.out.println("I am so smart");
  }
  System.out.println("S-M-R-T... I mean S-M-A-R-T");
for Loop Syntax

```java
for (initialization; test; update) {
    statement;
    statement;
    ...
    statement;
}
```

- Perform `initialization` once.
- Repeat the following:
  - Check if the `test` is true. If not, stop.
  - Execute the `statements`.
  - Perform the `update`.

### Initialization

```java
for (int i = 1; i <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tells Java what variable to use in the loop
- Performed once as the loop begins
- The variable is called a *loop counter*
  - can use any name, not just `i`
  - can start at any value, not just `1`
Test

```java
for (int i = 1; i <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tests the loop counter variable against a limit
- Uses comparison operators:
  - `<` less than
  - `<=` less than or equal to
  - `>` greater than
  - `>=` greater than or equal to

Increment And Decrement

- There are shortcuts to increase or decrease a variable's value by 1

<table>
<thead>
<tr>
<th>Shorthand version</th>
<th>Equivalent Longer</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>variable++;</code></td>
<td><code>variable = variable + 1;</code></td>
</tr>
<tr>
<td><code>--variable;</code></td>
<td><code>variable = variable -1;</code></td>
</tr>
</tbody>
</table>
Increment And Decrement - Examples

```java
int x = 2;
x++;    // x = x + 1;
       // x now stores 3

double gpa = 2.5;
gpa--;  // gpa = gpa - 1;
       // gpa now stores 1.5
```

Modify-and-Assign

- There are shortcuts to modify a variable's value.

<table>
<thead>
<tr>
<th>Shorthand version</th>
<th>Equivalent Longer</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable += value;</td>
<td>variable = variable + value;</td>
</tr>
<tr>
<td>variable -= value;</td>
<td>variable = variable - value;</td>
</tr>
<tr>
<td>variable *= value;</td>
<td>variable = variable * value;</td>
</tr>
<tr>
<td>variable /= value;</td>
<td>variable = variable / value;</td>
</tr>
<tr>
<td>variable %= value;</td>
<td>variable = variable % value;</td>
</tr>
</tbody>
</table>
Modify-and-Assign - Examples

```java
x += 3;  // x = x + 3;
gpa -= 0.5;  // gpa = gpa - 0.5;
number *= 2;  // number = number * 2;
```

Repetition Over A Range

```java
System.out.println("1 squared = " + 1 * 1);
System.out.println("2 squared = " + 2 * 2);
System.out.println("3 squared = " + 3 * 3);
System.out.println("4 squared = " + 4 * 4);
System.out.println("5 squared = " + 5 * 5);
System.out.println("6 squared = " + 6 * 6);
```

- Intuition: "I want to print a line for each number from 1 to 6"
- The for loop does exactly that!

```java
for (int i = 1; i <= 6; i++) {
    System.out.println(i + " squared = " + (i * i));
}
```

- "For each integer i from 1 through 6, print ..."
Loop walkthrough

```java
for (int i = 1; i <= 4; i++) {
    System.out.println(i + " squared = " + (i * i));
}
System.out.println("Whoo!");
```

Output:
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
Whoo!

Multi-Line Loop Body

```java
System.out.println("+-----+");
for (int i = 1; i <= 3; i++) {
    System.out.println("\" /\";
    System.out.println("/ \";
}
System.out.println("+-----+");
```

Output:

```
+-----+
| /   |
| \   |
| \   |
| \   |
+-----+
```
Expressions for counter

```java
int highTemp = 5;
for (int i = -3; i <= highTemp / 2; i++) {
    System.out.println(i * 1.8 + 32);
}

Output:
26.6
28.4
30.2
32.0
33.8
35.6
```

**System.out.print**

- Prints without moving to a new line
- It allows you to print partial messages on the same line

```java
int highestTemp = 5;
for (int i = -3; i <= highestTemp / 2; i++) {
    System.out.print((i * 1.8 + 32) + "  ");
}

Output:
26.6 28.4 30.2 32.0 33.8 35.6
```

- We can concatenate "  " to separate the numbers
Counting Down

• The update can use -- to make the loop count down.
• The test must say > instead of <

```java
System.out.print("T-minus ");
for (int i = 10; i >= 1; i--) {
    System.out.print(i + ", ");
}
System.out.println("blastoff!");
System.out.println("The end.");
```

Output:
T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, blastoff!
The end.

Generalizing HelloAgain

• We are going to write a program that allows the user to choose how many times to print the message “Hi, there!”
• Our algorithm will start as:
  1. Find out how many time to print the message.
  2. Print "Hi, there!" that many times.
Generalizing *HelloAgain* (continued)

1. Find out how many times to print the message.
2. Print "Hi, there!" that many times.

```java
System.out.println("How many times do you want to " + "say "hello"?");
totalTimes = keyb.nextInt();
```

2. Print "Hi, there!" that many times.
2. FOR Count goes from 1 TO TotalTimes
2.1 Write “Hi, there!”
Generalizing HelloAgain (continued)

System.out.println
("How many times do you want to "
 + "say \"hello\" ?");
totalTimes = keyb.nextInt();

2. FOR Count goes from 1 TO TotalTimes
2.1 Write "Hi, there!"

for (count = 0;  count < totalTimes;  count++)
    System.out.println("Hello, again");

The Revised HelloAgain

import java.util.Scanner;

public class HelloAgain3  {
    // HelloAgain3 - Write "Hello, again" as many times
    // as the user wants
    public static void main(String[] args)  {
        Scanner keyb = new Scanner(System.in);
        int i, count, totalTimes;

        System.out.println("How many times do you want to "
            + "say \"hello\"?\n");
        totalTimes = keyb.nextInt();
        for (count = 0;  count < totalTimes;  count++)
            System.out.println("Hello, again");
    }
}
Example: Averaging $n$ Numbers

- Let's get back to our original problem. We want to be able to average any number of values.
- Let's start by outlining our algorithm:
  1. Find out how many values there are.
  2. Add up all the values.
  3. Divide by the number of values
  4. Print the result

Refining Avg$n$

1. Find out how many values there are.
2. Add up all the values.
3. Divide by the number of values
4. Print the result

```java
System.out.println("How many values are you going to enter?");
numValues = keyb.nextInt();
```
Refining Avgn

System.out.println
("How many values are you going to enter?");
numValues = keyb.nextInt();

2. Add up all the values.
3. Divide by the number of values
4. Print the result

2.1 For CurrentValue goes from 1 to NumValues:
2.1.1 Get the next value
2.1.2 Add it to the total

2.0 Set the total to zero (initially there are no values)
Refining Avg

System.out.println
("How many values are you going to enter?");
numValues = keyb.nextInt();

2.0 Set the total to zero (initially there are no values)

2.1 For CurrentValue goes from 1 to NumValues:

2.1.1 Get the next value

2.1.2 Add it to the total

3. Divide by the number of values

4. Print the result

sum = 0.0;
for (currentValue = 1; currentValue <= numValues;
currentValue++)
{
    System.out.println("What is the next value?");
    value = keyb.nextInt();
    sum = sum + value;
}
Refining Avg

```java
System.out.println("How many values are you going to enter?");
numValues = keyb.nextInt();
sum = 0.0;
for (currentValue = 1; currentValue <= numValues;
    currentValue++) {
    System.out.println("What is the next value?");
    value = keyb.nextInt();
    sum = sum + value;
}
3. Divide by the number of values
4. Print the result

average = sum / numValues;
System.out.println("The average is "+ average);
```

The AverageN Program

```java
import java.util.Scanner;

public class AverageN {
    //AverageN - Find the average of N values
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        double sum, average, value;
        int numValues, currentValue;

        //Find out how many values there are
        System.out.println("How many values are you going to enter?");
        numValues = keyb.nextInt();
```
// Read in each value and add it to the sum
sum = 0.0;
for (currentValue = 1;
currentValue <= numValues;
currentValue++)
{
    System.out.println("What is the next value?");
    value = keyb.nextInt();
    sum = sum + value;
}

// Calculate and print out the average
average = sum / numValues;
System.out.println("The average is " + average);

Example: Interest Program

- Example - Write a program that calculates the interest that the Canarsie Indians would have accumulated if they had put the $24 that they had received for Manhattan Island in the bank at 5% interest.

  Input - none; all the values are fixed
  Output - Year and Principle
  Other Information -
  Principle is initially 24
  Interest = Interest Rate * Principle
  New Principle = Old Principle + Interest
Example: Interest Program

- Our initial algorithm is:
  1. Set the principle to 24
  2. For every year since 1625, add 5% interest to the principle and print out the principle.

Refining The Interest Algorithm

1. Set the principle to 24
2. For every year since 1625, add 5% interest to the principle and print out the principle.

2.1 FOR Year goes from 1625 TO Present:
   2.1.1 Add 5% interest to the principle
   2.1.2 Print the current principle
Refining The Interest Algorithm

1. Set the principle to 24

2.1 FOR Year goes from 1625 TO Present:
   2.1.1 Add 5% interest to the principle
   2.1.2 Print the current principle

   2.1.1.1 Calculate 5% Interest
   2.1.1.2 Add the interest to the principle
Refining The Interest Algorithm

```
principle = 24;
for (year = 1625; year < present; year++) {
    interest = rate * principle;
    principle = principle + interest;
}
```
Refining The Interest Algorithm

```
principle = 24;
for (year = 1625; year < present; year++) {
    interest = rate * principle;
    principle = principle + interest;
}
System.out.println("year = " + year + "\nprinciple = " + principle);
```

The Interest Program

```
public class Interest {
    // Calculate the interest that the Canarsie Indians could have accrued if they had deposited the $24 in an bank account at 5% interest.
    public static void main(String[] args) {
        final int present = 2014;
        int year;
        final double rate = 0.05;
        double interest, principle;

        // Set the initial principle at $24
        principle = 24;
```
// For every year since 1625, add 5% interest
// to the principle and print out
// the principle

for (year = 1625; year < present; year++) {
    interest = rate * principle;
    principle = principle + interest;
    System.out.println("year = " + year
               + "\tprinciple = "
               + principle);
}
**System.out.printf()**

- The method `System.out.printf()` gives us a way to write output that is formatted, i.e., we can control its appearance.

- We write the method:
  ```java
  System.out.printf(ControlString, 
  Arg1, Arg2, ... )
  ```

- The control string is a template for our output, complete with the text that will appear along with whatever values we are printing.

**System.out.printf()**: Some Simple Examples

- `System.out.printf()` will print whatever is in the control string with a few exceptions:
  ```java
  System.out.printf("This is a test");
  System.out.printf("This is a test").
  ```

  will produce:

  This is a test
  This is a test

  If you want these to be on two separate lines:
  ```java
  System.out.printf("This is a test\n");
  System.out.printf("This is a test\n").
  ```
Special Characters

- There are a number of special characters that all begin with a backslash:
  - \n new line
  - \b backspace
  - \t tab
- These can appear anywhere with a string of characters:
  System.out.printf("This is a test\nIt is!!\n");

%d and %f

- The specifiers %d and %f allow a programmer to specify how many spaces a number will occupy and (in the case of float values) how many decimal places will be used.
  - %n%d will use at least n spaces to display the integer value in decimal (base 10) format.
  - %w.d%f will use at least w spaces to display the value and will have exactly d decimal places.
### Changing the width

<table>
<thead>
<tr>
<th>Number</th>
<th>Formatting</th>
<th>Print as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>182</td>
<td>%2d</td>
<td>182</td>
</tr>
<tr>
<td>182</td>
<td>%3d</td>
<td>182</td>
</tr>
<tr>
<td>182</td>
<td>%5d</td>
<td>`182</td>
</tr>
<tr>
<td>182</td>
<td>%7d</td>
<td>`182</td>
</tr>
<tr>
<td>-182</td>
<td>%4d</td>
<td>-182</td>
</tr>
<tr>
<td>-182</td>
<td>%5d</td>
<td>`-182</td>
</tr>
<tr>
<td>-182</td>
<td>%7d</td>
<td>`'-182</td>
</tr>
</tbody>
</table>

### Changing the width (continued)

<table>
<thead>
<tr>
<th>Number</th>
<th>Formatting</th>
<th>Print as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>%1d</td>
<td>23</td>
</tr>
<tr>
<td>23</td>
<td>%2d</td>
<td>23</td>
</tr>
<tr>
<td>23</td>
<td>%6d</td>
<td>....23</td>
</tr>
<tr>
<td>23</td>
<td>%8d</td>
<td>......23</td>
</tr>
<tr>
<td>11023</td>
<td>%4d</td>
<td>11023</td>
</tr>
<tr>
<td>11023</td>
<td>%6d</td>
<td>11023.11023</td>
</tr>
<tr>
<td>-11023</td>
<td>%6d</td>
<td>-11023</td>
</tr>
<tr>
<td>-11023</td>
<td>%10d</td>
<td>......11023</td>
</tr>
</tbody>
</table>
Changing The Precision

<table>
<thead>
<tr>
<th>Number</th>
<th>Formatting</th>
<th>Prints as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.718281828</td>
<td>%8.5f</td>
<td><code>2.71828</code></td>
</tr>
<tr>
<td>2.718281828</td>
<td>%8.3f</td>
<td><code>2.718</code></td>
</tr>
<tr>
<td>2.718281828</td>
<td>%8.2f</td>
<td><code>2.72</code></td>
</tr>
<tr>
<td>2.718281828</td>
<td>%8.0f</td>
<td><code>3</code></td>
</tr>
<tr>
<td>2.718281828</td>
<td>%13.11f</td>
<td>2.71828182800</td>
</tr>
<tr>
<td>2.718281828</td>
<td>%13.12f</td>
<td>2.718281828000</td>
</tr>
</tbody>
</table>

The revised *Compound* program

```java
public class Interest {
    // Calculate the interest that the Canarsie
    // Indians could have accrued if they had
    // deposited the $24 in an bank account at
    // 5% interest.
    public static void main(String[] args) {
        final int present = 2014;
        int       year;
        final double rate = 0.05;
        double    interest, principle;

        // Set the initial principle at $24
        principle = 24;
```
// For every year since 1625, add 5% interest
// to the principle and print out
// the principle

for (year = 1625; year < present; year++) {
    interest = rate * principle;
    principle = principle + interest;

    System.out.printf
        ("year = %4d\tprinciple = $%13.2f\n", year, principle);

}