09. Looping Statements

Java

Summer 2008

Instructor: Dr. Masoud Yaghini

Outline

- The while Statement
- The do-while Statement
- The for Statement
- References

The while Statement

The while Statement

- The while statement continually executes a block of statements while a particular condition is True.
- The while statement has this general form:

```
while (expression) {
        statement (s)
}
```

- The while statement evaluates expression, which must return a boolean value.
- If the expression evaluates to true, the while statement executes the statement(s) in the while block.

The while Statement

- Using the while statement to print the values from 1 through 10:

```
class WhileDemo {
    public static void main(String[] args) {
        int count = 1;
        while (count < 11) {
            System.out.println("Count is: " + count);
            count++;
        }
    }
}
```

The while Statement

- You can implement an infinite loop using the while statement as follows:

```
while (true)  {
        // your code goes here
}
```

The do-while Statement

The do-while Statements

- The do-while statement can be expressed as follows:

```
do  {
    statement (s)
} while (expression);
```

- The difference between do-while and while is that do-while evaluates its expression at the bottom of the loop instead of the top.
- Therefore, the statements within the do block are always executed at least once.

The do-while Statements

```
class DoWhileDemo {
    public static void main(String[] args) {
        int count = 1;
        do {
            System.out.println("Count is: " + count);
            count++;
        } while (count < 11);
    }
}
```

The for Statement

The for Statement

- The for statement provides a compact way to iterate over a range of values.
- Programmers often refer to it as the "for loop"
- The general form of the for statement can be expressed as follows:

```
for (initialization; termination; increment) {
    statement(s)
}
```

The for Statement

- When using this version of the for statement:
  - The *initialization* expression initializes the loop; it's executed once, as the loop begins.
  - When the *termination* expression evaluates to false, the loop terminates.
  - The *increment* expression is invoked after each iteration through the loop; it is perfectly acceptable for this expression to increment or decrement a value.

The for Statement

- The following program uses the general form of the for statement to print the numbers 1 through 10:

```
class ForDemo {
  public static void main(String[] args){
    for(int i=1; i<11; i++){
      System.out.println("Count is: " + i);
    }
  }
}
```

Initialization

- Notice how the code declares a variable within the initialization expression.
- The scope of this variable extends from its declaration to the end of the block governed by the for statement.
- If the variable that controls a for statement is not needed outside of the loop, it's best to declare the variable in the initialization expression.
- The names i, j, and k are often used to control for loops

The for Statement

- The three expressions of the for loop are optional; an infinite loop can be created as follows:

```
for ( ; ; ) {   // infinite loop
    // your code goes here
}
```

References

References

- S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, **The Java Tutorial: A Short Course on the Basics**, 4th Edition, Prentice Hall, 2006. (Chapter 3)

The End