Chapter 4 : Controlling Execution

➢ **Iteration :**

Example :

```
//: control/WhileTest.java
// Demonstrates the while loop.
public class WhileTest {
      static boolean condition()
      {
      boolean result = Math.random() < 0.8;
      System.out.print(result + ", ");
      return result;
      }
      public static void main(String[] args)
      {
      while(condition())
      System.out.println("Inside ‘while’");
      System.out.println("Exited ‘while’");
      }
}
```

Above math.random() generate a number between(0 to 1) . The Boolean value get stored in result which is used as a condition in while. ‘Indisde while’ will be executed till the result is found true otherwise output will be ‘Exited while’.

➢ **For :**

Syntax for ‘for’ loop is given as

```
for(initialization; Boolean-expression; step)
         statement
```

Above we can put more than one statement separated by comma in place of initialization and step. Example for the same is show below.

```
//: control/CommaOperator.java
public class CommaOperator {
      public static void main(String[] args)
      {
            for(int i = 1, j = i + 10; i < 5; i++, j = i * 2)
            {
            System.out.println("i = " + i + " j = " + j);
            }
      }
}
Output:
i = 1 j = 11
i = 2 j = 4
i = 3 j = 6
i = 4 j = 8
```

➢ **Foreach Syntax** :

It is useful way to use 'for'i.e. by using foreach one can easily iterate through the arrays. Example is shown below.

```
//: control/ForEachFloat.java
import java.util.*;
public class ForEachFloat {
public static void main(String[] args) {
Random rand = new Random(47);
float f[] = new float[10];
for(int i = 0; i < 10; i++)
f[i] = rand.nextFloat();
for(float x : f)//remember foreach never ends with semicolon
System.out.println(x);
}
} /* Output:
0.72711575
0.39982635
0.5309454
0.0534122
0.16020656
0.57799757
0.18847865
0.4170137
0.51660204
0.73734957
```

Foreach can be used for a method that returns a array. As in string the method toCharArray() returns an array of characters so that one can easily iterate through characters in the string.

Example:

```
//: control/ForEachString.java
public class ForEachString {
      public static void main(String[] args) {
      for(char c : "An African Swallow".toCharArray() )
      System.out.print(c + " ");
      }
}
Output:
A n A f r i c a n S w a l l o w
```

It should be also remember that foreach never **ends** with a semicolon(;).

➢ There are some keywords which provide *unconditional branching* i.e. branch happens without any test. These include **return, break** and **continue.** Break quits the loop without executing the remaining statements and continue quits the current iteration and goes to the beginning of the loop to execute the next iteration.

➢ Goto cant be used in java. But label can be used with break and continue. As shown in the following program

```
//: control/LabeledFor.java
// For loops with "labeled break" and "labeled continue."
import static net.mindview.util.Print.*;
public class LabeledFor {
      public static void main(String[] args) {
      int i = 0;
      outer: // Can't have statements here
      for(; true ;) { // infinite loop
            inner: // Can't have statements here
            for(; i < 10; i++) {
            print("i = " + i);
            if(i == 2) {
            print("continue");
            continue;
      }
      if(i == 3) {
      print("break");
      i++; // Otherwise i never
      // gets incremented.
      break;
      }
      if(i == 7) {
      print("continue outer");
      i++; // Otherwise i never
      // gets incremented.
      Continue outer;
      }
      if(i == 8) {
      print("break outer");
      break outer;
      }
      for(int k = 0; k < 5; k++) {
      if(k == 3) {
      print("continue inner");
      continue inner;
      }
      }
      }
      }
      // Can't break or continue to labels here
      } } /* Output:
      i = 0
      continue inner
      i = 1
      continue inner
      i = 2
      continue
      i = 3
      break
      i = 4
      continue inner
      i = 5
      continue inner
      i = 6
      continue inner
      i = 7
      continue outer
      i = 8
      break outer
```