Department of Software Engineering
Mehran University of Engineering and Technology, Jamshoro

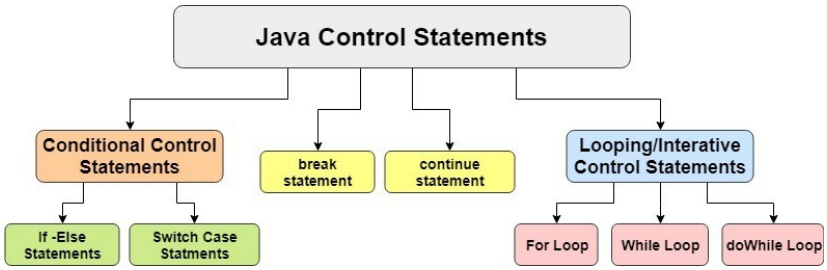
Course: SWE224 – Data Structure & Algorithms			
Instructor	Mariam Memon	**Practical/Lab No.**	01
Date		**CLOs**	3
Signature		**Assessment Score**	01

Topic	**To become familiar with control structures in Java**
Objectives	To implement interactive conditional & iterative structures- if, if-else, switch, for, for each,while and while..do loop

Lab Discussion: Theoretical concepts and Procedural steps

Control Structures

Execution of a program is linear. Control Structures are used to alter/change the flow of program.



Conditional Statements

Following conditional statements are supported by Java

1. if statement
2. nested if statement
3. if-else statement
4. if-else-if statement
5. Switch Case Statement

if statement:

The if statement is the most basic of all the control flow statements. The if statement tells our program to execute a certain section of code only if a particular test evaluates to true.

Syntax:

```
if(condition){
  Statement(s);
}
```

Nested if statement:

An if statement inside another the statement. If the outer if condition is true then the section of code under outer if condition would execute and it goes to the inner if condition. If inner if condition is true then the section of code under inner if condition would execute.

Syntax:

```
if(condition_1) {
   Statement1(s);

   if(condition_2) {
      Statement2(s);
   }
}
```

if-else statement:

If a condition is true then the section of code under if would execute else the section of code under else would execute.

Syntax:

```
if(condition) {
   Statement(s);
}
else {
   Statement(s);
}
```

if-else-if statement:

```
if(condition_1) {
   /*if condition_1 is true execute this*/
   statement(s);
}
else if(condition_2) {
   /* execute this if condition_1 is not met and
    * condition_2 is met
    */
   statement(s);
}
else if(condition_3) {
   /* execute this if condition_1 & condition_2 are
    * not met and condition_3 is met
    */
   statement(s);
}
.
.
.
else {
   /* if none of the condition is true
    * then these statements gets executed
    */
   statement(s);
}
```

Switch Case:

The switch statement in Java is a multi branch statement. We use this in Java when we have multiple options to select. It executes particular option based on the value of an expression.

Switch works with the byte, short, char, and int primitive data types. It also works with enumerated types, the String class, and a few special classes that wrap certain primitive types such as Character, Byte, Short, and Integer.

Syntax:

```
switch(expression) {
    case valueOne:
        //statement(s
        break;
    case valueTwo:
        //statement(s
        break;
:
:
:
    default: //optional
    //statement(s) //This code will be executed if all cases are not matc
}
```

Iterative Statements

Iteration statements cause statements (or compound statements) to be executed zero or more times, subject to some loop-termination criteria. When these statements are compound statements, they are executed in order, except when either the break statement or the continue statement is encountered.

Loops are basically means to do a task multiple times, without actually coding all statements over and over again. For example, loops can be used for displaying a string many times, for counting numbers and of course for displaying menus. Loops in Java are mainly of three types :-

1.'while'loop
2. 'do while' loop
3. 'for' loop

The 'while' loop

```
class WhileLoop
 {
    public static void main(String args[])
     {
          int i=1;
          while(i<=3)
               {
                    System.out.println(i);
                    i++;
               }
        }
 }
```

The 'do while' loop

It is very similar to the 'while' loop shown above. The only difference being, in a 'while' loop, the condition is checked beforehand, but in a 'do while' loop, the condition is checked after one execution of the loop.

Example code for the same problem using a 'do while' loop would be :-

```
class
DoWhileLoop
  {
     public static void main(String args[])
       {
            int i=1;
            do
              {
                  System.out.println(i);
                  i++;
               }while(i<=3);
       }
}
```

The 'for' loop

This is probably the most useful and the most used loop in Java The syntax is slightly more complicated than that of 'while' or the 'do while' loop.

The general syntax can be defined as :-

for(<initial value>;<condition>;<increment>)

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

          for(int i=1;i<=3;i++)
              {
                    System.out.println(i);
                }
        }
  }
```

Lab Tasks

Task 1: Develop a java program for calculating the total electricity bill when a user enters the number of units consumed. For the calculations of the total bill consider the following tariff:

Number of Units	**Price PER unit in Rupees**
Initial 50 units	10
50-100	15
101-200	20
201-300	25
301-onwards	30

Task 2: Develop java code that prints following.

```
    *
   ***
  *****
 *******
*********
```

```
      1
    2 3 2
  3 4 5 4 3
4 5 6 7 6 5 4
5 6 7 8 9 8 7 6 5
```

```
*****
****
***
**
*
```

Task 3: Write a java program that takes the table, starting and ending point of the table and prints the output in the following way:

5x5 = 25
5x6 = 30
5x7 = 35
5x8 = 40
5x9 = 45
5x10 = 50

Task 4: Write a program to display first n terms of a Fibonacci series.

Sample Output:

Input number of terms to display : 10
Fibonacci Series: 1 1 2 3 5 8 13 21 34

Task 5: Write a program that calculates the square of a number provided by the user without using any built-in function or the * operator.

Lab Rubrics for Evaluation

Rubrics	Proficient	Adequate	Poor
Programming Logic	Produce correct results with good use of control structures **(0.4)**	Correct use of control structures but produces partially correct results **(0.2)**	Unable to apply control structures **(0.0)**
Originality	Submitted work shows large amount of original thought **(0.3)**	Submitted work shows some amount of original thought **(0.2)**	Submitted work shows no original thought **(0.0)**
Troubleshooting	Easily traced and corrected faults **(0.3)**	Traced and corrected faults with some guidance **(0.2)**	No troubleshooting skills **(0.0)**