'do' and 'for' loops

October 1, 2007

ComS 207: Programming I (in Java)
Iowa State University, FALL 2007
Instructor: Alexander Stoytchev

Quick review of last lecture

Comparing Float Values

- To determine the equality of two floats, you may want to use the following technique:

```
if (Math.abs(f1 - f2) < TOLERANCE)
    System.out.println ("Essentially equal");
```

- If the difference between the two floating point values is less than the tolerance, they are considered to be equal

- The tolerance could be set to any appropriate level, such as 0.000001

Comparing Characters

- In Unicode, the digit characters (0-9) are contiguous and in order

- Likewise, the uppercase letters (A-Z) and lowercase letters (a-z) are contiguous and in order

Characters	Unicode Values
0 – 9	48 through 57
A – Z	65 through 90
a – z	97 through 122

Comparing Strings

- Remember that in Java a character string is an object

- The `equals` method can be called with strings to determine if two strings contain exactly the same characters in the same order

- The `equals` method returns a boolean result

```
if (name1.equals(name2))
    System.out.println ("Same name");
```

Comparing Strings

- We cannot use the relational operators to compare strings

- The `String` class contains a method called `compareTo` to determine if one string comes before another

- A call to `name1.compareTo(name2)`

  - returns zero if `name1` and `name2` are equal (contain the same characters)

  - returns a negative value if `name1` is less than `name2`

  - returns a positive value if `name1` is greater than `name2`

Comparing Strings

```
if (name1.compareTo(name2) < 0)
    System.out.println (name1 + "comes first");
else
    if (name1.compareTo(name2) == 0)
        System.out.println ("Same name");
    else
        System.out.println (name2 + "comes first");
```

- **Because comparing characters and strings is based on a character set, it is called a *lexicographic ordering***

Lexicographic Ordering

- **Lexicographic ordering is not strictly alphabetical when uppercase and lowercase characters are mixed**

- **For example, the string `"Great"` comes before the string `"fantastic"` because all of the uppercase letters come before all of the lowercase letters in Unicode**

- **Also, short strings come before longer strings with the same prefix (lexicographically)**
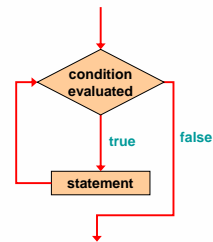
- **Therefore `"book"` comes before `"bookcase"`**
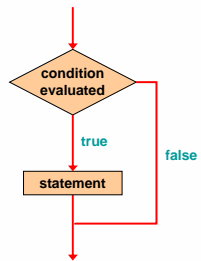
Logic of an if statement

Logic of a while Loop

The while Statement

- **A *while statement* has the following syntax:**

```
while ( condition )
    statement;
```

- **If the `condition` is true, the `statement` is executed**

- **Then the condition is evaluated again, and if it is still true, the statement is executed again**

- **The statement is executed repeatedly until the condition becomes false**

The while Statement

- **An example of a while statement:**

```
int count = 1;
while (count <= 5)
{
    System.out.println (count);
    count++;
}
```

- **If the condition of a `while` loop is false initially, the statement is never executed**

- **Therefore, the body of a `while` loop will execute zero or more times**

Other Stuff from Section 5.5

Infinite Loops

- **The body of a `while` loop eventually must make the condition false**

- **If not, it is called an *infinite loop*, which will execute until the user interrupts the program**

- **This is a common logical error**

- **You should always double check the logic of a program to ensure that your loops will terminate normally**

Infinite Loops

- **An example of an infinite loop:**

```
int count = 1;
while (count <= 25)
{
    System.out.println (count);
    count = count - 1;
}
```

- **This loop will continue executing until interrupted (Control-C) or until an underflow error occurs**

Nested Loops

- **How many times will the string `"Here"` be printed?**

```
count1 = 1;
while (count1 <= 10)
{
    count2 = 1;
    while (count2 <= 20)
    {
        System.out.println ("Here");
        count2++;
    }
    count1++;
}               10 * 20 = 200
```
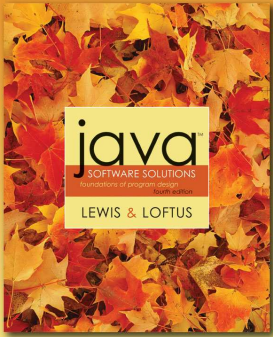
Nested Loops

- **Similar to nested `if` statements, loops can be nested as well**

- **That is, the body of a loop can contain another loop**

- **For each iteration of the outer loop, the inner loop iterates completely**

- **See PalindromeTester.java (page 235)**

Example: PalindromeTester.java (page 235)

3

Chapter 5
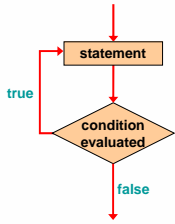
Sections 5.7 & 5.8

The do Statement

- **A *do statement* has the following syntax:**

```
do
{
    statement;
}
while ( condition )
```

- **The `statement` is executed once initially, and then the `condition` is evaluated**

- **The statement is executed repeatedly until the condition becomes false**
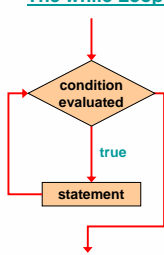
Logic of a do Loop
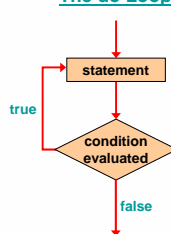
Comparing while and do

The do Statement

- **An example of a `do` loop:**

```
int count = 0;
do
{
    count++;
    System.out.println (count);
} while (count < 5);
```

- **The body of a `do` loop executes at least once**

- **See ReverseNumber.java (page 244)**

Example: ReverseNumber.java (page 244)

The for Statement

- A *for statement* has the following syntax:

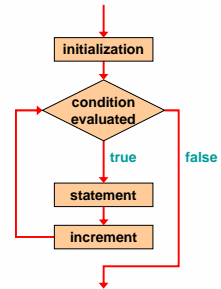The *initialization* is executed once before the loop begins

The *statement* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
    statement;
```

The *increment* portion is executed at the end of each iteration

Logic of a for loop

The for Statement

- A `for` loop is functionally equivalent to the following `while` loop structure:

```
initialization;
while ( condition )
{
    statement;
    increment;
}
```

The for Statement

- An example of a `for` loop:

```
for (int count=1; count <= 5; count++)
    System.out.println (count);
```

- The initialization section can be used to declare a variable

- Like a `while` loop, the condition of a `for` loop is tested prior to executing the loop body

- Therefore, the body of a `for` loop will execute zero or more times

The for Statement

- The increment section can perform any calculation

```
for (int num=100; num > 0; num -= 5)
    System.out.println (num);
```

- A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance

- See **Multiples.java** (page 248)

- See **Stars.java** (page 250)

Example: Multiples.java (page 248)

5

Example: Stars.java (page 250)

The for Statement

- **Each expression in the header of a `for` loop is optional**

- **If the initialization is left out, no initialization is performed**

- **If the condition is left out, it is always considered to be true, and therefore creates an infinite loop**

- **If the increment is left out, no increment operation is performed**

THE END