Mahindra École Centrale
COLLEGE OF ENGINEERING

Object Oriented Programming JAVA

Dr. Prafulla Kalapatapu
Computer Science Engineering
Mahindra Ecole Centrale
prafulla.kalapatapu@mechyd.ac.in

Mahindra
École Centrale
COLLEGE OF ENGINEERING

LANGUAGE FUNDAMENTALS

Prafulla Kalapatapu

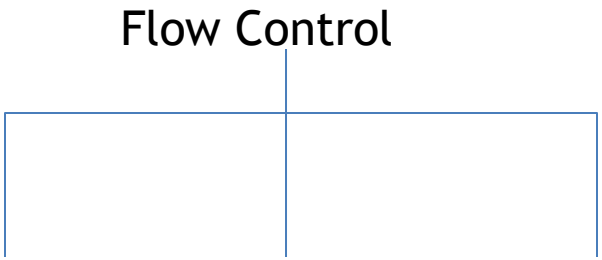
Mahindra École Centrale
COLLEGE OF ENGINEERING

Flow Control

Flow Control

Mahindra École Centrale
COLLEGE OF ENGINEERING

- It describes the order in which statements will be executed at runtime.



Selection Statements
- if-else
- Switch

Iterative Statements
- while
- do-while
- For
- For each loop

Transfer Statements
- break
- continue
- return

Iterative Statements

Mahindra École Centrale
COLLEGE OF ENGINEERING

1. while :

- Here statements execution depends on condition.
- Syntax : while(b)
  {
  }
- The argument to the while loop should be boolean type ,if we are using any other type we will get compile time error.

```
while(1)
{
}
```

CE : incompatible types

Found : int

Required : boolean

- Curly braces are optional
- With out curly braces, we can write only one statement i.e which should not be declarative.

Ex :

a.
```
while(true)
System.out.println("hi");
```

b.
```
while(true);
```

c.
```
while(true)
{
int x=10;
}
```

d.
```
while(true)
int x=10;
```
Error

a.
```
while(true)
{
System.out.println(“hi”);
}
System.out.println(“hello”);
```

Unreachable statement

b.
```
while(false)
{
System.out.println(“hi”);
}
System.out.println(“hello”);
```

Unreachable statement

c.
```
int a=10,b=20;
while(a<b)
{
System.out.println(“hi”);
}
System.out.println(“hello”);
```

No Error

d.
```
final int a=10,b=20;
while(a<b)
{
System.out.println(“hi”);
}
System.out.println(“hello”);
```

Unreachable statement

2. do-while :

- If we want to execute loop body atleast once then we should go for do while loop.
- Syntax : do

  { <statements>

  } while(condition);
- Curly braces are optional
- With out curly braces, we can write only one statement between do and while.

a.
```
do
System.out.println("hi");
while(true);
```

b.
```
do
{
int x=10;
}while(true);
```

c.
```
do;
while(true);
```

d.
```
do
int x=10;
while(true);
```

CE :Statement should not be declarative

a.

```
do
while(true);
```

Error: between do and while, atleast one statement required

b.

```
do while(true)
System.out.println("hello");
while(false);
```

No Error

Interprets as

b.

```
do
while(true)
System.out.println("hello");
while(false);
```

No Error

a.
```
do
{
System.out.println(“hello”);
}while(true);
System.out.println(“hi”);
```
Unreachable statement

b.
```
do
{
System.out.println(“hello”);
}while(false);
System.out.println(“hi”);
```
No Error

c.
```
int a=10,b=20;
do
{
System.out.println(“hello”);
}while(a<b);
System.out.println(“hi”);
```
No Error

d.
```
int a=10,b=20;
do
{
System.out.println(“hello”);
}while(a>b);
System.out.println(“hi”);
```
No Error

e.
```
final int a=10,b=20;
do
{
System.out.println(“hello”);
}while(a>b);
System.out.println(“hi”);
```
No Error

f.
```
final int a=10,b=20;
do
{
System.out.println(“hello”);
}while(a<b);
System.out.println(“hi”);
```
Unreachable statement

3. for :

- Syntax: for(initialization; conditional; updation)
  { }
- Curly braces are optional
- Without curly braces, we can write only one statement i.e not to be declarative.

a. Initialization section.

- This will be executed only once

- usually we are declaring and performing initialization for the variables in this section.

- Here we can declare multiple variables of same type but different data type variables we cant declare.

Ex: int x=0,y=0; valid

int x=0, byte k=2; Not valid

int x=0, int y=0; valid

- In the initialization section, we can take any valid java statements including System.out.println() also.

Ex:
```
int i=0;
for( System.out.println("hello");i<3;i++)
{
    System.out.println("hi");
}
```

o/p:
```
hello
hi
hi
hi
```

b. Conditional expression :

- Here, we can take any java expression but the result should be boolean type.
- It is optional and if we are not specifying then compiler will always place "true".

c. Updation :

- We can take any valid java statement including System.out.println() also.

Ex:
```
int i=0;
for( System.out.println("hello"); i<3; System.out.println("i"))
{
    i++;
}
```

All 3 parts of for loop are optional.

Ex:
```
for( ; ; )
```
represents infinite loop.

a.

```
for(int i=0;true;i++)
{
System.out.println("hello");
}
System.out.println("hi");
```

Unreachable statement

b.

```
for(int i=0;false;i++)
{
System.out.println("hello");
}
System.out.println("hi");
```

Unreachable statement

c.

```
for(int i=0; ;i++)
{
System.out.println("hello");
}
System.out.println("hi");
```

Unreachable statement

d.

```
int a=10,b=20;
for(int i=0;a<b ;i++)
{
System.out.println("hello");
}
System.out.println("hi");
```

No error

e.

```
final int a=10,b=20;
for(int i=0;a<b ;i++)
{
System.out.println("hello");
}
System.out.println("hi");
```

Unreachable statement

Mahindra École Centrale
COLLEGE OF ENGINEERING

4. For each loop : (Enhanced for loop)

- Introduced in 1.5v
- This is the most convenient loop to retrieve the elements from arrays or collections.

Ex: print elements of single dimensional array by using general and for each loop.

int a[] = { 2,3,4,5,6};

General for loop	For each loop

for(int i=0;i<a.length;i++)
{
System.out.println(a[i]);
}
	for(int y:a)
	{
	System.out.println(y);
	}

o/p: 2 3 4 5 6	o/p: 2 3 4 5 6

Ex: print elements of two dimensional array by using general and for each loop.

int a[][] = {{10,20,30},{40,50}};

General for loop	For each loop

for(int i=0;i<a.length;i++)
{
for(int j=0;j<a[i].length;j++)
System.out.println(a[i][j]);
}
	for(int[] y:a)
	{
	for(int x:y)
	System.out.println(x);
	}

o/p: 10 20 30 40 50	o/p: 10 20 30 40 50

For each loop limitation.

- It is not a general purpose loop
- It is applicable only for arrays and collections
- By using for each loop, we should retrieve all values of arrays and collections, cant be used to retrieve a particular set of values.

Transfer Statements

Mahindra École Centrale
COLLEGE OF ENGINEERING

1. break:

We can use break statement in the following cases

- Within the switch to stop fall through
- Inside loops to break the loop execution
- Inside label blocks to break that block execution.

a.

```
switch(a)
{
case 1:
System.out.println("hello");
break;
}
```

b.

```
for(int i=0;i<10;i++)
{
if(i==5)
break;
}
```

c.

```
l1:
{
System.out.println("H");
break l1;
System.out.println("I");
}
System.out.println("E");
```

o/p: H
E

- If we are using break outside of loop/switch, we will get compile time error.

2. continue :

- We can use continue statement to skip current iteration and continue for the next iteration inside loops.

Ex:

```
for(int i=0;i<=10;i++)
{
        if( i%2==0)
        continue;
        System.out.println(i);
}
```

If we are using continue outside of loops, we will get compile time error

Note: compiler will check for unreachable statements only in the case of loops but not in “if-else” or “simple if”.

3. return :

- It returns from current executing method to called method.
- Syntax: return variable/literal value;

Ex: return x;

or

return 10;

or

return ; // is valid for void return type methods.