if-else Construct

The ***if*** statement is Java's conditional branch statement. It can be used to route program execution through two different paths.

Objectives

Upon completion of this topic, we will be able to:

- Understand the syntax and use of if-else construct in Java

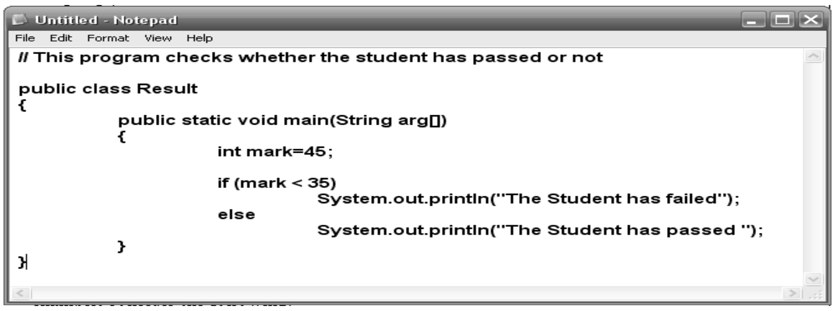
If-else statement

Here is the general syntax of the ***if*** statement:

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

Here, each statement may be a single statement or a compound statement enclosed in curly brackets { } (block). The condition is any expression that returns a **boolean** value. The ***else*** clause is optional.

The ***if*** statement works like this: If the condition is true, then statement1 is executed. Otherwise, statement2 (if it exists) is executed. In no case will both statements be executed.



```
// This program checks whether the student has passed or not

public class Result
{
		public static void main(String arg[])
		{
				int mark=45;

				if (mark < 35)
						System.out.println("The Student has failed");
				else
						System.out.println("The Student has passed ");
		}
}
```

if… else statement example

Most often, the expression used to control the ***if*** will involve the relational operators. However, this is not technically necessary. It is possible to control the **if** using a single **boolean** variable, as shown in this code fragment:

```
boolean dataAvailable;

// ...

if (dataAvailable)

    ProcessData();
```

```
else
        waitForMoreData();
```

Remember, only one statement can appear directly after the **if** or the **else**. If you want to include more statements, you'll need to create a block, as in this fragment:

```
int bytesAvailable;
// ...
if (bytesAvailable > 0) {
        ProcessData();
        bytesAvailable -= n;
} else
waitForMoreData();
```

Here, both statements within the **if** block will execute if **bytesAvailable** is greater than zero. Some programmers find it convenient to include the curly braces when using the **if**, even when there is only one statement in each clause. This makes it easy to add another statement at a later date, and you don't have to worry about forgetting the brackets. In fact, forgetting to define a block when one is needed is a common cause of errors. For example, consider the following code fragment:

```
int bytesAvailable;
// ...
if (bytesAvailable > 0) {
        ProcessData();
        bytesAvailable -= n;
} else
        waitForMoreData();
        bytesAvailable = n;
```

It seems clear that the statement **bytesAvailable = n;** was intended to be executed inside the **else** clause, because of the indentation level. However, as you recall, whitespace is insignificant to Java, and there is no way for the compiler to know what was intended. This code will compile without complaint, but it will behave incorrectly when run.

The preceding example is fixed in the code that follows:

```
int bytesAvailable;
```

```
// ...
if (bytesAvailable > 0) {
        ProcessData();
        bytesAvailable -= n;
} else {
        waitForMoreData();
        bytesAvailable = n;
}
```

The if-else-if Ladder

A common programming construct that is based upon a sequence of nested **if** is the ***if-else-if*** *ladder*. It looks like this:

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

The **if** statements are executed from the top down. As soon as one of the conditions controlling the **if** is **true**, the statement associated with that **if** is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final **else** statement will be executed. The final **else** acts as a default condition; that is, if all other conditional tests fail, then the last **else** statement is performed. If there is no final **else** and all other conditions are **false**, then no action will take place.

Here is a program that uses an **if-else-if** ladder to determine which season a particular month is in.

```
// Demonstrate if-else-if statements.

class IfElse {

        public static void main(String args[ ]) {

                int month = 4; // April

                String season;

                if(month == 12 || month == 1 || month == 2)

                        season = "Winter";

                else if(month == 3 || month == 4 || month == 5)

                        season = "Spring";

                else if(month == 6 || month == 7 || month == 8)

                        season = "Summer";

                else if(month == 9 || month == 10 || month == 11)

                        season = "Autumn";

                else

                        season = "Bogus Month";

                System.out.println("April is in the " + season + ".");

        }

}
```

Here is the output produced by the program:

```
April is in the Spring.
```

You might want to experiment with this program before moving on. As you will find, no matter what value you give **month**, one and only one assignment statement within the ladder will be executed.

Summary

Here are the key takeaways:

- The if statement is Java's conditional branch statement.
- Only one statement can appear directly after the if or the else.
- The if statements are executed from the top down.