Zoho Schools for Graduate Studies

Notes

Control Statement

1. What are Control Statements?

In programming, statements are executed sequentially, one after another, by default. However, sometimes we need to change the flow of execution based on certain conditions or requirements. This is where Control Statements are used.

Definition:Control statements are programming constructs that alter the normal, sequential execution of code.

Purpose:They enable decision-making, looping, and unconditional jumps in a program.

2. Categories of Control Statements:

Control statements can be broadly classified into two types:

Conditional Control Statements:

These control the flow of execution based on whether a condition is true or false.

Examples: if, if-else, if-else-if ladder, switch-case, ternary operator (?:).

Unconditional Control Statements:

These transfer control from one part of the program to another without evaluating any condition.

Examples: break, continue, return, goto (in some languages).

Conditional Control Statements:

1. if Statement :

Syntax:

```
if (condition) {
   // Code executed when condition is true
}
```

Explanation:

Executes a block of code only if the condition evaluates to true..

Example:

```
int age = 18;
if (age >= 18) {
System.out.print("You are eligible to vote.");
}
```

2. if-else Statement:

Syntax:

```
if (condition) {
   // Code executed if condition is true
} else {
   // Code executed if condition is false
}
```

Explanation:

Provides an alternative block of code when the condition is false.

Example:

```
int marks = 40;
if (marks >= 50) {

System.out.print("Pass");

} else {

System.out.print("Fail");

}
```

3. if-else if Ladder:

Syntax:

```
if (condition1) {

 // Code block 1

}

else if (condition2) {

  // Code block 2

} else {    // Default block

}
```

Explanation: Used to check multiple conditions in sequence.

Example:

```
int marks = 85;
 if (marks >= 90) {
 System.out.print("Grade A");
 }
else if (marks >= 75) {

System.out.print("Grade B");

}
 else {


System.out.print("Grade C");
 }
```

4. switch-case Statement:

Syntax:

```
switch (expression) {
case value1:    // Code block 1
break;
case value2:    // Code block 2
 break;
 default:       // Default block

}
```

Explanation:

Used when multiple values of a single expression are compared.

Example:

```
 int day = 3;switch (day) {

 case 1: System.out.print("Monday"); break;
 case 2: System.out.print("Tuesday"); break;

  case 3: System.out.print("Wednesday"); break;
default: System.out.print("Invalid Day");

}
```

5. Ternary Operator (?:):

Syntax:

```
variable = (condition) ? value_if_true : value_if_false;
```

Explanation:
condition – An expression that results in either true or false.
value_if_true – Value assigned to variable if condition is true.
value_if_false – Value assigned to variable if condition is false.

Example:

```
int age = 20;

String result = (age >= 18) ? "Adult" : "Minor";

System.out.println(result);
```

Types of Unconditional Control Statements in Java:

1. break Statement:

Purpose:

Used to terminate a loop or a switch statement immediately and transfer control to the next statement outside the loop/switch.

Syntax:

```
break;
```

Example:

```
for (int i = 1; i <= 5; i++) {
if (i == 3) {
 break;  // Exits loop when i = 3
 }
 System.out.println(i); //output=1 2

}
```

2. continue Statement:

Purpose:
Skips the current iteration of a loop and continues with the next iteration.

Syntax:

```
continue;
```

Example:

```
for (int i = 1; i <= 5; i++) {
 if (i == 3) {
 continue;  // Skips printing when i = 3

    }

System.out.println(i);  //output 1 2 4 5

}
```

3. return Statement:

Purpose:

Ends the execution of a method and optionally returns a value.

Syntax:

```
  return;     // For methods with void return type

return value;  // For methods returning a value
```

Example:

```
public static int add(int a, int b) {

return a + b;  // Returns sum to caller

}

public static void main(String[] args) {

 System.out.println(add(5, 3));  //output=8

}
```

4. goto Statement (Not in Java):

Note:Java does not support goto because it makes code difficult to read and maintain.Instead, Java uses labels with break or continue to control flow in nested loops.

Example (Using Labeled Break):

```
outer:for (int i = 1; i <= 3; i++) {

 for (int j = 1; j <= 3; j++) {

 if (i == 2 && j == 2) {

 break outer;  // Exits both loops

}
 System.out.println(i + " " + j); //output=1 1 1 2 1 3 2 1
 }

}
```

1) what is the output of the following code upon execution?

```
public class First{
    public static void main(String[] args){
     int x=1;
   if(x)
   System.out.println("Success");
   else
   System.out.println("Failure");
}
}
```

a) compiler error

b) runtime error

c)java

d)none of above

Reason:

The image shows a code snippet with an if(x) condition. In Java, the condition inside an if statement must be a boolean value (true or false).

An int value cannot be directly used as a boolean condition. This would result in a compiler error.

2) what is the output of the following code upon execution?

```
  public class First{
   public static void main(String[] args){
    int x=1;String s=null;
    if(x==s)
    System.out.ptintln("success");
    else
    System.out.println("Failure");
    }
    }
```

a) compiler error

b) runtime error

c)java

d)none of above

Reason:

Another image shows if(x==s), where x is an int and s is a StringYou cannot directly compare a primitive

type (int) with a reference type (String) using the == operator.This will also cause a compiler error.

3) what is the output of the following code upon execution?

```
public class First{

public static void main(String[] args){
  int x=10,y=23;
   boolean b=x>=y;
   if(b==true)
   System.out.ptintln("Success");
   else
   System.out.println("Failure");
  }
  }
```

a) compiler error
b) run time error
c)success
d)Failure

Reason:

The statement b=true is an assignment operator (=), not a comparison operator (==).
It assigns the value true to b.The result of an assignment operation is the value that was assigned

Therefore, b=true evaluates to true, and the if condition is met.As a result, the code will print Success.

Tautology:

Definition:A proposition is called a tautology if it is always true, regardless of the truth values of its individual components.

Contradiction:

Definition:A proposition is called a contradiction if it is always false, regardless of the truth values

of its individual components.

4)what is the output of the following code upon execution?

```
public class First{
 public static void main(String[] args){
     int=4;
      long y=x*4-x++;
      if(y>12)
      System.out.println("Success");
      else
      System.out.println("Failure");
      else
      System.out.println("Draw");
      }

     }
```

Reason:

This final example shows an if-else structure with an extra else clause. * An if statement can only have one corresponding else block.
Having two else blocks is a syntax error.This will result in a compiler error.

5) **For what value is 'x' the following the code yields "Failure" of output?**

```
public class First{
 public static void main(String[] args){
     if(x<4){

     System.out.println("success");

     else if(x<3)

     System.out.println("Failure");
```

a) **values greater then x**

b) **values lesser than x**

c) **For value of x=3**

d) **none of the above**

Reason:

The image shows if (x<4) ... else if (x<3). An if-else if ladder evaluates conditionssequentially.The first condition x<4 is checked.If x=3, this condition is true, and "Success" is printed.

The else if block is never checked because the first if condition was met.

This demonstrates how the ladder structure works: once a condition is true,the rest of the ladder is skipped.

6) **what is the output of the following code upon execution?**

```
public class First{

 public static void main(String[] args){
 int x=1,y=1
 int z=x>5?y++:x++;
 System.out.println(x+ " " +y+ " " +z);
 }
}
```

Reason:
The code int z=x>5 ? y++ : x++; is a ternary operator (also known as a conditional operator).

This is a shorthand for an if-else statement.

Here, x is 1.

The condition x>5 is false.

The code after the colon (:) is executed: x++.

The value of x is incremented to 2.

The value of the expression x++ (which is the original value of x, 1) is assigned to z.

The output would be: 2 1 1.

7) what is the output of the following code upon execution?

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

int x=20/3;

switch(x){

case  3:System.out.println("1");

case  5:System.out.println("5");

case  2:System.out.println("2");

default:System.out.println("3");

}

}

}
```

Reason:

The switch statement is used to execute one of many code blocks based on a value. *

Here, x is 20/3, which in integer division is 6.

The switch statement checks the value of x against the case labels.

None of the case labels (3, 5, or 2) match the value 6.

The code falls through to the default block.

The output will be: 3.