Control statements

What you’ll learn

- Introduction to control statements
- Types
- Decision-making statements and its types
- Jump statements and its types
- Loops and its types
- Code snippets
- Gaming scenario

Introduction to control statements

- controls the flow of a program
- determines whether the other **statements** will be executed or not
- Types:
    1) Decision making statements
    2) Jump statements
    3) Loops

Decision making statements

- Evaluates the expression
- Controls the flow of code based on the condition result
- Two types:
    1) if
    2) Switch

if statement

- Evaluates a condition
- Controls based on the condition result
- Either true or false
- Four types:
    1) if
    2) if-else
    3) else-if
    4) nested if

if statement

- Enables to enter the block of code only if condition evaluates to **true**
- **Syntax:**

```
if(<condition>) {

//block of code

}
```

if statement - Example

```
public class Numerics{
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");
        }
        }
```

if-else statement

- The else block is executed if the condition of the if-block is evaluated as **false**
- **Syntax:**

```
if(<condition>) {
//block of code
}
else{
//block of code
}
```

if-else statement - Example

```
    public class Numerics{
            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");  }
             }
```

else-if statement

- Contains the if-statement followed by multiple else-if statements
- Also define an else statement at the end of the chain
- **Syntax:**

```
if(<condition>) {
 //block of code
 }
 else if(<condition>){
 //block of code
 }
 else{
 //block of code
 }
```

else-if statement - Example

```
    public class Language{
    public static void main(String[] args) {
          String lang = "Java";
          if(lang == "Python") {
                 System.out.println("language is python");
           }else if (lang == "C++") {
                 System.out.println("lang is C++");
           }else if(lang == "Java") {
                 System.out.println("lang is Java");
           }else {
                  System.out.println(lang);  }
           }}
```

Nested-if statement

- if statement contains multiple if-else statements as a separate block of code.
- **Syntax:**

```
if(<condition>) {
    if(<condition>) {
        //block of code
        }  }
else if(<condition>){
//block of code
}
else{
//block of code  }
```

Nested if statement - Example

```
public class Language{
public static void main(String[] args) {
String lang = "Java";
if(lang == "HLL & IL") {
    if(lang == “python”)
          System.out.println("language is python");
   }else if (lang == "C++") {
          System.out.println("lang is C++");
 }else if(lang == "Java") {
          System.out.println("lang is Java");
}else {
          System.out.println(lang);  }
}}
```

Switch statement

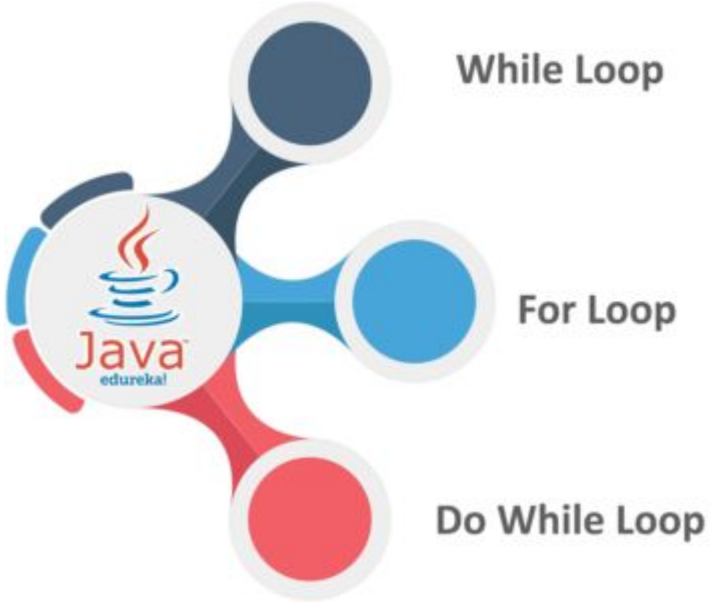
- Checks the variable for the range of values defined for multiple cases
- **Syntax:**

```
switch <variable>
{
Case <option 1>:
//block of statements
Case <option n>:
//block of statements
Default:
//block of statements  }
```

Loops in Java

Executes a set of instructions/functions repeatedly when some conditions become true.
There are three types of loops in Java.

- for loop
- while loop
- do-while loop



For Loops

Iterates a part of the program several times if the number of iteration is fixed.

- Simple For Loop
- Nested for loop
- For-each or Enhanced For Loop
- Labeled For Loop

Simple For Loop

Syntax:

```
for(Initialization; Condition; Increment/Decrement)
{
    Statements;
}
```

For Loop

```
class Main {
 public static void main(String[] args)
{
   for(int i = 0; i < 3; i++) {
     System.out.println(i);
   }
 }
}
```

i = 0	i < 3	o/p	i++
0	true	0	1
1	true	1	2
2	true	2	3
3	False	-	-

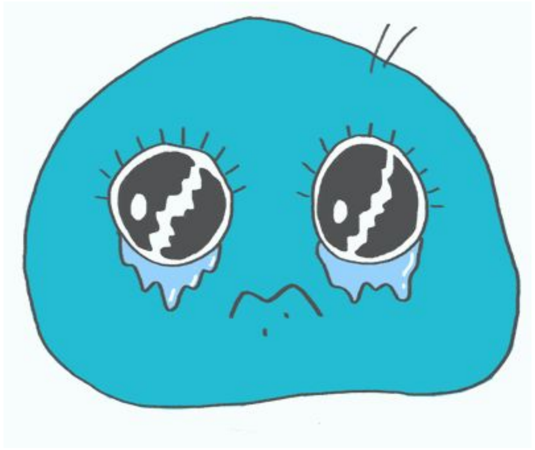
Example

```
class Main {
 public static void main(String[] args) {
   for(int i = 0; i < 3; i++) {
     System.out.println(i);
   }
   System.out.println(i);
 }
}
```

NOT WORKING, WHY ?

Example

```
class Main {
public static void main(String[] args) {
    int i;
    for(int i = 0; i < 3; i++) {
      System.out.println(i);
    }
  }
}
```



NOT WORKING, WHY ?

Nested For Loop

```
public class Main {
  public static void main(String[] args) {
    for(int i = 1;i <= 3;i++) {
      for(int j = 1;j <= 3;j++) {
        System.out.println(i+" "+j);
      }
    }
  }
}
```

Predict the output

```
public class Main {
public static void main(String[] args) {
  int term = 6;
  for(int i = 1;i <= term;i++) {
    for(int j = term;j >= i;j--) {
      System.out.print("* ");
    }
    System.out.println();
  }  }  }
```

Predict the output

```
public class Main {
  public static void main(String[] args) {
    for(int i = 1;i <= 5;i++) {
      for(int j = 1;j <= i;j++) {
        System.out.print("* ");
      }
      System.out.println();
    }
  }
}
```

For Each Loop

```
public class Main {
  public static void main(String[] args) {
    int arr[] = {12,23,44,56,78};
    for(int i:arr) {
      System.out.println(i);
    }
  }
}
```

Jump statements

- Transfer the execution control to the other part of the program
- Two types:
    1) break
    2) continue

break

- breaks the current flow of the program
- transfers the control to the next statement outside the current flow.
- breaks the loop and switch statement - forceful terminations

```
for(int i = 0; i<= 10; i++) {
System.out.println(i);
if(i==6) {
break;
}  }
```

continue

- skips the specific part of the loop
- jumps to the next iteration of the loop immediately
- forceful iterations of the loop

```
for(int i = 0; i<= 2; i++) {
for(int j = i; j<=5; j++) {
if(j == 4) {
continue;
}
System.out.println(j);
}  }  }
```

Infinite For Loop

```
public class Main {
  public static void main(String[] args) {
    for(;;) {
      System.out.println("infinitive loop");
    }
  }
}
```

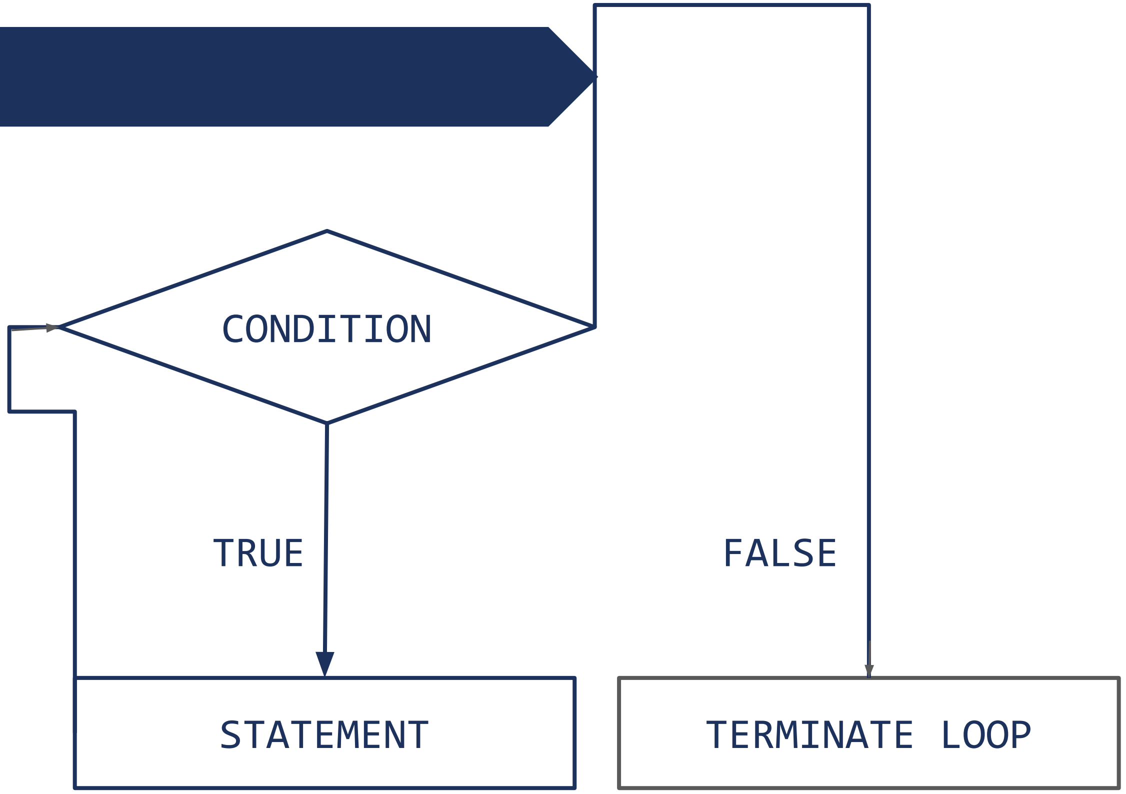
While Loop

- Iterates a part of the program several times.
- The number of iteration is not fixed
- **Syntax:**

```
 while(condition)
{
  //block of code


 }
```

While Loop



Example

```java
public class Main {
  public static void main(String[] args) {
    int i = 1;
    while(i <= 10) {
      System.out.println(i);
      i++;
    }
  }
}
```

Infinite While Loop

If you pass **true** in the while loop, it will be infinitive while loop

```
while(true)
{
//code to be executed
}
```

Infinite While Loop

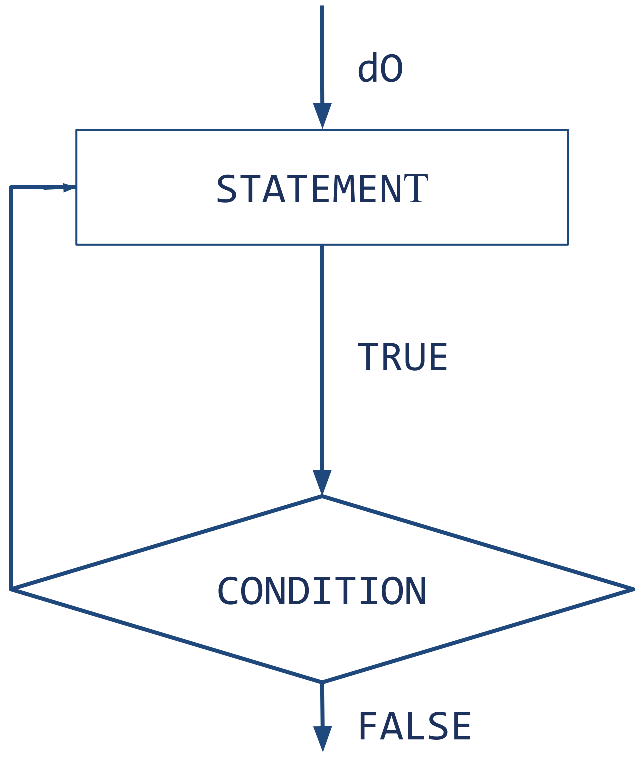
```
public class Main {
  public static void main(String[] args) {
    while(true) {
      System.out.println("infinitive while loop");
    }
  }
}
```

do While Loop

- Iterates a part of the program several times.
- The number of iteration is not fixed and you must have to execute the loop at least once.
- **Syntax:**

```
do
{
//block of code
}while(condition);
```

do While Loop



Example

```
public class Main {
  public static void main(String[] args) {
    int i = 1;
    do{
      System.out.println(i);
      i++;
    }while(i <= 10);
  }
}
```

Infinite do While - Syntax

- If you pass **true** in the do-while loop, it will be infinitive do-while loop

```
do
{
//block of code
}while(true);
```

Infinite do While

```
public class Main {
  public static void main(String[] args) {
    do{
      System.out.println("infinitive do while loop");
    }while(true);
  }
}
```