Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-two)

2.5: Control Structures

for

Java derives its set of control structures from C. If you're familiar with that language, there will be few surprises for you here. Otherwise, I think you'll find that Java's control structures are easy to use and understand. I'm only going to point out a few ways in which these control structures differ from those of C and from Visual Basic.

We saw a use of the for loop already.

```
int i, j=0;
for (i=0; i<10; i++)
   j += i;
```

One nice feature that Java shares with C++ is that you can declare the loop-counter variable (which is used only inside the loop) in the for statement itself:

```
int j=0;
for (int i=0; i<10; i++)
   j += i;
```

In Java, such a loop counter lives only in the scope of the for statement itself. The following code is illegal in Java:

```
int j=0;
for (int i=0; i<10; i++)
   j += i;
System.out.println("i is now" + i);
```

Compiler says:

```
test.java:8: Undefined variable: i
   System.out.println("i is now" + i);
                       ^
   1 error
```

The equivalent code in Visual Basic or C++ would be perfectly valid.

switch

The switch statement in C and in Java:

```
//get value from somewhere
  int j = interestingFunction();
```

```
    switch(j)
    {
    case 0:
      doSomething(); break;

    case 1:
       doSomethingElse(); break;

    case 2:
      doSomethingAltogetherDifferent();
      break;

    default:
      System.out.println("Unexpected j value!");
      break;
    }
```

is very similar to the SELECT CASE statement in Visual Basic, with one important difference: in Java and C, the individual cases can "fall through," so that more than one can be executed during a singleswitch statement. That is the purpose of the break statement. If the break statements were removed, in the case of j being equal to 1, both doSomethingElse() and doSomethingAltogetherDifferent() would be executed, and then the error message would be printed! Forgetting the break statements is a major source of errors in the use of switch, so be careful!

goto

The folks who designed the Java language had a healthy, if dry, sense of humor. Case in point: "goto" is a reserved word in Java, just as it is in C, C++, and Visual Basic. In those languages, it is much reviled as the cause of "spaghetti code." (Anti-goto sentiment, in fact, gave rise to one of the biggest revolutions in programming history, the rise of "structured programming.") In Java, "goto" is reserved, which means that it is not available for use as a variable name, for example; but it is not implemented as a language keyword, either, so effectively the evil "goto" is banished from appearing anywhere in Java code, entirely out of spite! The compiler even knows about it:

```
test.java:14: "goto" not supported.
    goto foo;
    ^
1 error
```

labeled and break continue

Real programmers know that once in a while the evil "goto" becomes a necessary evil. What do you do in such a case? For example:

```
 for (int i=0; i<10; i++)
    for (int j=0; j<10; j++)
       for (int k=0; k<10; k++)
```

```
         for (int m=0; m<10; m++)
            {
            if (userCancelled())
               // Oh-oh - what do I do?
            else
               // do i,j,k,m processing
                    }
```

Here a goto would be justified; anything else would make the code harder to understand. In this case, Java supports the notion of labeled breaks:

```
do_ijkm:
   for (int i=0; i<10; i++)
      for (int j=0; j<10; j++)
         for (int k=0; k<10; k++)
            for (int m=0; m<10; m++)
               {
               if (userCancelled())
                  break do_ijkm;
               else
                  // do i,j,k,m processing
               }
```

A break statement like this will halt all the nested loops in progress and exit out the bottom of the labeled one. Similarly, continue statements, which abort the current iteration of a loop and begin the next one, can include a label, so that outer nested loops can be restarted as well.