Decision Making in Java

(if, if-else, switch, break, continue, jump)

Decision Making in programming is similar to decision-making in real life. In programming also face some situations where we want a certain block of code to be executed when some condition is fulfilled.

A programming language uses control statements to control the flow of execution of a program based on certain conditions. These are used to cause the flow of execution to advance and branch based on changes to the state of a program.

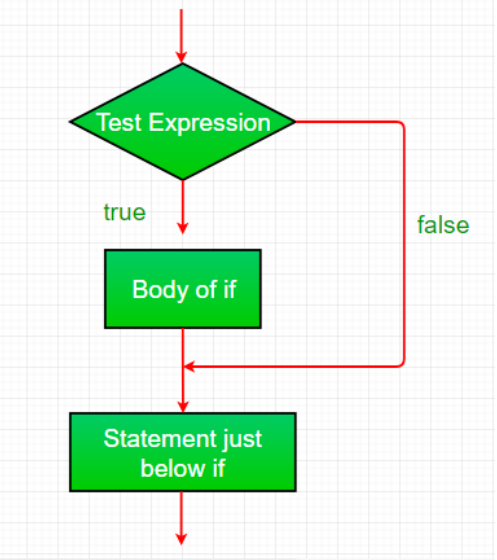
Java's Selection statements:

- **if**
- **if-else**
- **nested-if**
- **if-else-if**
- **switch-case**
- **jump** – break, continue, return

---

1. if

```
if(condition)
{
   // Statements to execute if
   // condition is true
}
```

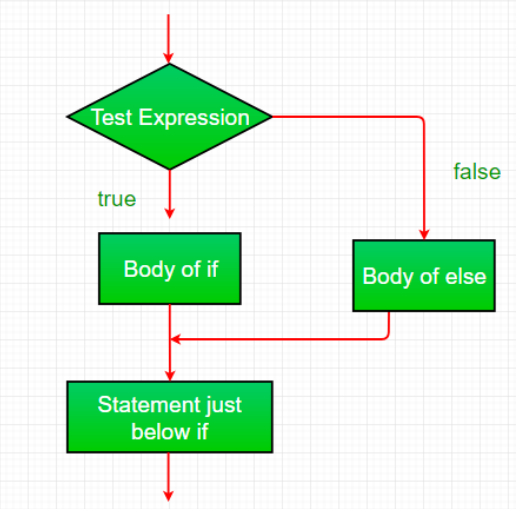


Time Complexity: O(1)

Auxiliary Space : O(1)

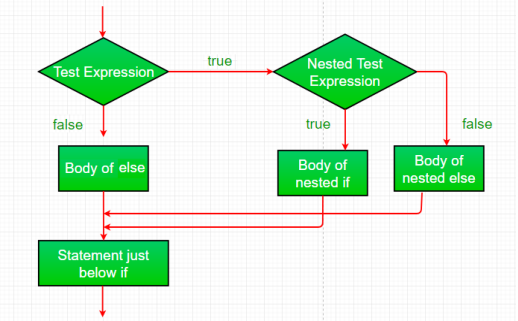
2. if-else

```
if (condition)
{
    // Executes this block i
    // condition is true
}
else
{
    // Executes this block i
    // condition is false
}
```



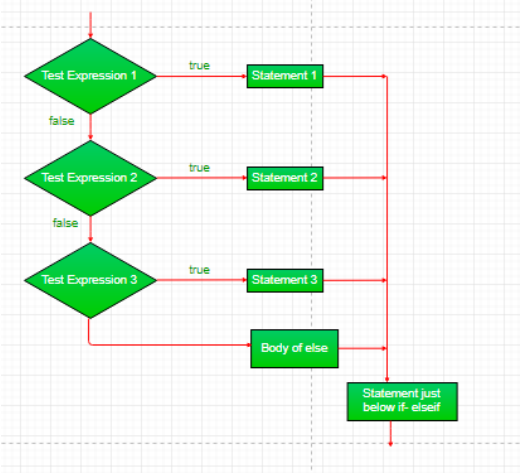
3. nested-if

```
if (condition1)
{
   // Executes when conditio
   if (condition2)
   {
      // Executes when condi
   }
}
```



4. if-else-if

```
if (condition)
    statement;
else if (condition)
    statement;
.
.
else
    statement;
```



5. switch-case

```
switch (expression)
{
  case value1:
    statement1;
    break;
  case value2:
    statement2;
    break;
  .
  .
  case valueN:
    statementN;
    break;
  default:
    statementDefault;
}
```

6. Jump

- **Break:**
  - Terminate a sequence in a switch statement.
  - To exit a loop.

- Used as a “civilized” form of goto.

- **Continue:** Sometimes it is useful to force an early iteration of a loop. That is, you might want to continue running the loop but stop processing the remainder of the code in its body for this particular iteration. This is, in effect, a goto just past the body of the loop, to the loop’s end. The continue statement performs such an action.
- **Return:** The return statement is used to explicitly return from a method. That is, it causes program control to transfer back to the caller of the method.