Control Structures

Control Statements provide us flexibility to control the flow of execution

Slides include:-

Logical operators applications

Precedence order

Nested if’s

Switch-Case

For loop

Objectives and Learning Outcoms

- Learn Logical operator applications
- Learn useful selection statement
- Learn interactive program implementation and nested conditions
- Advantages of multiway branching
- Iterations and its implementation

Logical Operators

- Conditions in selection statements and loops can use logical operators to form complex expressions

```
if (b >= a && a >= c)
    max = b;
if (a >= b && b >= c)
    max = a;
if (c>=a  && a>= b )
   max = c;
```

- Logical operators have precedence relationship between themselves and other operators

Operator Precedence

Highest

NOT!	!
multiplicative	* / %
additive	+ -
relational	< > <= >=
equality	== !=
conditional AND	&&
conditional OR	||
assignment	= += -= *= /= %=

Lowest

Nested If Statement

- The if-true-statement and if-false-statement of an if statement could be another if statement
- These are called nested if statements

```
if (a >= b)
  if (b >= c) min = c;
  else min = b;
else
  if (a >= c) min = c;
  else min = a;
```

- An else clause is matched to nearest if (no matter what the indentation implies)

```
import java.util.Scanner;
class data
{
public static void main(String []a)
{
Scanner s=new Scanner(System.in);
  double p=s.nextDouble();
if (p >= 90) System.out.println("You got an A");
else if ( >= 80) System.out.println("You got a B");
else if(score>=60) System.out.println("You got a C");
else if(score>=40) System.out.println("You got a D");
else
  System.out.println("You got F");
}}
```

The Switch Statement

- The switch statement provides another means to decide which statement to execute next

- The switch statement evaluates an expression, then attempts to match the result to one of several possible cases

- Each case contains a value and a list of statements

- The flow of control transfers to statement list associated with the first value that matches

The Switch-case Statement

- A switch statement can have an optional default case which has no associated value
- If the default case is present, control will transfer to it if no other case value matches
- The default case can be positioned anywhere in the switch, it is usually placed at the end
- If there is no default case, and no other value matches, control falls through to the next statement after the switch

The Switch Statement

- Often a break statement is used as the last statement in each case’s statement list
- A break statement causes control to transfer to the end of the switch statement
- If a break statement is not used, the flow of control will continue into the next case
- The expression of a switch statement must result in an integral data type, like an integer or character
- You cannot perform relational checks with a switch staement

```
switch(num) {
  case 0:
  case 1:
  case 2:
  case 3:
  case 4: System.out.println("F");
       break;
  case 5: System.out.println("D");
       break;
  case 6:
  case 7: System.out.println("C");
       break;
  case 8: System.out.println("B");
       break;
  case 9:
  case10: System.out.println("A");
```

Iterative Stmt- For Statement

- The for statement has the following syntax:

```
for ( initialization ; condition ; increment )
 {          statement1 ;
             staetment2;
}
```

- The initialization is executed once before the loop begins

- The statements are executed until the condition becomes false
- The increment portion is executed at the end of each iteration