CORE JAVA

With

SCJP / OCJP

Study Material

Chapter 3 : Flow Control

DURGA M.Tech

(Sun certified & Realtime Expert)

Ex. IBM Employee

Trained Lakhs of Students
for last 14 years across INDIA

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Flow Control

Agenda :

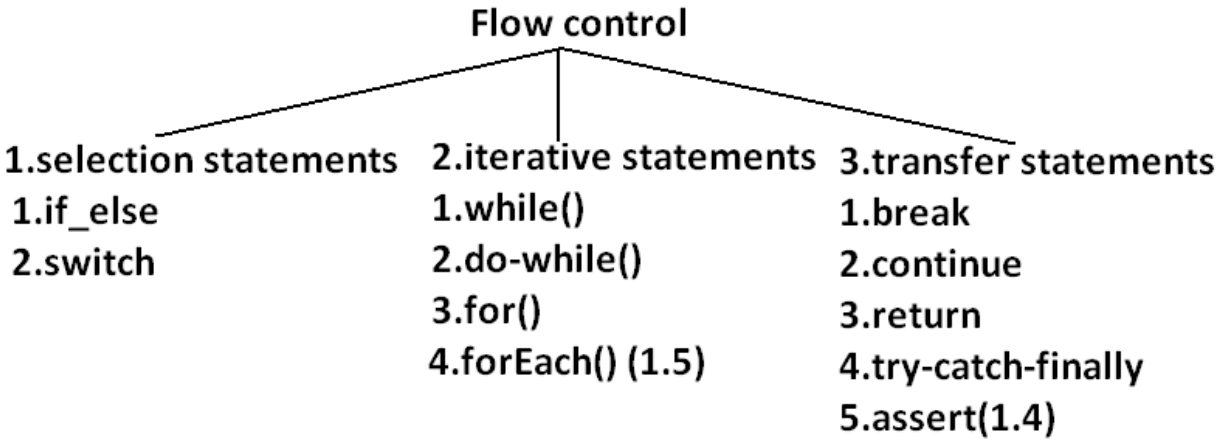
1. **Introduction**
2. **Selection statements**
   - i. **if-else**
   - ii. **Switch**
     - **Case Summary**
     - **fall-through inside a switch**
     - **default case**
3. **Iterative Statements**
   - i. **While loop**
     - **Unreachable statement in while**
   - ii. **Do-while**
     - **Unreachable statement in do while**
   - iii. **For Loop**
     - **Initilizationsection**
     - **Conditional check**
     - **Increment and decrement section**
     - **Unreachable statement in for loop**
   - iv. **For each**
     - **Iterator Vs Iterable(1.5v)**
     - **Difference between Iterable and Iterator**
4. **Transfer statements**
   - **Break statement**
   - **Continue statement**
   - **Labeled break and continue statements**
   - **Do-while vs continue (The most dangerous combination)**

JAVA Means DURGASOFT

Introduction :

Flow control describes the order in which all the statements will be executed at run time.

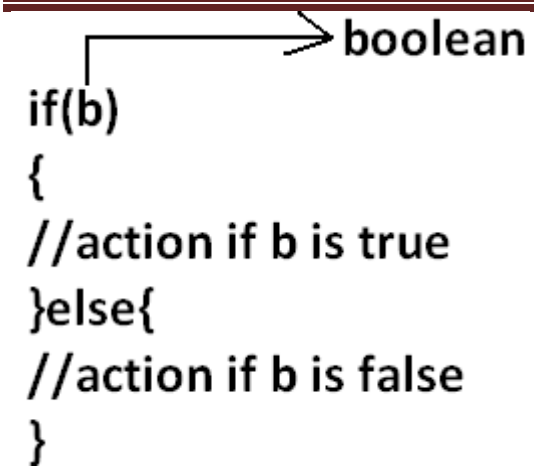
Diagram:



Selection statements:

if-else:

syntax:



The argument to the if statement should be Boolean by mistake if we are providing any other type we will get "compile time error".

Example 1:

```
 public class ExampleIf{
public static void main(String args[]){
int x=0;
if(x)
{
System.out.println("hello");
}else{
System.out.println("hi");
}}}
OUTPUT:
Compile time error:
D:\Java>javac ExampleIf.java
ExampleIf.java:4: incompatible types
found   : int
required: boolean
if(x)
```

Example 2:

```
public class ExampleIf{
public static void main(String args[]){
int x=10;
if(x=20)
{
System.out.println("hello");
}
else{
System.out.println("hi");
```

```
}}}
OUTPUT:
Compile time error
D:\Java>javac ExampleIf.java
ExampleIf.java:4: incompatible types
found   : int
required: boolean
if(x=20)
```

Example 3:

```
public class ExampleIf{
public static void main(String args[]){
int x=10;
if(x==20)
{
System.out.println("hello");
}else{
System.out.println("hi");
}}}
OUTPUT:
Hi
```

Example 4:

```
public class ExampleIf{
public static void main(String args[]){
boolean b=false;
if(b=true)
{
System.out.println("hello");
}else{
System.out.println("hi");
}}}
OUTPUT:
Hello
```

Example 5:

```
public class ExampleIf{
public static void main(String args[]){
boolean b=false;
if(b==true)
{
System.out.println("hello");
}else{
System.out.println("hi");
}}}
OUTPUT:
Hi
```

Both else part and curly braces are optional.
Without curly braces we can take only one statement under if, but it should not be declarative statement.

Example 6:

```
 public class ExampleIf{
public static void main(String args[]){
if(true)
System.out.println("hello");
}}
OUTPUT:
Hello
```

JAVA Means DURGASOFT

Example 7:

```
public class ExampleIf{
public static void main(String args[]){
if(true);
}}
OUTPUT:
No output
```

Example 8:

```
public class ExampleIf{
public static void main(String args[]){
if(true)
int x=10;
}}
OUTPUT:
Compile time error
D:\Java>javac ExampleIf.java
ExampleIf.java:4: '.class' expected
int x=10;
ExampleIf.java:4: not a statement
int x=10;
```

Example 9:

```
public class ExampleIf{
public static void main(String args[]){
if(true){
int x=10;
}}}
OUTPUT:
D:\Java>javac ExampleIf.java
D:\Java>java ExampleIf
```

LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...
JAVA MEANS DURGASOFT
INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE
AN ISO 9001:2008 CERTIFIED DURGA SOFTWARE SOLUTIONS
#202 2nd FLOOR
www.durgasoft.com
040-64512786
+91 9246212143
+91 8096969696

Example 10:

```
public class ExampleIf{
public static void main(String args[]){
if(true)
System.out.println("hello"); ——————> dependent statement on if
System.out.println("hi"); ——————> this is independent statement on if
}
}
```

```
OUTPUT:
Hello
Hi
```

Semicolon(;) is a valid java statement which is call empty statement and it won't produce any output.

Switch:

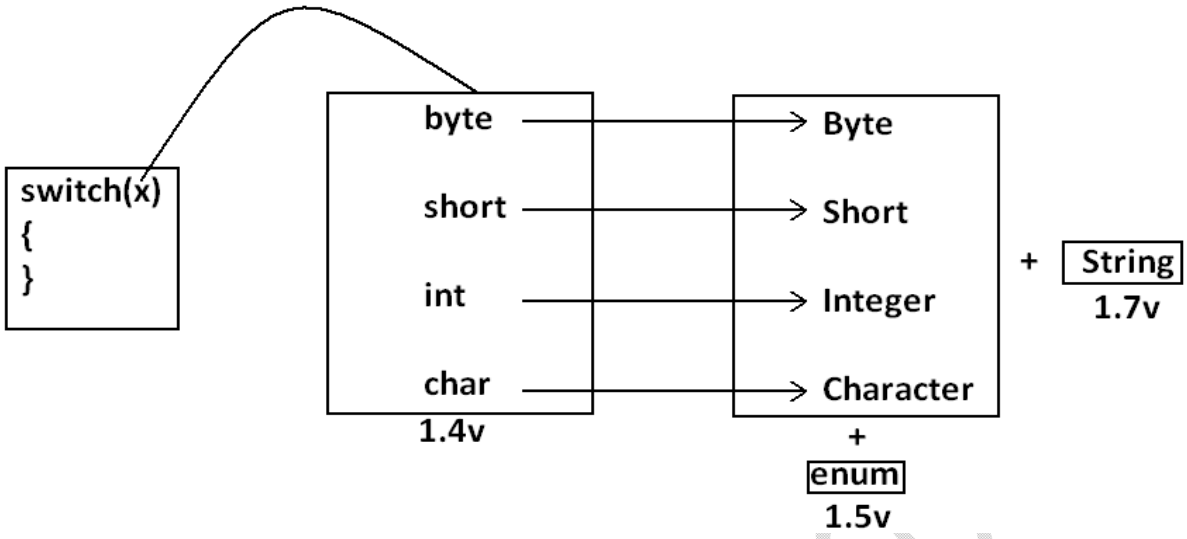
If several options are available then it is not recommended to use if-else we should go for switch statement.Because it improves readability of the code.

Syntax:

```
switch(x)
{
case 1:
action1
case 2:
action2
.
.
.
default:
default action
}
```

Until 1.4 version the allow types for the switch argument are byte, short, char, int but from 1.5 version on wards the corresponding wrapper classes (Byte, Short, Character, Integer) and "enum" types also allowed.

Diagram:



- **Curly braces are mandatory.(except switch case in all remaining cases curly braces are optional)**
- **Both case and default are optional.**
- **Every statement inside switch must be under some case (or) default. Independent statements are not allowed.**

Example 1:

```
public class ExampleSwitch{
public static void main(String args[]){
int x=10;
switch(x)
{
System.out.println("hello");
}}}
OUTPUT:
Compile time error.
D:\Java>javac ExampleSwitch.java
ExampleSwitch.java:5: case, default, or '}' expected
System.out.println("hello");
```

Every case label should be "compile time constant" otherwise we will get compile time error.

Example 2:

```
public class ExampleSwitch{
public static void main(String args[]){
int x=10;
int y=20;
switch(x)
{
case 10:
System.out.println("10");
case y:
System.out.println("20");
```

```
}}}
OUTPUT:
Compile time error
D:\Java>javac ExampleSwitch.java
ExampleSwitch.java:9: constant expression required
case y:
```

If we declare y as final we won't get any compile time error.

Example 3:

```
public class ExampleSwitch{
public static void main(String args[]){
int x=10;
final int y=20;
switch(x)
{
case 10:
System.out.println("10");
case y:
System.out.println("20");
}}}
OUTPUT:
10
20
```

But switch argument and case label can be expressions , but case label should be constant expression.

Example 4:

```
public class ExampleSwitch{
public static void main(String args[]){
int x=10;
switch(x+1)
{
case 10:
case 10+20:
case 10+20+30:
}}}
OUTPUT:
No output.
```

Every case label should be within the range of switch argument type.

Example 5:

```
public class ExampleSwitch{
public static void main(String args[]){
byte b=10;
switch(b)
{
case 10:
System.out.println("10");
case 100:
System.out.println("100");




case 1000:
System.out.println("1000");
}}}
OUTPUT:
Compile time error
D:\Java>javac ExampleSwitch.java
ExampleSwitch.java:10: possible loss of precision
found   : int
required: byte
case 1000:
```

Example :

```
public class ExampleSwitch{
public static void main(String args[]){
byte b=10;
switch(b+1)
{
case 10:
System.out.println("10");
case 100:
System.out.println("100");
case 1000:
System.out.println("1000");
}}}
OUTPUT:
```

Duplicate case labels are not allowed.

Example 6:

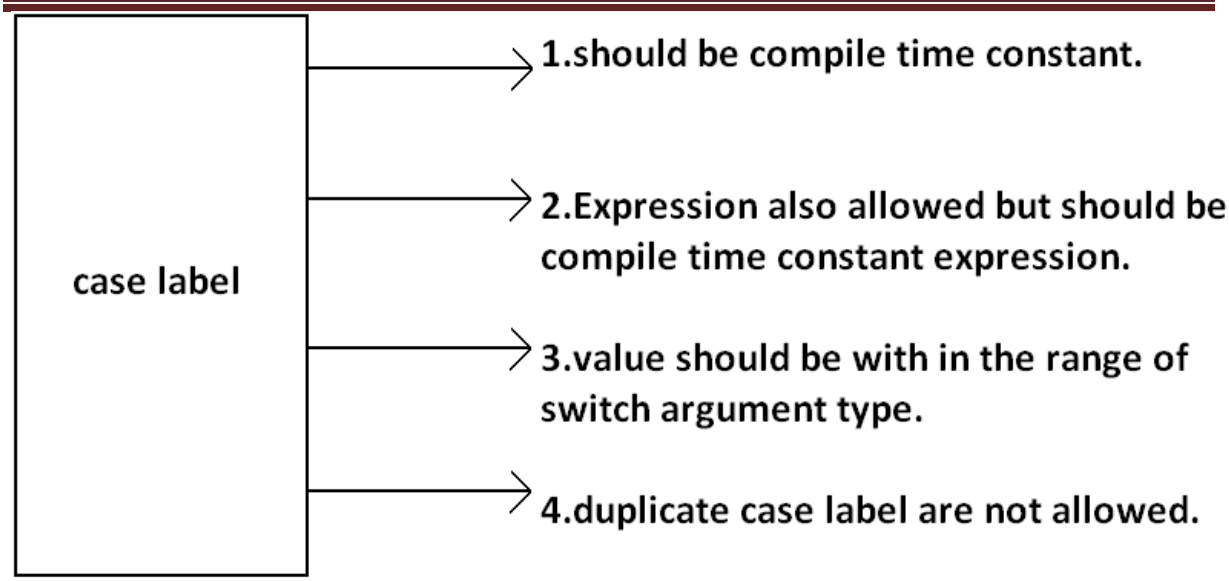
```
public class ExampleSwitch{
public static void main(String args[]){
int x=10;
switch(x)
{
case 97:
System.out.println("97");
case 99:
System.out.println("99");
case 'a':
System.out.println("100");
}}}
OUTPUT:
Compile time error.
D:\Java>javac ExampleSwitch.java
ExampleSwitch.java:10: duplicate case label
```

JAVA Means DURGASOFT

```
case 'a':
```

CASE SUMMARY:

Diagram:



FALL-THROUGH INSIDE THE SWITCH:

With in the switch statement if any case is matched from that case onwards all statements will be executed until end of the switch (or) break. This is call "fall-through" inside the switch .

The main advantage of fall-through inside a switch is we can define common action for multiple cases

Example 7:

```
public class ExampleSwitch{
public static void main(String args[]){
int x=0;
switch(x)
{
case 0:
System.out.println("0");
case 1:
System.out.println("1");
break;
case 2:
System.out.println("2");
default:
System.out.println("default");
}}}
```

```
OUTPUT:
x=0             x=1             x=2             x=3
0               1               2               default
1                               default
```

DEFAULT CASE:

- **With in the switch we can take the default only once**
- **If no other case matched then only default case will be executed**
- **With in the switch we can take the default any where, but it is convension to take default as last case.**

Example 8:

```
public class ExampleSwitch{
public static void main(String args[]){
int x=0;
switch(x)
{
default:
System.out.println("default");
case 0:
System.out.println("0");
break;
case 1:
System.out.println("1");
case 2:
System.out.println("2");
}}}
```

```
OUTPUT:
X=0             x=1             x=2             x=3
0               1               2               default
                2                               0
```

ITERATIVE STATEMENTS:

JAVA Means DURGASOFT

While loop:

if we don't know the no of iterations in advance then best loop is while loop:

Example 1:

```
while(rs.next())
{
}
```

Example 2:

```
while(e.hasMoreElements())
{
----------
----------
----------
}
```

Example 3:

```
while(itr.hasNext())
{
----------
----------
----------
}
```

The argument to the while statement should be Boolean type. If we are using any other type we will get compile time error.

Example 1:

```
public class ExampleWhile{
public static void main(String args[]){
while(1)
{
System.out.println("hello");
}}}
OUTPUT:
Compile time error.
D:\Java>javac ExampleWhile.java
ExampleWhile.java:3: incompatible types
found   : int
required: boolean
while(1)
```

Curly braces are optional and without curly braces we can take only one statement which should not be declarative statement.

Example 2:

```
public class ExampleWhile{
public static void main(String args[]){
while(true)
System.out.println("hello");
}}
OUTPUT:
Hello (infinite times).
```

Example 3:

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

JAVA Means DURGASOFT

```
while(true);
}}
OUTPUT:
No output.
```

Example 4:

```
public class ExampleWhile{
public static void main(String args[]){
while(true)
int x=10;
}}
OUTPUT:
Compile time error.
D:\Java>javac ExampleWhile.java
ExampleWhile.java:4: '.class' expected
int x=10;
ExampleWhile.java:4: not a statement
int x=10;
```

Example 5:

```
public class ExampleWhile{
public static void main(String args[]){
while(true)
{
int x=10;
}}}
OUTPUT:
No output.
```

Unreachable statement in while:

Example 6:

```
public class ExampleWhile{
public static void main(String args[]){
while(true)
{
System.out.println("hello");
}
System.out.println("hi");
}}
OUTPUT:
Compile time error.
D:\Java>javac ExampleWhile.java
```

```
ExampleWhile.java:7: unreachable statement
System.out.println("hi");
```

Example 7:

```
public class ExampleWhile{
public static void main(String args[]){
while(false)
{
System.out.println("hello");
}
System.out.println("hi");
}}
OUTPUT:
D:\Java>javac ExampleWhile.java
ExampleWhile.java:4: unreachable statement
{
```

Example 8:

```
public class ExampleWhile{
public static void main(String args[]){
int a=10,b=20;
while(a<b)
{
System.out.println("hello");
}
System.out.println("hi");
}}
OUTPUT:
Hello (infinite times).
```

Example 9:

```
public class ExampleWhile{
public static void main(String args[]){
final int a=10,b=20;
while(a<b)
{
System.out.println("hello");
}
System.out.println("hi");
}}
OUTPUT:
Compile time error.
D:\Java>javac ExampleWhile.java
ExampleWhile.java:8: unreachable statement
System.out.println("hi");
```

Example 10:

```
public class ExampleWhile{
public static void main(String args[]){
final int a=10;
while(a<20)
{
System.out.println("hello");
}
System.out.println("hi");
}}
OUTPUT:
D:\Java>javac ExampleWhile.java
ExampleWhile.java:8: unreachable statement
System.out.println("hi");
```

Note:

JAVA Means DURGASOFT

- **Every final variable will be replaced with the corresponding value by compiler.**
- **If any operation involves only constants then compiler is responsible to perform that operation.**
- **If any operation involves at least one variable compiler won't perform that operation. At runtime jvm is responsible to perform that operation.**

Example 11:

```
public class ExampleWhile{
public static void main(String args[]){
int a=10;
while(a<20)
{
System.out.println("hello");
}
System.out.println("hi");
}}
OUTPUT:
Hello (infinite times).
```

Do-while:

If we want to execute loop body at least once then we should go for do-while.

Syntax:

```
do
{
----------
----------
----------
}while(b); ——→semicolon is the mandatory.
```

Curly braces are optional.
Without curly braces we can take only one statement between do and while and it should not be declarative statement.

Example 1:

```
public class ExampleDoWhile{
public static void main(String args[]){
do
System.out.println("hello");
while(true);
}}
Output:
Hello (infinite times).
```

Example 2:

```
public class ExampleDoWhile{
public static void main(String args[]){
do;
while(true);
}}
Output:
Compile successful.
```

Example 3:

```
public class ExampleDoWhile{
public static void main(String args[]){
do
int x=10;
while(true);
}}
Output:
D:\Java>javac ExampleDoWhile.java
ExampleDoWhile.java:4: '.class' expected
int x=10;
ExampleDoWhile.java:4: not a statement
int x=10;
ExampleDoWhile.java:4: ')' expected
int x=10;
```

Example 4:

```
public class ExampleDoWhile{
public static void main(String args[]){
do
{
int x=10;
}while(true);
}}
Output:
Compile successful.
```

Example 5:

```
public class ExampleDoWhile{
public static void main(String args[]){
do while(true)
System.out.println("hello");
while(true);
}}
Output:
Hello (infinite times).
```

Rearrange the above Example:

```
public class ExampleDoWhile{
```

JAVA Means DURGASOFT

```
public static void main(String args[]){
do
  while(true)
       System.out.println("hello");
while(true);
}}
Output:
Hello (infinite times).
```

Example 6:

```
public class ExampleDoWhile{
public static void main(String args[]){
do
while(true);
}}
Output:
Compile time error.
D:\Java>javac ExampleDoWhile.java
ExampleDoWhile.java:4: while expected
while(true);
ExampleDoWhile.java:5: illegal start of expression
}
```

Unreachable statement in do while:

Example 7:

```
public class ExampleDoWhile{
public static void main(String args[]){
do
{
System.out.println("hello");
}
while(true);
System.out.println("hi");
}}
Output:
Compile time error.
D:\Java>javac ExampleDoWhile.java
ExampleDoWhile.java:8: unreachable statement
System.out.println("hi");
```

Example 8:

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

```
do
{
System.out.println("hello");
}
while(false);
System.out.println("hi");
}}
Output:
Hello
Hi
```

Example 9:

```
public class ExampleDoWhile{
public static void main(String args[]){
int a=10,b=20;
do
{
System.out.println("hello");
}
while(a<b);
System.out.println("hi");
}}
Output:
Hello (infinite times).
```

Example 10:

```
public class ExampleDoWhile{
public static void main(String args[]){
int a=10,b=20;
do
{
System.out.println("hello");
}
while(a>b);
System.out.println("hi");
}}
Output:
Hello
Hi
```

Example 11:

```
public class ExampleDoWhile{
public static void main(String args[]){
final int a=10,b=20;
do
{
System.out.println("hello");
}
while(a<b);
System.out.println("hi");
}}
Output:
Compile time error.
D:\Java>javac ExampleDoWhile.java
ExampleDoWhile.java:9: unreachable statement
System.out.println("hi");
```

Example 12:

```
public class ExampleDoWhile{
public static void main(String args[]){
final int a=10,b=20;
do
{
```

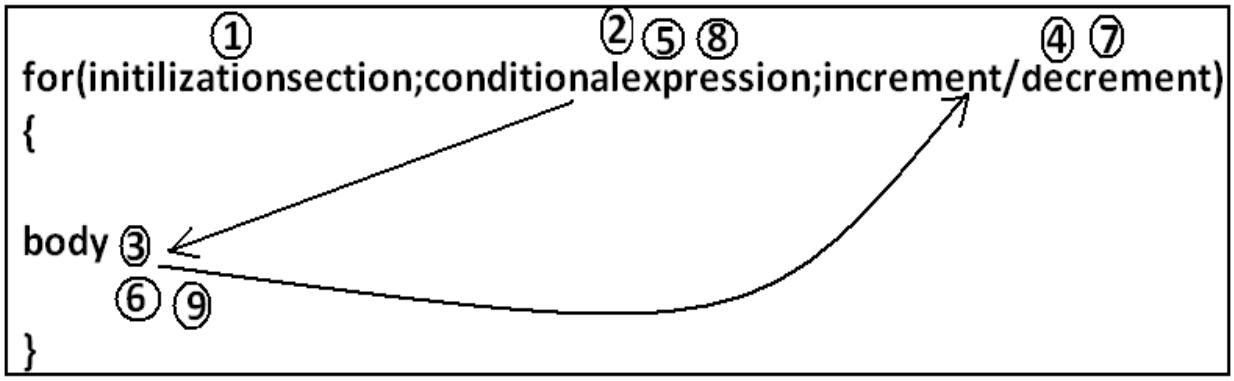
JAVA Means DURGASOFT

```
System.out.println("hello");
}
while(a>b);
System.out.println("hi");
}}
Output:
D:\Java>javac ExampleDoWhile.java
D:\Java>java ExampleDoWhile
Hello
Hi
```

For Loop:

This is the most commonly used loop and best suitable if we know the no of iterations in advance.

Syntax:



Curly braces are optional and without curly braces we can take only one statement which should not be declarative statement.

Initilizationsection:

This section will be executed only once.
Here usually we can declare loop variables and we will perform initialization.
We can declare multiple variables but should be of the same type and we can't declare different type of variables.

Example:

```
Int i=0,j=0; valid
Int i=0,Boolean b=true; invalid
Int i=0,int j=0; invalid
```

In initialization section we can take any valid java statement including "s.o.p" also.

Example 1:

```
public class ExampleFor{
public static void main(String args[]){
int i=0;
for(System.out.println("hello u r sleeping");i<3;i++){
System.out.println("no boss, u only sleeping");
}}}
Output:
D:\Java>javac ExampleFor.java
D:\Java>java ExampleFor
Hello u r sleeping
No boss, u only sleeping
No boss, u only sleeping
No boss, u only sleeping
```

Conditional check:

We can take any java expression but should be of the type Boolean.
Conditional expression is optional and if we are not taking any expression compiler will place true.

Increment and decrement section:

Here we can take any java statement including s.o.p also.

Example:

```
public class ExampleFor{
public static void main(String args[]){
int i=0;
for(System.out.println("hello");i<3;System.out.println("hi")){
i++;
}}}
Output:
D:\Java>javac ExampleFor.java
D:\Java>java ExampleFor
Hello
Hi
Hi
Hi
```

All 3 parts of for loop are independent of each other and all optional.

Example:

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

```
Output:
Hello (infinite times).
```

Curly braces are optional and without curly braces we can take exactly one statement and it should not be declarative statement.

Unreachable statement in for loop:

Example 1:

```
public class ExampleFor{
public static void main(String args[]){
for(int i=0;true;i++){
System.out.println("hello");
}
System.out.println("hi");
}}
Output:
Compile time error.
D:\Java>javac ExampleFor.java
ExampleFor.java:6: unreachable statement
System.out.println("hi");
```

Example 2:

```
public class ExampleFor{
public static void main(String args[]){
for(int i=0;false;i++){
System.out.println("hello");
}
System.out.println("hi");
}}
Output:
Compile time error.
D:\Java>javac ExampleFor.java
ExampleFor.java:3: unreachable statement
for(int i=0;false;i++){
```

Example 3:

```
public class ExampleFor{
public static void main(String args[]){
for(int i=0;;i++){
System.out.println("hello");
}
System.out.println("hi");
}}
Output:
Compile time error.
D:\Java>javac ExampleFor.java
ExampleFor.java:6: unreachable statement
System.out.println("hi");
```

Example 4:

```
public class ExampleFor{
public static void main(String args[]){
int a=10,b=20;
for(int i=0;a<b;i++){
System.out.println("hello");
}
System.out.println("hi");
}}
Output:
```

JAVA Means DURGASOFT

```
Hello (infinite times).
```

Example 5:

```
public class ExampleFor{
public static void main(String args[]){
final int a=10,b=20;
for(int i=0;a<b;i++){
System.out.println("hello");
}
System.out.println("hi");
}}
Output:
D:\Java>javac ExampleFor.java
ExampleFor.java:7: unreachable statement
System.out.println("hi");
```

For each:(Enhanced for loop)

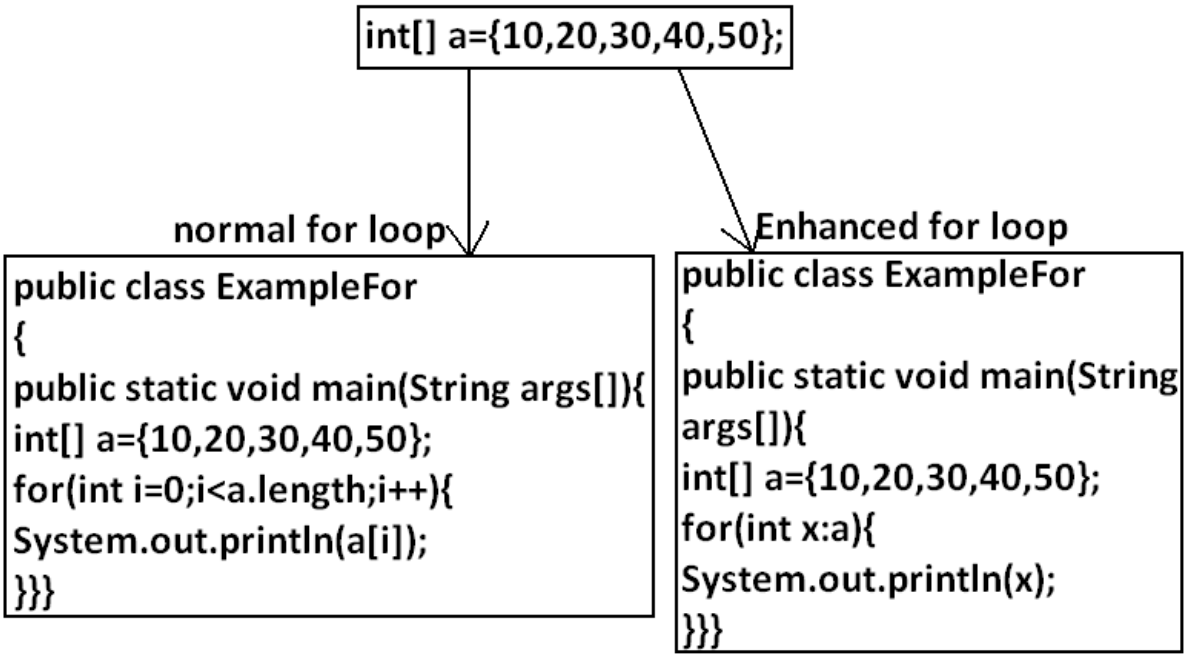
- **For each Introduced in 1.5version.**
- **Best suitable to retrieve the elements of arrays and collections.**

Example 1:

Write code to print the elements of single dimensional array by normal for loop and enhanced for loop.

Example:

JAVA Means DURGASOFT

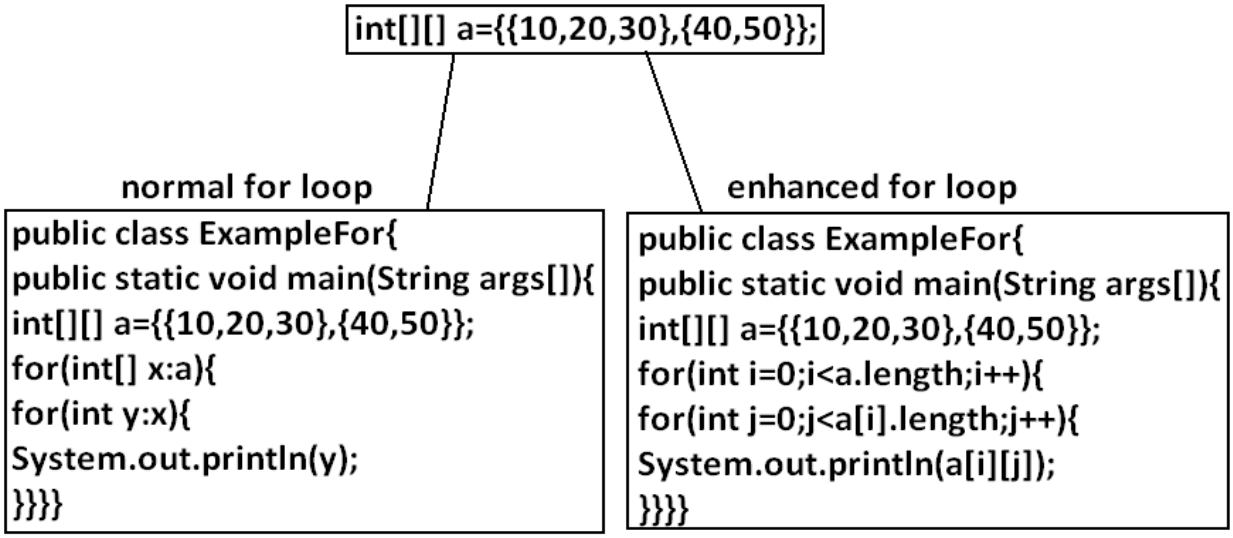



```
Output:
D:\Java>javac ExampleFor.java
D:\Java>java ExampleFor
10
20
30
40
50
```

Example 2:

Write code to print the elements of 2 dimensional arrays by using normal for loop and enhanced for loop.

JAVA Means DURGA SIR



Example 3:

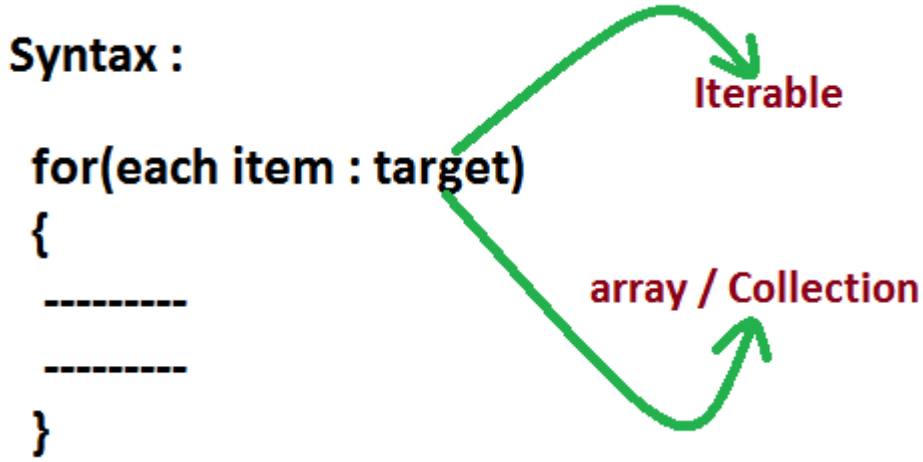
Write equivalent code by For Each loop for the following for loop.

```
public class ExampleFor{
public static void main(String args[]){
for(int i=0;i<10;i++)
{
System.out.println("hello");
}}}
```

```
Output:
D:\Java>javac ExampleFor1.java
D:\Java>java ExampleFor1
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

- **We can't write equivalent for each loop.**
- **For each loop is the more convenient loop to retrieve the elements of arrays and collections, but its main limitation is it is not a general purpose loop.**
- **By using normal for loop we can print elements either from left to right or from right to left. But using for-each loop we can always print array elements only from left to right.**

Iterator Vs Iterable(1.5v)



- **The target element in for-each loop should be Iterable object.**
- **An object is set to be iterable iff corresponding class implements java.lang.Iterable interface.**
- **Iterable interface introduced in 1.5 version and it's contains only one method iterator().**

Syntax : public Iterator iterator();

Every array class and Collection interface already implements Iterable interface.

Difference between Iterable and Iterator:

Iterable	Iterator
It is related to forEach loop	It is related to Collection
The target element in forEach loop should be Iterable	We can use Iterator to get objects one by one from the collection
Iterator present in java.lang package	Iterator present in java.util package
contains only one method iterator()	contains 3 methods hasNext(), next(), remove()
Introduced in 1.5 version	Introduced in 1.2 version

Transfer statements:

Break statement:

We can use break statement in the following cases.

- **Inside switch to stop fall-through.**
- **Inside loops to break the loop based on some condition.**
- **Inside label blocks to break block execution based on some condition.**

Inside switch :

We can use break statement inside switch to stop fall-through

Example 1:

```
class Test{
public static void main(String args[]){
int x=0;
switch(x)
{
case 0:
    System.out.println("hello");
     break ;
case 1:
    System.out.println("hi");
}

Output:
D:\Java>javac Test.java
D:\Java>java Test
Hello
```

Inside loops :

We can use break statement inside loops to break the loop based on some condition.

Example 2:

```
class Test{
public static void main(String args[]){
for(int i=0; i<10; i++) {
 if(i==5)
  break;
  System.out.println(i);
}
}}
Output:
D:\Java>javac Test.java
D:\Java>java Test
0
1
2
3
4
```

Inside Labeled block :

We can use break statement inside label blocks to break block execution based on some condition.

Example:

```
class Test{
public static void main(String args[]){
int x=10;
l1 : {
  System.out.println("begin");
  if(x==10)
  break l1;
  System.out.println("end");
```

JAVA Means DURGASOFT

```
 }
 System.out.println("hello");
}
}
Output:
D:\Java>javac Test.java
D:\Java>java Test
begin
hello
```

These are the only places where we can use break statement. If we are using anywhere else we will get compile time error.

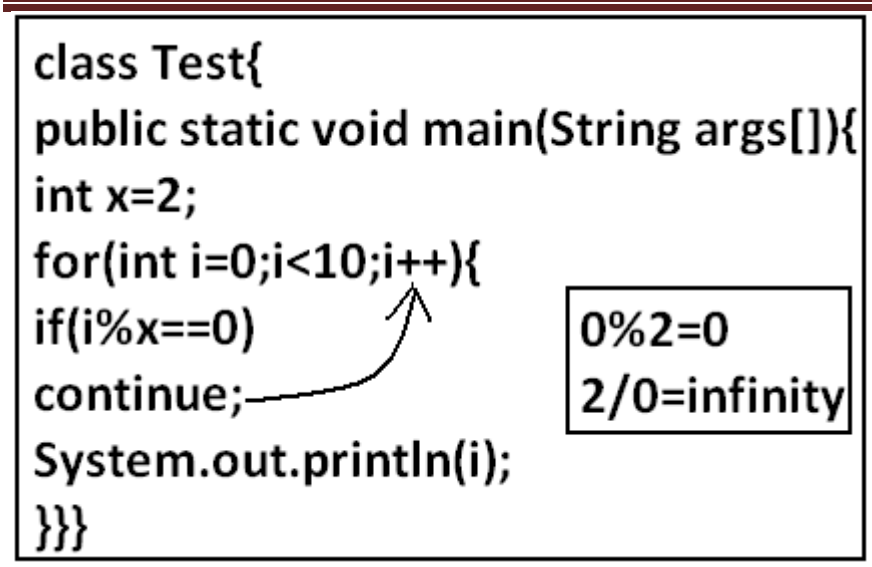
Example:

```
class Test{
public static void main(String args[]){
int x=10;
 if(x==10)
 break;
 System.out.println("hello");
}
}
Output:
Compile time error.
D:\Java>javac Test.java
Test.java:5: break outside switch or loop
break;
```

Continue statement:

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

Example:




```
Output:
D:\Java>javac Test.java
D:\Java>java Test
1
3
5
7
9
```

We can use continue only inside loops if we are using anywhere else we will get compile time error saying "continue outside of loop".

Example:

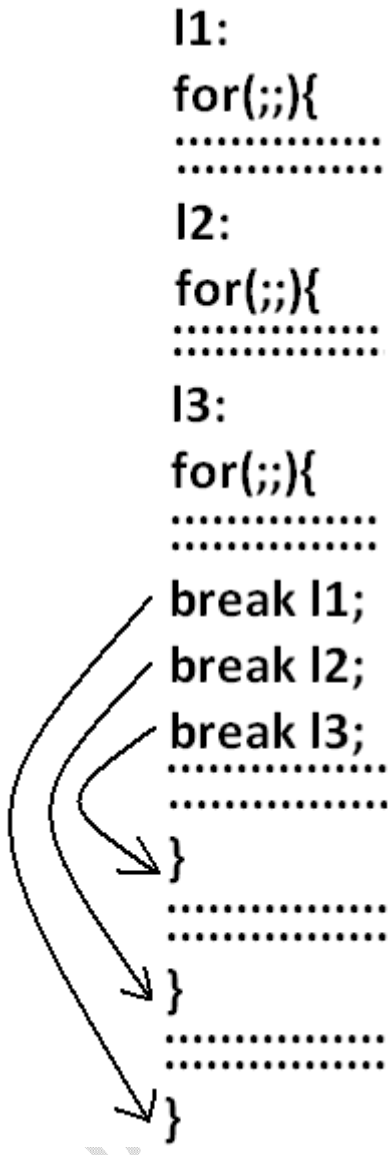
```
class Test
{
public static void main(String args[]){
int x=10;
if(x==10);
continue;
System.out.println("hello");
}
}
Output:
Compile time error.
D:\Enum>javac Test.java
Test.java:6: continue outside of loop
continue;
```

Labeled break and continue statements:

In the nested loops to break (or) continue a particular loop we should go for labeled break and continue statements.

JAVA Means DURGASOFT

Syntax:



J...Means DURGASOFT

Example:

```
class Test
{
public static void main(String args[]){
l1:
 for(int i=0;i<3;i++)
 {
   for(int j=0;j<3;j++)
   {
     if(i==j)
       break;
         System.out.println(i+".........."+j);
    }


 }
 }
}
```

```
Break:
1.........0
2.........0
2.........1

Break l1:
No output.

Continue:
0.........1
0.........2
1.........0
1.........2
2.........0
2.........1

Continue l1:
1.........0
2.........0
2.........1
```

JAVA Means DURGASOFT

Do-while vs continue (The most dangerous combination):

```
class Test
{
public static void main(String args[]){
int x=0;
do
{
++x;
System.out.println(x);
if(++x<5)
continue;
++x;
System.out.println(x);
}while(++x<10);
}
}
```

```
Output:
1
4
6
8
10
```

Compiler won't check unreachability in the case of if-else it will check only in loops.

Example 1:

```
class Test
{
public static void main(String args[]){
while(true)
{
System.out.println("hello");
}
System.out.println("hi");
}
}
Output:
Compile time error.
D:\Enum>javac Test.java
Test.java:8: unreachable statement
System.out.println("hi");
```

Example 2:

```
class Test
{
public static void main(String args[]){
if(true)
{
System.out.println("hello");
}
else
{
System.out.println("hi");
}}}
Output:
Hello
```

JAVA Means DURGASOFT

CORE JAVA with
OCJP/SCJP
JAVA CERTIFICATION

Mr. DURGA M.Tech
JAVA EXPERT
Trained Thousands of Students

SATISFACTION 100% GUARANTEED

One to One
VIDEO CLASSES
EVERYTHING AT YOUR CONVENIENCE
At your convenient Time
With in your convenient duration

AN ISO 9001:2008 CERTIFIED
DURGA
Software Solutions®
www.durgasoft.com

202, 2nd Floor, HUDA Maitrivanam,
Ameerpet, Hyd. Ph: 040-64512786,
9246212143, 8096969696

LEARN FROM EXPERTS ...
COMPLETE JAVA
CORE JAVA, ADV. JAVA, ORACLE, STRUTS, HIBERNATE, SPRING, WEB SERVICES,...
COMPLETE .NET
C#.NET, ASP.NET, SQL SERVER, MVC 5 & WCF
TESTING TOOLS
MANUAL + SELENIUM
ORACLE
D2K
MSBI
SHARE POINT
HADOOP
ANDROID
C, C++, DS, UNIX
CRT & APTITUDE TRAINING
AN ISO 9001:2008 CERTIFIED
DURGA
Software Solutions®
202, 2nd Floor, HUDA Maitrivanam,
Ameerpet, Hyd. Ph: 040-64512786,
9246212143, 8096969696
www.durgasoft.com