Paste HTML element by inspecting here!

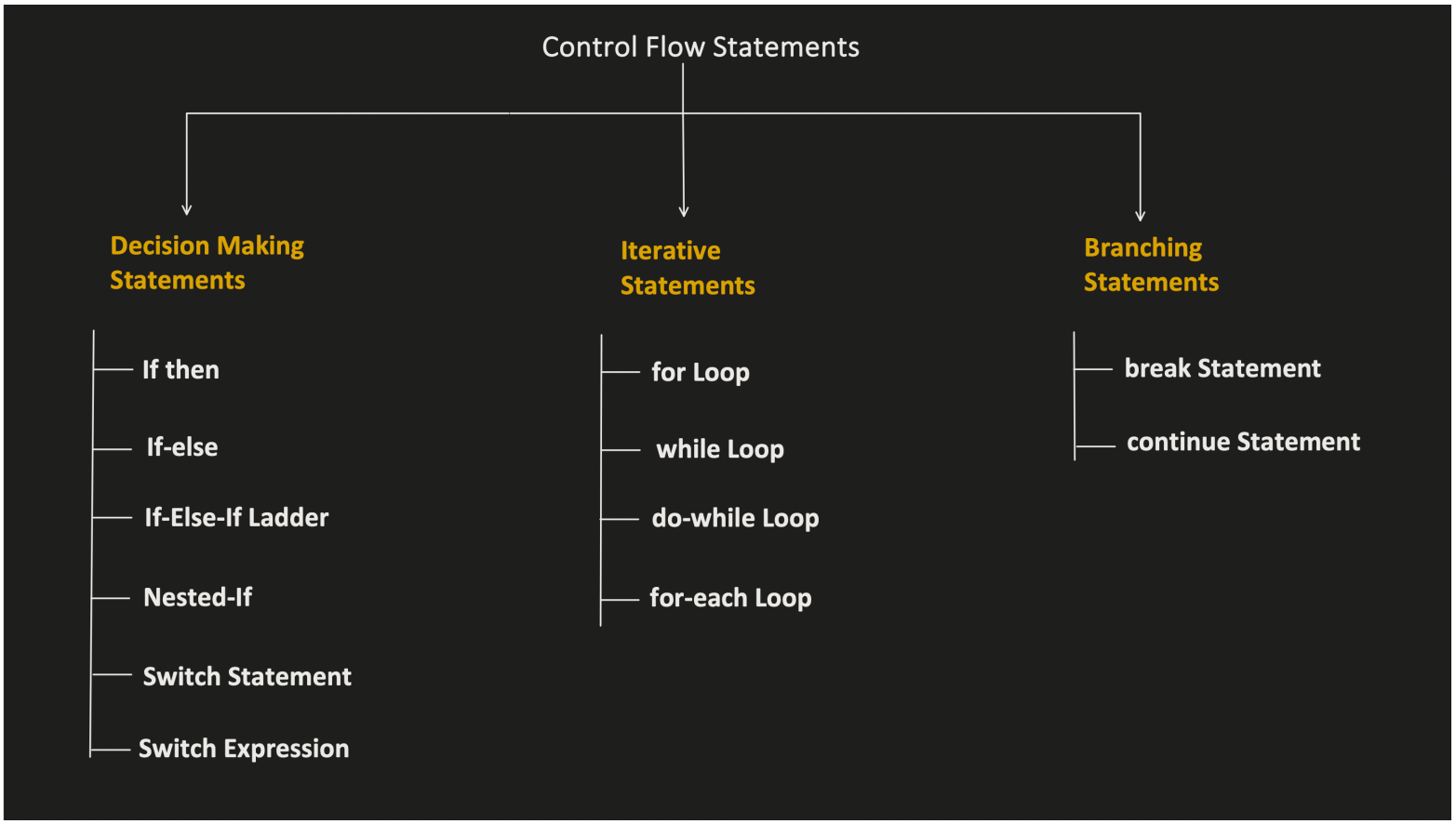
d="u3i1sca144313d213484788aa34cc342e15e9">

/div>
     </text-note-view><template is="component"
omponent-name="text-note-view"> </template>

Download as PDF

JAVA: Control flow statements

"Concept && Coding" YT Video Notes



Decision Making Statements

1. **If then (Simple If): If condition is true, then if-block will get executed.**

```
if(boolean condition) {
        //code here get execute when condition is true
}
```

```java
public class Main {

    public static void main(String[] args) {

        int val = 10;

        if(val > 8){
            System.out.println("only get executes when val is greater than 8 ");
        }
        System.out.println("this code will executes, even if val is less than 8");
    }
}
```

Output:

```
only get executes when val is greater than 8
this code will executes, even if val is less than 8

Process finished with exit code 0
```

2. **If-else**: if condition is true, if-block will get executed else if condition is false, then else-block will get executed.

```java
public class Main {

    public static void main(String[] args) {

        int val = 7;

        if(val > 8){
            System.out.println("only get executes when val is greater than 8 ");
        }
        else {
            System.out.println("this code will executes, if val is less than or equals to 8");

        }
        System.out.println("this code will executes, no matter condition is true or false");
    }
}
```

Output:

```
this code will executes, if val is less than or equals to 8
this code will executes, no matter condition is true or false

Process finished with exit code 0
```

3. **If-else-if Ladder**: It contains if-statement with multiple (chain of) else-if statements. It evaluate the condition from top and goes down and any condition gets true, its block will get executed.

```java
public class Main {

    public static void main(String[] args) {

        int val = 13;

        if(val == 1){
            System.out.println("val is 1");
        }
        else if(val == 2) {
            System.out.println("val is 2");
        }
        else if(val == 3) {
            System.out.println("val is 3");
        }
        else {
            System.out.println("val is: " + val);
        }

        System.out.println("this code will executes anyhow");
    }
}
```

Output:

```
val is: 13
this code will executes anyhow
```

4. Nested-If Statement: If-else statement within If-block or else-block.

```java
public class Main {

    public static void main(String[] args) {

        int val = 13;

       if(val > 8){
           System.out.println("value is greater than 8 ");

           if(val < 15){
               System.out.println("value is greater than 8 but else than 15");
           }
           else {
               System.out.println("value is greater than 15");
           }
       }
       else{
           System.out.println("value is less than equal to 8");
       }

        System.out.println("this code will executes anyhow");
    }
}
```

Output:

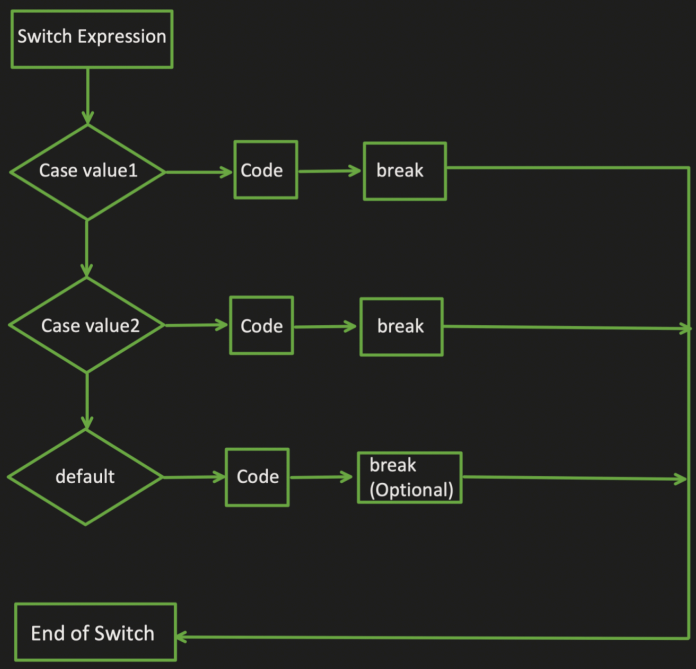
```
value is greater than 8
value is greater than 8 but else than 15
this code will executes anyhow

Process finished with exit code 0
```

5. Switch Statement: It is similar to *if-else-if* ladder, based on condition particular block will get executed out of many alternatives.

Syntax:

```
switch(expression){
        case value_1:
              //code statements
              break;
        case value_2:
              //code statements
              break;
         ...
         ...
        case value_N:
             //code statements
             break;
       default:
            //default statements
            break;  // (optional)
}
```



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

        int a = 1;
        int b = 2;

        switch (a + b) {
            case 1:
                System.out.println("a+b is 1");
                break;
            case 2:
                System.out.println("a+b is 2");
                break;
            case 3:
                System.out.println("a+b is 3");
                break;
            default:
                System.out.println(a + b);
        }
    }
}
```

Output:

```
a+b is 3

Process finished with exit code 0
```

Without break statement

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

        int a = 1;
        int b = 2;

        switch (a + b) {
            case 1:
                System.out.println("a+b is 1");
                break;
            case 2:
                System.out.println("a+b is 2");
                break;
            case 3:
                System.out.println("a+b is 3");
            case 4:
                System.out.println("a+b is 4");
            default:
                System.out.println(a + b);
        }
    }
}
```

Output:

```
a+b is 3
a+b is 4
3

Process finished with exit code 0
```

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

        int a = 1;
        int b = 2;

        switch (a + b) {
            case 1:
                System.out.println("a+b is 1");
                break;
            default:
                System.out.println(a + b);
            case 2:
                System.out.println("a+b is 2");
                break;
            case 3:
                System.out.println("a+b is 3");
            case 4:
                System.out.println("a+b is 4");

        }
    }
}
```

Output:

```
a+b is 3
a+b is 4

Process finished with exit code 0
```

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

        int a = 1;
        int b = 9;

        switch (a + b) {
            case 1:
                System.out.println("a+b is 1");
                break;
            default:
                System.out.println(a + b);
            case 2:
                System.out.println("a+b is 2");
                break;
            case 3:
                System.out.println("a+b is 3");
            case 4:
                System.out.println("a+b is 4");

        }
    }
}
```

Output:

```
10
a+b is 2

Process finished with exit code 0
```

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


        String month = "March";
        switch (month) {
            case "January":
            case "February":
            case "March":
                System.out.println("month value is in Q1");
                break;
            case "April":
            case "May":
            case "June":
                System.out.println("month value is in Q2");
                break;
            default:
                System.out.println("month value is in Q3 or Q4");
        }
    }
}
```

Output:

```
month value is in Q1

Process finished with exit code 0
```

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

        String month = "March";
        switch (month){
            case "January", "February", "March":
                System.out.println("month value is in Q1");
                break;
            case "April", "May", "June":
                System.out.println("month value is in Q2");
                break;
            default:
                System.out.println("month value is in Q3 or Q4");
        }
    }
}
```

Output:

```
month value is in Q1

Process finished with exit code 0
```

Few things we need to take care:

1. Two Cases can not have the same value.
2. Switch expression data type and case values/constant data type should be same.
3. Case value should be either LITERAL or CONSTANT.

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

        int value = 1;

        switch (2+1-2) {
            case value:
                System.out.println("some code here");
            default:
                System.out.println("default code here");
        }
    }
}
```

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

        final int value = 1;

        switch (2+1-2) {
            case value:
                System.out.println("some code here");
            default:
                System.out.println("default code here");
        }
    }
}
```

4. All use case need not to be handled:

```java
enum Day{
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

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

    Day dayEnumVal = Day.FRIDAY;
    int outputValue = 0;

    switch (dayEnumVal){
        case MONDAY:
            outputValue = 1;
            break;
        case TUESDAY:
            outputValue = 2;
            break;
        case WEDNESDAY:
            outputValue = 3;
            break;
        case THURSDAY:
            outputValue = 4;
            break;
    }

    System.out.println(outputValue);
}
```

Output:

```
0

Process finished with exit code 0
```

5. Nested Switch statement is possible:

```
enum Day{
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

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

    Day dayEnumVal = Day.MONDAY;
    int outputValue = 0;

    switch (dayEnumVal){
        case MONDAY:
            outputValue = 1;
            switch (outputValue){
                case 1:
                    System.out.println("output value:" + 1);
                    break;
                case 2:
                    System.out.println("output value:" + 2);
                    break;
                default:
                    System.out.println("output value:" + outputValue);
            }
            break;
        case TUESDAY:
            outputValue = 2;
            break;
        case WEDNESDAY:
            outputValue = 3;
            break;
        case THURSDAY:
            outputValue = 4;
            break;
    }

    System.out.println(outputValue);
}
```

Output:

```
output value:1
1

Process finished with exit code 0
```

6. Supported data types:
   - 4 Primitive types: int, short, byte, char
   - Wrapper Types of above primitive data types i.e. Integer, Short, Byte, Character.
   - Enum
   - String

7. Return is not possible within Switch case.

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

        String day = "";
        int val = 1;
         String outputVal = switch (val){

            case 1 :
                return "One";
        }
        System.out.println(day);
    }
}
```

There are 2 ways to do it:

1. Using "case N ->" Label.
2. Using "yield" statement.

Use of "case N ->" :

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

        String day = "";
        int val = 1;
         String outputVal = switch (val){

            case 1 -> "One";
            case 2 -> "Two";
            default -> "None";
        };
        System.out.println(outputVal);
    }
}
```

1. All possible use cases need to be handled for the expression.

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

        String day = "";
        int val = 1;
         String outputVal = switch (val){

            case 1 -> "One";
            case 2 -> "Two";
        };
        System.out.println(outputVal)
    }
}
```

'switch' expression does not cover all possible input values

Insert 'default' branch ⌥⇧⏎ More actions... ⌥⏎

int val = 1

untitled

2. Using this "->" we can not have block of statements. If we want block of statements and then return the value, we need to use "yield".

Use of "yield" :

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

        String day = "";
        int val = 1;
        String outputVal = switch (val){
            case 1 -> {
                //some code logic here
                yield "One";
            }
            case 2 -> {
                //some code logic here
                yield "Two";
            }
            default -> "None";
        };

        System.out.println(outputVal);
    }
}
```

Iterative Statements:

1. For Loop:

for(*initialization of variable*; *condition check*; increment/decrement of variable) {
//statement block
}

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

      for(int val=1; val<=10; val++){
          System.out.println(val);
      }
    }
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10

Process finished with exit code 0
```

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

      for(int x=1; x<=3; x++){
          for(int y=1; y<=3; y++){
              System.out.println("x="+x + " : y=" +y);
          }
      }
    }
}
```

```
x=1 : y=1
x=1 : y=2
x=1 : y=3
x=2 : y=1
x=2 : y=2
x=2 : y=3
x=3 : y=1
x=3 : y=2
x=3 : y=3

Process finished with exit code 0
```

2. While Loop:

Initialize variable;
While(condition check){
//block of statements
increment/decrement variable
}

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

      int val = 1;
      while(val<=5){
          System.out.println(val);
          val++;
      }
    }
}
```

Output:

```
1
2
3
4
5

Process finished with exit code 0
```

3. Do-While Loop:

```
Initialize variable;
do{
      //block of statements
  increment/decrement variable
}while(condition check);
```

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

      int val = 1;
      do{
          System.out.println(val);
          val++;
      }while (val<=5);
    }
}
```

Output:

```
1
2
3
4
5

Process finished with exit code 0
```

4. For each loop:

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

        int valArray[] = {1,2,3,4,5};

        for(int val : valArray){
            System.out.println(val);
        }
    }
}
```

Output:

```
1
2
3
4
5

Process finished with exit code 0
```

Branching Statements:

1. break statement:

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

        for(int val=1; val<=10 ; val++){

            if(val == 3){
                break;
            }
            System.out.println(val);
        }
    }
}
```

Output:

```
1
2

Process finished with exit code 0
```

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

        for (int outerLoop = 1; outerLoop <= 5; outerLoop++)
            for (int innerLoop = 1; innerLoop <= 5; innerLoop++) {

                if (innerLoop == 2) {
                    break;
                }
                System.out.println(outerLoop + "," + innerLoop);
            }
    }
}
```

Output:

```
1,1
2,1
3,1
4,1
5,1

Process finished with exit code 0
```

2. continue statement:

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

        for(int val=1; val<=10 ; val++){

            if(val == 3){
                continue;
            }
            System.out.println(val);
        }
    }
}
```

Output:

```
1
2
4
5
6
7
8
9
10

Process finished with exit code 0
```