3. Java Language Structure

Data types

Operators

Control Statements

Control Statements

- Control statements cause the flow of execution to advance and branch based on changes to the state of a program.
- Java control statements are classified into:
  - Selection Statements
  - Iteration Statements
  - Jump Statements

Selection Statements

- Java supports two selection statements:
  - ***if*** statement
  - ***switch*** statement
- These statements allow us to control the flow of our program's execution based upon conditions known only during run time.

The *if* statement

```
if (condition) {
        statement1;
} else
        statement2;
```

Example:

```
if (response == OK) {
        // code to perform OK action
} else {
        // code to perform Cancel action
}
```

The *if* statement

- The nested *if* s

```
if (i=10) {
        if (j<20)
                a = b;
        if (k > 100)
                c = d;
        else
                a = c;
}
else
        a = d;
```

The *if-else-if* Ladder:

```
public class IfElseDemo  {
    public static void main(String[] args)  {
            int testscore = 76;  char grade;
            if (testscore >= 80)
                    grade = 'A';
            else if (testscore >= 70)
                    grade = 'B';
            else if (testscore >= 60)
                    grade = 'C';
            else
                    grade = 'F';
            System.out.println("Grade = " + grade);
    }
}
```

The switch Statement

```
switch(expression) {
    case value1:
        // statement sequence
        break;
    case value2:
        // statement sequence
        break;
    ……
    default:
        // default statement sequence
}
```

```
public class SwitchTest {
    public static void main(String args[]) {
            for (int i=0; i<5; i++)
                    switch (i)  {
                            case 0:
                                    System.out.println("i is ZERO.");
                                    break;
                            case 1:
                                    System.out.println("i is ONE.");
                                    break;
                            case 2:
                                    System.out.println("i is TWO.");
                                    break;
                            default:
                                    System.out.println("i is > TWO.");
                    }
    }
}
```

Output

```
i is ZERO.
i is ONE.
i is TWO.
i is >  TWO.
i is >  TWO.
```

What happens if all the '*break*' s are removed?

Iteration Statements

- Iteration statements are also called looping statements. They are:
  - ***for*** loop
  - ***while*** loop
  - ***do-while*** loop
- They repeatedly execute the same set of instructions until a termination condition is met.

The *for* Statement

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

Example:

```
for (int i=0; i<5; i++) {
    . . . .
}
```

```
public class ForDemo {

   public static void main(String[] args) {

        int[] intArray = { 32, 87, 3, 589, 12,107, 200, 8, 62 };

        for (int i = 0; i < intArray.length; i++) {
                System.out.print(intArray[i] + ", ");
        }

        System.out.println();
   }
}
```

Output:

```
32, 87, 3, 589, 12, 107, 200, 8, 62,
```

Equivalent forms of ***for***

```
for (int i=0; i<5; i++)  { . . . }
    . . . .
}
```

```
int i = 0;
for ( ; i<5; i++)  {. . . }
```

```
 int i = 0;
for ( ; i<5; )  {
    . . .
    i++;
}
```

```
for( ; ; ) {  // infinite loop
    . . .
}
```

What is the output of this code snippet?

```
int count = 0;
for (int i = 0; i < 7; i++);
     count++;
System.out.print(“Count Value is: “ + count);
```

The *while* Statement

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

The expression evaluates to boolean.

```
public class WhileDemo  {
   public static void main(String[ ] args)  {
        String copyFromMe = "Copy this string until 'g'.";
        StringBuffer copyToMe = new StringBuffer();
        int i = 0;
        char c = copyFromMe.charAt(i);

        while (c != 'g')  {
                copyToMe.append(c);
                c = copyFromMe.charAt(++i);
        }
        System.out.println(copyToMe);
   }
}
```

The *do-while* Statement

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

Note: Always executes its body at least once, even if the conditional expression is false to begin with.

```
public class DoWhileDemo  {
   public static void main(String[ ] args)  {
        String copyFromMe = "Copy this string until 'g'.";
        StringBuffer copyToMe = new StringBuffer();
        int i = 0;
        char c = copyFromMe.charAt(i);
         do  {
                copyToMe.append(c);
                c = copyFromMe.charAt(++i);
        } while (c != 'g');
        System.out.println(copyToMe);
   }
}
```

Branching Statements

- The Java programming language supports three branching statements:
  - The ***break*** statement
  - The ***continue*** statement
  - The ***return*** statement

The ***break*** statement

- Has three uses:
  - To terminate a statement sequence in a switch statement.
  - To exit a loop
  - As a civilized form of '***goto***'– *break label*;

```
public class BreakDemo  {
    public static void main(String[] args)  {
        int[] intArray = { 32, 87, 3, 589, 12, 1076, 8, 622, 127 };
        int searchFor = 12;
        boolean foundIt = false;

        for (int i=0; i < intArray.length; i++)  {
                if (intArray[i] == searchFor)  {
                        foundIt = true;
                        break;
                }
        }
        if (foundIt) {
                System.out.println("Found " + searchFor + " at index " + i);
        }
        else {
                System.out.println(searchFor + "not in the array");
        }
    }
}
```

The ***continue*** statement

- Used when we want to continue running the loop, but stop processing the remainder of the code in its body for this particular iteration.

```
public class ContinueDemo  {

    public static void main(String[] args)  {

        String searchMe = new String (
                          "peter piper picked a peck of pickled peppers");
        int max = searchMe.length();
        int numPs = 0;

        for (int i = 0; i < max; i++)  {
                if (searchMe.charAt(i) != 'p')  // interested only in 'P's
                        continue;
                numPs++;  // increment the 'P' count.
        }

        System.out.println("Found " + numPs + " p's in the string");
        System.out.println(searchMe);
    }
}
```

The ***return*** statement

- Used to explicitly return from a method.
- It causes program control to transfer back to the caller of the method.

```
public class ReturnDemo  {
  public static void main(String[] args)  {
      boolean t = true;
      System.out.println(" Before the return.");
      if(t) return;  //  return to caller.
      System.out.println("This won't execute");
  }
}
```