

2.1 Working with Control Statements in Java

What we need to understand at this point in time is that Java or any other programming language deals with 3 basic constructs or approaches – Sequence, Selection and Iteration.

Sequence: A program that follows a sequence is basically one that does not do anything except execute each line of the program code one after another. There is no condition check and no branching of control in the program.

Selection: A program that executes only one set of instructions, based on a condition check producing a true or a false value.

Iteration: No one wants to repeat code when a looping approach can be used. You want to print your 5 times on the screen then you use an iterative construct which executes **1** printout command 5 times.

Whenever the computer runs a Java program, it goes straight from the first line of code to the last. Control statements allow you to change the computer's control from automatically reading the next line of code to reading a different one.

Let's assume you only want to run some code on condition. For example, let's say that you are making a program for a bank. If someone wants to see his records, and gives his password, you don't always want to let him see the records. First, you need to see if his password is correct. You then create a control statement saying "if" the password is correct, and then run the code to let him see the records. "else" run the code for people who enter the wrong password. You can even put one control statement inside another. In our example, the banking system has to worry about people trying to get someone else's records. So, as the programmer, you give the user three shots. If he misses them, then the records are locked for a day. You can have code with control statements that look like this (not real code, only meant to help you understand. also, Java doesn't have go back to earlier lines- again just trying to focus on the control part.)

Using an IF-Statement

```
class IfElseDemo {
    public static void main(String[] args) {

        int testscore = 76;
        char grade;

        if (testscore >= 90) { //comparison of values starts here
            grade = 'A';
        } else if (testscore >= 80) {
            //executed only if previous "if" was false

            grade = 'B';
        }
    }
}
```

Program Construction in Java

Class notes - Session 2

```
        } else if (testscore >= 70) {
            grade = 'C';
        } else if (testscore >= 60) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        System.out.println("Grade = " + grade);
    }
}
```

2.2 Understanding the different kinds

The basic selection control statements that Java uses are the following:

- If
- If – Else
- If – Else – If (Nested If, which means one if condition within another.)
- Switch- Case

IF / IF-ELSE / IF-ELSE-IF statements

The **if** statement enables your program to selectively execute other statements, based on some criteria. For example, suppose that your program prints debugging information, based on the value of an integer variable named `NUM`. If `NUM` is 0, your program prints debugging information, such as the value of a variable, such as `x`. Otherwise, your program proceeds normally

```
if (NUM == 0) {
    System.out.println("DEBUG: x = " + x);
}
```

The `else` block is executed if the `if` part is false. Another form of the `else` statement, `else if`, executes a statement based on another expression. An `if` statement can have any number of companion `else if` statements but only one `else`. Following is a program `IfElseDemo` that assigns a grade based on the value of a test score: an A for a score of 90% or above, a B for a score of 80% or above, and so on:

```
public class IfElseDemo {
    public static void main(String[] args) {

        int testscore = 76;
        char grade;

        if (testscore >= 90) {
            grade = 'A';
        } else if (testscore >= 80) {
            grade = 'B';
        } else if (testscore >= 70) {
```

Program Construction in Java

Class notes - Session 2

```
        grade = 'C';
    } else if (testscore >= 60) {
        grade = 'D';
    } else {
        grade = 'F';
    }
    System.out.println("Grade = " + grade);
}
}
```

The output from this program is:

Grade = C

SWITCH-CASE

The **switch** statement to conditionally perform statements based on an integer expression. Following is a sample program, `SwitchDemo` that declares an integer named `month` whose value supposedly represents the month in a date. The program displays the name of the month, based on the value of `month`, using the `switch` statement:

```
public class SwitchDemo {
    public static void main(String[] args) {

        int month = 8;
        switch (month) {
            case 1: System.out.println("January"); break;
            case 2: System.out.println("February"); break;
            case 3: System.out.println("March"); break;
            case 4: System.out.println("April"); break;
            case 5: System.out.println("May"); break;
            case 6: System.out.println("June"); break;
            case 7: System.out.println("July"); break;
            case 8: System.out.println("August"); break;
            case 9: System.out.println("September"); break;
            case 10: System.out.println("October"); break;
            case 11: System.out.println("November"); break;
            case 12: System.out.println("December"); break;
        }
    }
}
```

The `switch` statement evaluates its expression, in this case the value of `month`, and executes the appropriate *case* statement. Thus, the output of the program is: August. Of course, you could implement this by using an `if` statement:

```
int month = 8;
if (month == 1) {
    System.out.println("January");
} else if (month == 2) {
    System.out.println("February");
}
. . . // and so on
```