NMK20703 OBJECT ORIENTED PROGRAMMING
LAB MODULE 3

JAVA CODES AND COMMAND STATEMENTS

FACULTY OF ELECTRONIC ENGINEERING TECHNOLOGY

Universiti Malaysia Perlis

Learning Outcome:

After completing this lab module, students will be able to:

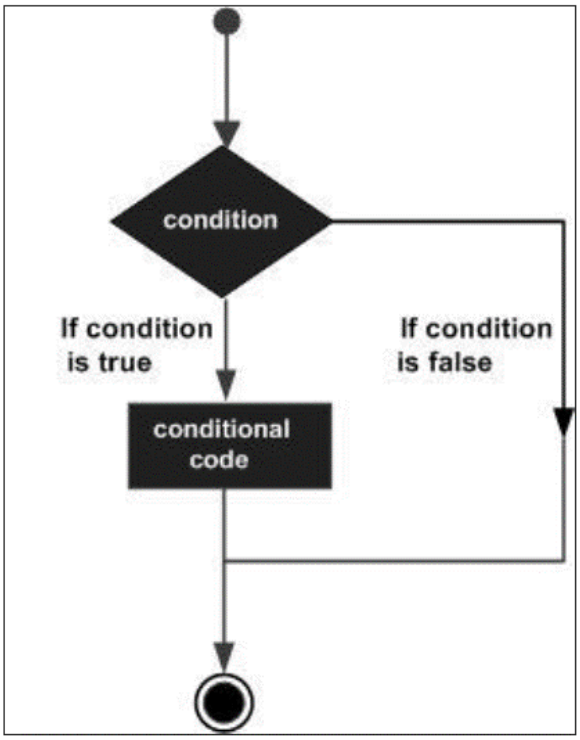
- Use and write Condition statements codes in java
- Write code with nested condition statements
- Write and use code with else statement in JAVA
- Understanding the basics of loops in programming
- Write code with common loops statements in JAVA

Introduction

When you write a computer program, there are times when you need the computer to be more selective about what it does. For example, if you have written a program to balance your checkbook, you might want the computer to display a warning message if your account is overdrawn. The computer should display this message only if your account is overdrawn. If it isn't, the message would be inaccurate and emotionally upsetting. The way to accomplish this task in a Java program is to use a *conditional*, a statement that causes something to happen in a program only if a specific condition is met.

When a Java program makes a decision, it does so by employing a conditional statement. During this hour, you check the condition of things in your Java programs using the conditional keywords if, else, switch, case, and break. You also use the conditional operators ==, !=, <, >, <=, >=, and ?, along with Boolean variables.

Decision making structures have one or more conditions to be evaluated or tested by the program, along with a statement or statements that are to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false. Following is the general form of a typical decision making structure found in most of the programming languages:

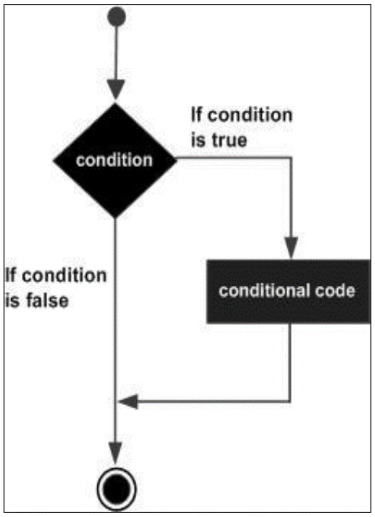


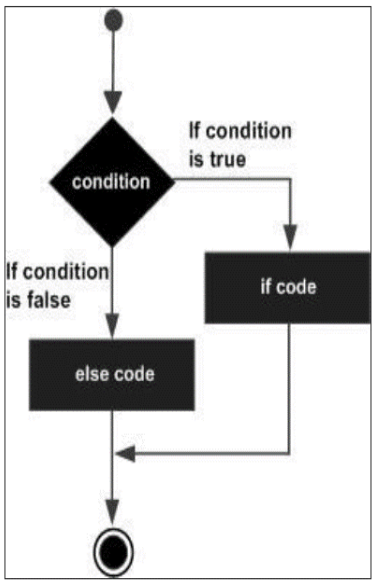
Java programming language provides the following types of decision-making statements.

Statement	**Description**
if statement	An **if statement** consists of a Boolean expression followed by one or more statements.
If..,else statement	An **if statement** can be followed by an optional **else statement**, which executes when the Boolean expression is false.
nested if statement	You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).
switch statement	A **switch** statement allows a variable to be tested for equality against a list of values.

Decision control statements in JAVA

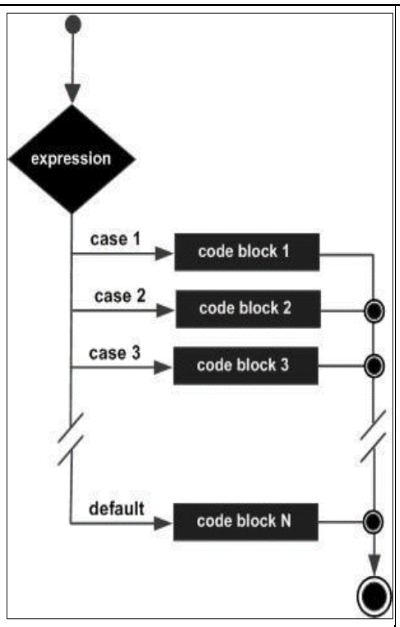
Syntax	Flow Diagram	Example

1- If Statement
 	
`if(Boolean_expression )`
`{`
`//Statements will execute if the Boolean expression is true`
`}`
		`public class Test {`
		`public static void main(String args[]){`
		`int x = 10;`
		`if( x < 20 ){`
		`System.out.print("This is if statement");`
		`}`
		`}`
		`}`

2- If-else Statement
`if(Boolean_expression ){`
`//Executes when the Boolean expression is true`
`}else{`
`//Executes when the Boolean expression is false`
`}`
		`public class Test {`
		`public static void main(String args[]){`
		`int x = 30;`
		`if( x < 20 ){`
		`System.out.print("This is if statement");`
		`}else{`
		`System.out.print("This is else statement");`
		`}`
		`}`
		`}`

3- Switch Statement

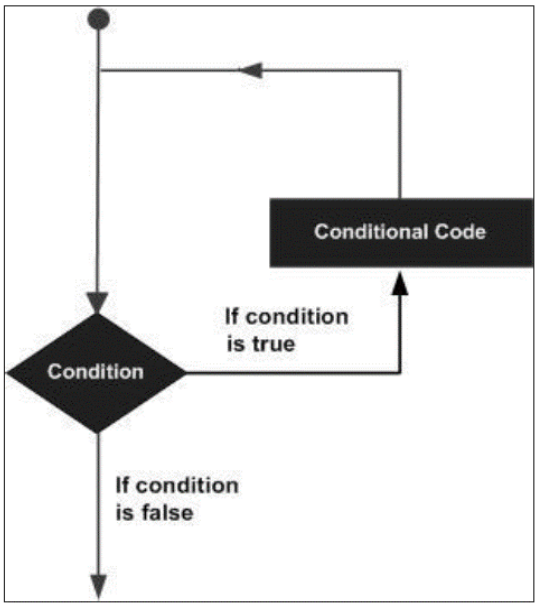
```
switch(expression){
case value :
//Statements
break; //optional
case value :
//Statements
break; //optional
//You can have any number of case statements.
default : //Optional
//Statements
}
```




```
public class Test {
public static void main(String args[]){
//char grade = args[0].charAt(0);
char grade = 'C';
switch(grade)
{
case 'A' :
System.out.println("Excellent!");
break;
case 'B' :
case 'C' :
System.out.println("Well done");
break;
case 'D' :
System.out.println("You passed");
case 'F' :
System.out.println("Better try again");
break;
default :
System.out.println("Invalid grade");
}
System.out.println("Your grade is " + grade);
}
}
```

Loops

A **loop** statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages:

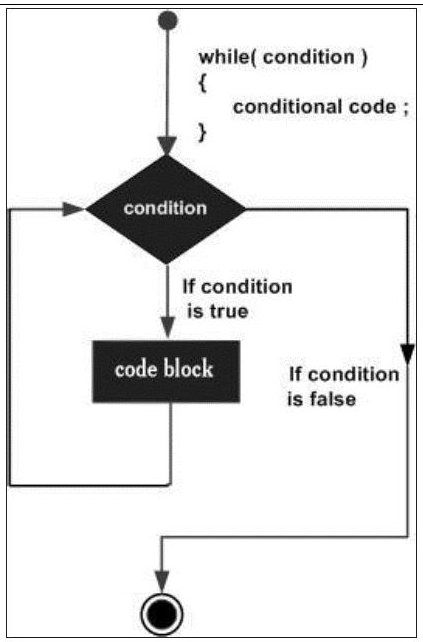


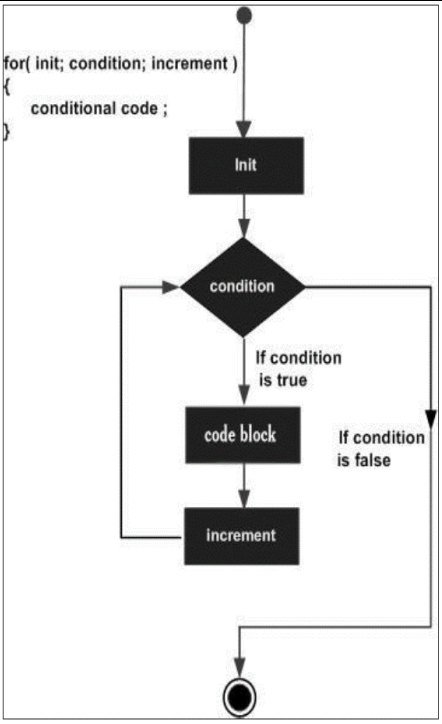
Java programming language provides the following types of loops to handle looping requirements. Click the following links to check their detail.

Loop type	Description
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	Like a while statement, except that it tests the condition at the end of the loop body.

Loop statements in JAVA

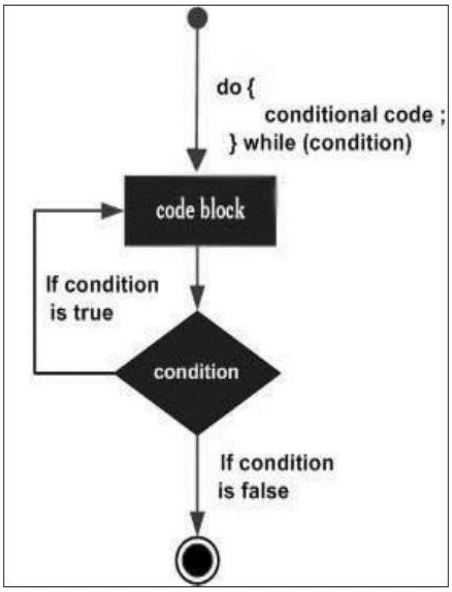
statement	Flow Diagram	Example

1- While Loop in Java
`while(Boolean_expression)`
`{`
`//Statements`
`}`
		`public class Test {`
		`public static void main(String args[]) {`
		`int x = 10;`
		`while( x < 20 ) {`
		`System.out.print("value of x : " + x );`
		`x++;`
		`System.out.print("\n");`
		`}`
		`}`
		`}`

2- for Loop in Java
`for(initialization;`
`Boolean_expression;`
`update)`
`{`
`//Statements`
`}`
		`public class Test {`
		`public static void main(String args[]) {`
		`for(int x = 10; x < 20; x = x+1) {`
		`System.out.print("value of x : " + x );`
		`System.out.print("\n");`
		`}`
		`}`
		`}`

3- Do While Loop in Java

```
do
{
//Statements
}while(Boolean_expression)
;
```




```
public class Test {
public static void
main(String args[]){
int x = 10;
do{
System.out.print("valu
e of x : " + x );
x++;
System.out.print("\n")
;
}while( x < 20 );
}
}
```

Loop Control Statements

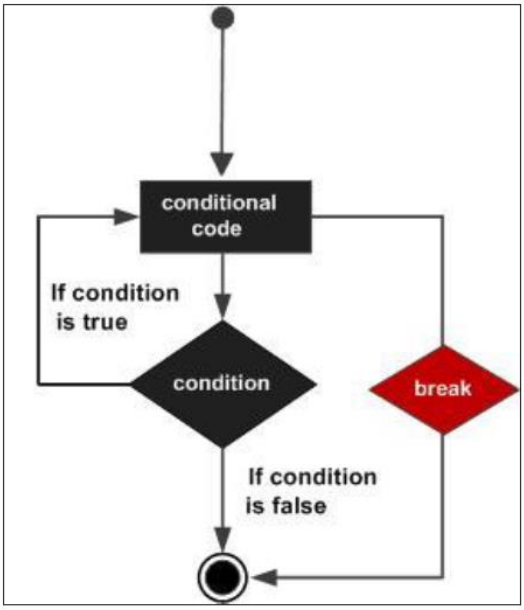
Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

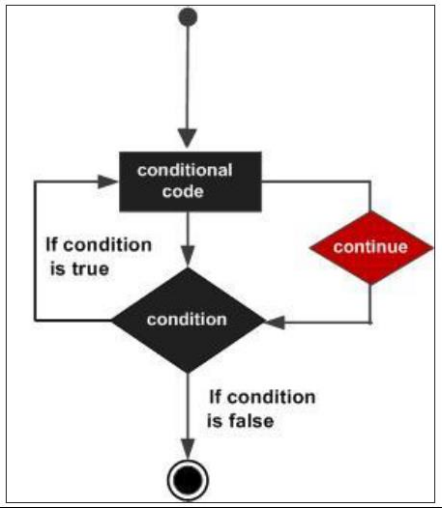
Java supports the following control statements. Click the following links to check their detail.

Control statement	Description
break statement	Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

Break statements Description

statement	Flow Diagram	Example

break;		public class Test {
		public static void main(String args[]) {
		int [] numbers = {10, 20, 30, 40, 50};
		for(int x : numbers) {
		if(x == 30) {
		break;
		}
		System.out.print(x);
		System.out.print("\n");
		}
		}
		}

continue;		public class Test {
		public static void main(String args[]) {
		int [] numbers = {10, 20, 30, 40, 50};
		for(int x : numbers) {
		if(x == 30) {
		continue;
		}
		System.out.print(x);
		System.out.print("\n");
		}
		}
		}

Enhanced for loop in Java

As of Java 5, the enhanced for loop was introduced. This is mainly used to traverse collection of elements including arrays.

Syntax

```
for(declaration : expression)
{
//Statements
}
```

- **Declaration:** The newly declared block variable, is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.

- **Expression:** This evaluates to the array you need to loop through. The expression can be an array variable or method call that returns an array.

Example

```
public class Test {
public static void main(String args[]){
int [] numbers = {10, 20, 30, 40, 50};
for(int x : numbers ){
System.out.print( x );
System.out.print(",");
}
System.out.print("\n");
String [] names ={"James", "Larry", "Tom", "Lacy"};
for( String name : names ) {
System.out.print( name );
System.out.print(",");
}
}
}
```

Execution of the above code will produce:

```
10,20,30,40,50,

James,Larry,Tom,Lacy,
```

Answered Question

A. I declared a variable inside a block statement for an if. When the if was done, the definition of that variable vanished. Where did it go?

Answer:

In technical terms, block statements form a new lexical scope. This means that if you declare a variable inside a block, it's visible and usable only inside that block. When the block finishes executing, all the variables you declared go away.

It's a good idea to declare most of your variables in the outermost block in which they'll be needed—usually at the top of a block statement. The exception might be simple variables, such as index counters in for loops, where declaring them in the first line of the for loop is an easy shortcut.

B. Why can't I use switch with strings?

Answer:

You can. If it isn't working in NetBeans, you must make sure that you have a current version of Java installed and your development environment has been set up to use it. In NetBeans, to see whether the current project is set up for Java 8, choose File, Project Properties to open the properties dialog. Choose Librariesvin the Categories pane; then set Java Platform to JDK 8 if it isn't already.

Exercises:

1- What kind of loop is used to execute the statements in the loop at least once before the conditional expression is evaluated?

A. do-while

B. for

C. while

2- Which of the following cannot be used as the test in a case statement?

A. characters

B. strings

C. objects

3- Which instance variable of an array is used to find out how big it is?

A. size

B. length

C. MAX_VALUE

4- Given the code in (Figure Ex 4) bellow, What will be the value of x when it is displayed?

A. 9.0

B. 11.0

C. 15.0

D. The program will not compile.

```
public class Cases {
    public static void main(String[] arguments) {
        float x = 9;
        float y = 5;
        int z = (int)(x / y);
        switch (z) {
            case 1:
                x = x + 2;
            case 2:
                x = x + 3;
            default:
                x = x + 1;
        }
        System.out.println("Value of x: " + x);
    }
}
```

Figure Exercise 4.

5- Create a class that takes words for the first 10 numbers (“one” to “ten”) and converts them into a single long integer. Use a switch statement for the conversion and command-line arguments for the words.