JAVA CONTROL STATEMENTS

TEAM CRYSTALS

Members:

Anuja Ajay

Aysha Fathima

Nimisha R S

Sankari Suresh

JAVA CONTROL STATEMENTS

- Decision Making statements
  - if statements
  - switch statement
- Loop statements
  - for loop
  - for-each loop
  - while loop
  - do while loop
- Jump statements
  - break statement
  - continue statement

1) Decision-Making statements

If Statement

if is used to evaluate a condition. In Java, there are four types of if-statements given below.

1. Simple if statement
2. if-else statement
3. if-else-if ladder
4. Nested if-statement

Simple if statement

It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.

Syntax of if statement:

```
if(condition) {
statement 1;
}
```

Example :**Student.java**

```
public class Student {
public static void main (String[]  args) {
int x = 10;
int y = 12;
if(x+y > 20) {
System.out.println(" x + y is greater than 20 ");
}
}
}
```

Output :

```
x + y is greater than 20
```

if-else statement

The else block is executed if the condition of the if-block is evaluated as false.

Syntax of if-else statement:

```
if(condition) {
statement 1; //executes when condition is true
}
else{
statement 2; //executes when condition is false
}
```

Example: **Student.java**

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y < 10) {
System.out.println("x + y is less than      10");
}   else {
System.out.println("x + y is greater than 20");
}
}  }
```

Output

```
x + y is greater than 20
```

if-else-if ladder

The if-else-if statement contains the if-statement followed by multiple else-if statements. If-else statements that create a decision tree where the program may enter in the block of code where the condition is true.

Syntax of if-else-if statement:

```
if(condition 1) {
statement 1; //executes when condition 1 is true
}
else if(condition 2) {
statement 2; //executes when condition 2 is true
}
else {
statement 2; //executes when all the conditions are false
}
```

Example: **Student.java**

```
public class Student {
public static void main(String[] args) {
String city = "Delhi";
if(city == "Meerut") {
System.out.println("city is meerut");
}else if (city == "Noida") {
System.out.println("city is noida");
}else if(city == "Agra") {
System.out.println("city is agra");
}else {
System.out.println(city);
}
}
}
```

<u>Output</u>

```
Delhi
```

<u>Nested if-statement</u>

In nested if-statements, the if statement can contain a **if** or **if-else** statement inside another if or else-if statement.

Syntax of Nested if statement:

```
if(condition 1) {
statement 1; //executes when condition 1 is true
if(condition 2) {
statement 2; //executes when condition 2 is true
}
else{
statement 2; //executes when condition 2 is false
}
}
```

Example:**Student.java**

```
 public class Student {
public static void main(String[] args) {
String address = "Delhi, India";
if(address.endsWith("India")) {
if(address.contains("Meerut")) {
System.out.println("Your city is Meerut");
}else if(address.contains("Noida")) {
System.out.println("Your city is Noida");
}else {
System.out.println(address.split(",")[0]);
}
}else {
System.out.println("You are not living in India");
}
}
```

<u>Output</u>

```
Delhi
```

Switch Statement

The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched.

Syntax of switch statement:

```
switch (expression){
   case value1:
    statement1;
    break;
   .
   .
```

```
            case valueN:
             statementN;
             break;
            default:
             default statement;
   }
```

Example:**Student.java**

```
public class Student implements Cloneable {
public static void main(String[] args) {
int num = 2;
switch (num){
case 0:
System.out.println("number is 0");
break;
case 1:
System.out.println("number is 1");
break;
default:
System.out.println(num);
}
}
}
```

<u>Output</u>

```
2
```

2) Loop Statements

<u>For loop</u>

It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.

Syntax of for loop:

```
for(initialization; condition; increment/decrement){
//statement or code to be executed
}
```

Example: **Calculation.java**

```
public class Calculattion {
public static void main(String[] args) {
// TODO Auto-generated method stub
int sum = 0;
for(int j = 1; j<=10; j++) {
sum = sum + j;
}
System.out.println("The sum of first 10 natural numbers is " + sum);
}
}
```

<u>Output</u>

```
The sum of first 10 natural numbers is 55
```

Nested for Loop

for loop inside the another loop

for-each Loop

for-each loop is used to traverse array or collection in Java

Syntax of for-each:

```
for(data_type variable : array_name){

//code to be executed

}
```

Example: Calculation.java

```
public class Calculation {

public static void main(String[] args) {

// TODO Auto-generated method stub

String[] names = {"Java","C","C++","Python","JavaScript"};

System.out.println("Printing the content of the array names:\n");

for(String name:names) {

System.out.println(name);

}

}

}
```

Output

```
Printing the content of the array names:

Java

C

C++

Python

JavaScript
```

while loop

The while loop is also used to iterate over the number of statements multiple times. It is also known as the entry-controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

Syntax of the while loop:

```
while(condition){
//looping statements
}
```

Example: **Calculation .java**

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n");
while(i<=10) {
System.out.println(i);
i = i + 2;
}
}
}
```

Output

```
Printing the list of first 10 even numbers

0
2
4
6
8
10
```

do-while loop

The do-while loop checks the condition at the end of the loop after executing the loop statements. When the number of iteration is not known and we have to execute the loop at least once, we can use do-while loop.

Syntax of the do-while loop:

```
do
{
//statements
} while (condition);
```

Example: **Calculation.java**

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n");
do {
System.out.println(i);
i = i + 2;
}while(i<=10);
}
}
```

Output

```
Printing the list of first 10 even numbers
0
2
4
6
8
10
```

3) Jump Statements

1.Break

When a break statement is encountered inside a loop, the loop is immediately terminated, and the program control resumes at the next statement following the loop.

The Java *break* statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition.

Syntax:

```
jump-statement;
break;
```

Example: **BreakExample.java**

```
public class BreakExample {

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

```
// TODO Auto-generated method stub

for(int i = 0; i<= 10; i++) {

System.out.println(i);

if(i==6) {

break;

    }  }

}  }
```

Output

```
0

1

2

3

4

5

6
```

2. Continue

It continues the current flow of the program and skips the remaining code at the specified condition.

Syntax:

```
jump-statement;

continue;
```

Example:**ContinueExample.java**

```
public class ContinueExample {

public static void main(String[] args) {

// TODO Auto-generated method stub

for(int i = 0; i<= 2; i++) {

for (int j = i; j<=5; j++) {

 if(j == 4) {
continue;

}  System.out.println(j);

}  }}
```

<u>Output</u>

```
0
1
2
3
5
1
2
3
5
2
3
5
```