Conditions and Loops

CS 3: Computer Programming in Java
Objectives

- Learn how to use decision (if and switch) and looping mechanisms (while, do-while, and for)
- Move on to use built-in classes that incorporate iteration
Decision Statements

- if/else
- switch
if/else

- The simplest decision tool in Java
- As in other languages, a test condition is evaluated and, if the result evaluates to the Boolean value true, the associated one or more statements enclosed in curly braces following the condition are executed
- The if..else in Java is superficially identical to C/C++
It is formed from one or more statements with a form:

```java
if ( test_condition_is_true ) {
    one or more executable statements;
}
else if ( test_condition_is_true ) {
    one or more executable statements;
}
else {
    one or more executable statements;
}
```

// First condition
// Second evaluated if first is false
// Default if first and second are false
Unlike C/C++, the test_condition must evaluate to type Boolean
  - That is, it must evaluate to a result of true or false
- It can be any valid expression composed of variables and logical operators, including method calls to methods returning Boolean values
The switch statement is a multi-value test for an int or char variable. It compares the value of the variable to the values of each case constant. If a matching case is found, the statements within the case are executed up to encountering a break statement. However, break is optional for each case, so that flow can proceed from one case to the next if no break statement is found. The default case is only executed if no matches for the case constants are found.
The form of the switch statement is as follows:

```
switch(variable)
{
    case constant1:  statements;
        break; (optional)
    case constant2:  statements;
        break;
    etc.

    default:         statements; (optional)
}
```
Looping Statements

- Types
  - while()
  - do/while()
  - for()

- All depend on Boolean conditions for control

- The while() and do..while() are best used where the terminating conditions are going to occur at unknown intervals

- The for() construct is used where the exact number of execution steps is known in advance
Looping Statements (2)

- They all depend on a `test_condition_is_true` which is enclosed in parentheses for control.
- The `test_condition` is an expression which evaluates as a Boolean true/false and, as long as it is true, execution of the statements in the control block (delineated by curly braces) continues.
While

- Like the if..else mechanism, the basic syntax of the while statement is also superficially identical to C/C++
- That is:
  ```
  while ( test_condition_is_true ) { one or more executable statements; }
  ```
- Once again, unlike C/C++, the expression which evaluates to the test_condition_is_true must be of type Boolean
  - That is, it must evaluate to a result value of true or false
This looping/decision mechanism performs in the same manner as the while() with the difference being that the associated statements are always executed at least once prior to the condition being tested.

In the following code:

```cpp
    do { one or more executable statements; } while (test_condition is true);
```

The one or more executable statements are processed first, then the test_condition is evaluated (to true or false) and the statements processed again only if the condition is true.
for

- Syntax of a for loop should be very familiar to C++ programmers and is also nearly identical to the C for loop.
- It varies from the C syntax in that the control variable can be both declared and initialized inside the for parentheses.
- The basic construction is:
  ```
  for ( initial condition ; test_condition_is_true ; terminating condition ) {
    one or more executable statements;
  }
  ```
The initial condition can include one or more variables, including their type declaration and value initialization as in:

```c
for ( int i = 0, j = 5 ; i < 12 && j > 0; i += 2, j-- )
```

The scope of the variables i and j is limited to the loop in which they are declared.
Iterators

- An Iterator is an object that has methods that allow you to process a collection of items at one time
- It lets you step through each item in the collection and interact with it as needed
- It is defined using the Java Iterator interface which provides methods like hasNext() which can be used as a condition of a loop to control the processing of each item
- The Scanner class implements the Iterator interface
Iterators Example

- The following example shows how the Scanner class object could be used to read in a list of URLs and break them up into their path components using the “/” as the delimiter (token separator).

- The input file named url.inp, containing URLs, is as follows:
  - www.google.com
  - java.sun.com/j2se/1.5
  - www.linux.org/info/gnu.html
  - duke.csc.villanova.edu/lewis/
  - www.csc.villanova.edu/academics/index.jsp
Iterators Example (2)

- The following Java application defines a Scanner object to read the input file lines into a String object.
- Then it defines another Scanner object to process the text in the String object using Iterator methods to separate the tokens of each URL using “/” as the delimiter.
import java.util.Scanner;
import java.io.*;

public class URLDissector
{
    // Reads urls from a file and prints their path components.
    public static void main (String[] args) throws IOException
    {
        String url;
        Scanner fileScan, urlScan;

        fileScan = new Scanner (new File("c:\student\urls.inp"));

        // Read and process each line of the file
        while (fileScan.hasNext())
        {
            url = fileScan.nextLine();
            System.out.println ("URL: " + url);

            urlScan = new Scanner (url);
            urlScan.useDelimiter("/");

            // Print each part of the url
            while (urlScan.hasNext())
            {
                System.out.println ("   " + urlScan.next());
            }
        }
    }
}
The following is the output from the run of the program above:
run:
URL: www.google.com
	www.google.com
URL: java.sun.com/j2se/1.5
	java.sun.com
	j2se
	1.5
URL: www.linux.org/info/gnu.html
	www.linux.org
	info
	nu.html
URL: duke.csc.villanova.edu/lewis/
	duke.csc.villanova.edu
	lewis
URL: www.csc.villanova.edu/academics/index.jsp
	www.csc.villanova.edu
	academics
	index.jsp
Summary

- Decision statements
  - if/else
  - switch
- Looping statements
  - while
  - do/while
  - for
- Iterators
  - Object that has methods that allow you to process a collection of items at one time
  - Lets you step through each item in the collection and interact with it as needed