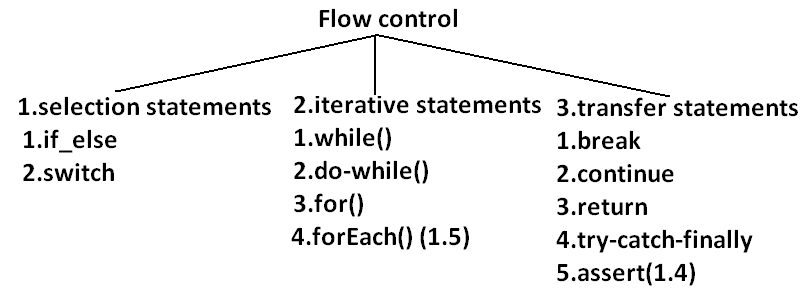
**Flow Control**

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

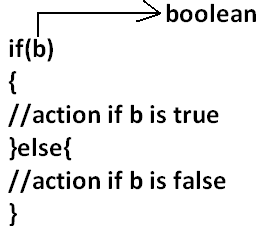
**Diagram:**



**Selection statements:**

1. **if-else:**

**syntax:**



* The argument to the if statement should be Boolean 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** 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

**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;

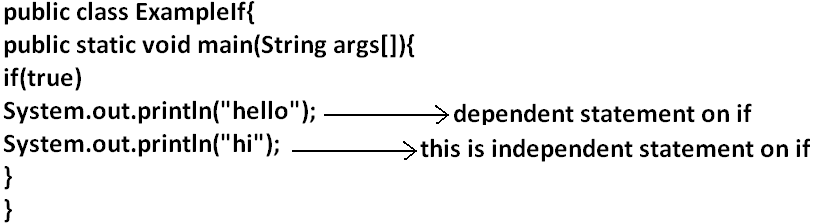
}}}

**OUTPUT:**

D:\Java>javac ExampleIf.java

D:\Java>java ExampleIf

**EXAMPLE 10:**



**OUTPUT:**

Hello

Hi

* Semicolon is a valid java statement which is call empty statement and it won’t produce any output.
* If several options are available then it is not recommended to use **if-else** we should go for **switch statement**.

**Switch:**

**Syntax:**

switch(x)

{

case 1:

action1

case 2:

action2

.

.

.

default:

default action

}

* **Curly braces are mandatory**.
* 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[]){

switch(x)

{

System.out.println("hello");

}}}

**OUTPUT:**

Compile time error.

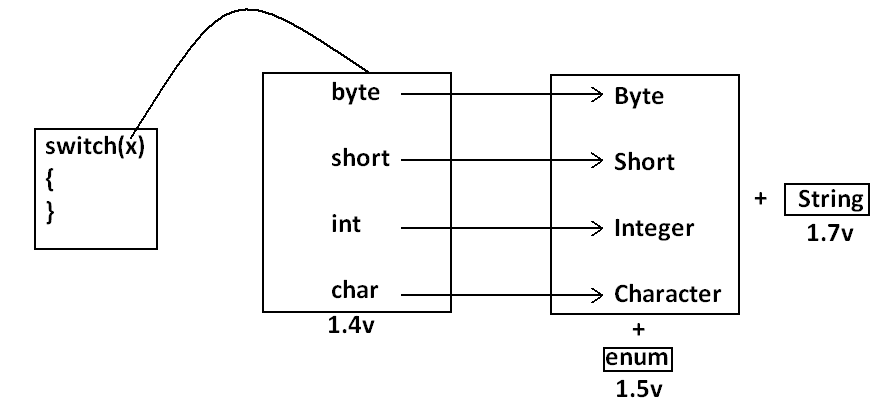
D:\Java>javac ExampleSwitch.java

ExampleSwitch.java:5: case, default, or '}' expected

System.out.println("hello");

* 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 are allowed.

**DIAGRAM:**



* 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

* Switch argument and case label can be expressions also, **but case 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:

* 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.

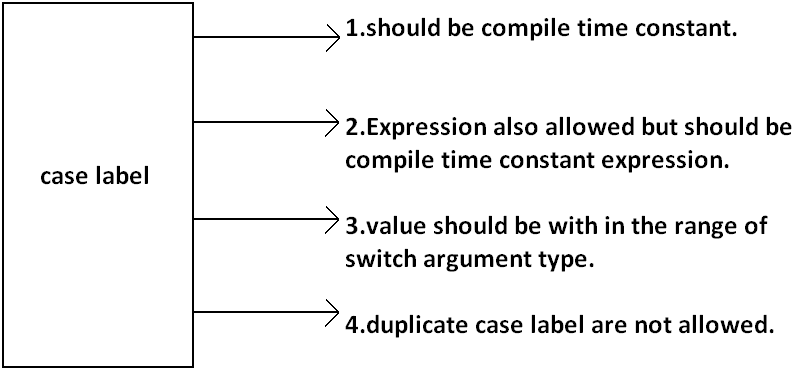
D:\Java>javac ExampleSwitch.java

ExampleSwitch.java:10: duplicate case label

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 .

**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** any where, but at most once 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:**

**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[]){

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:

* 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");

}}

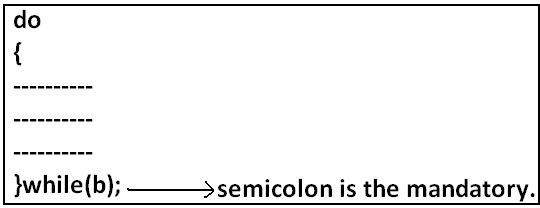
**OUTPUT:**

Hello (infinite times).

**Do-while:**

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

**Syntax:**



* 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{

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

{

System.out.println("hello");

}

while(a>b);

System.out.println("hi");

}}

**Output:**

D:\Java>javac ExampleDoWhile.java

D:\Java>java ExampleDoWhile

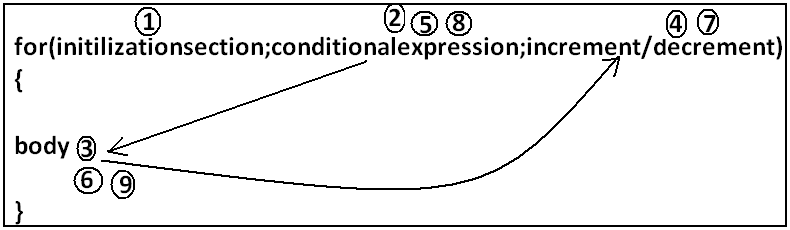
Hello

Hi

**For Loop:**

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

**Syntax:**



1. **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:**

1. Int i=0,j=0; **valid**
2. Int i=0,Boolean b=true; **invalid**
3. 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

1. **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.

1. **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:**

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

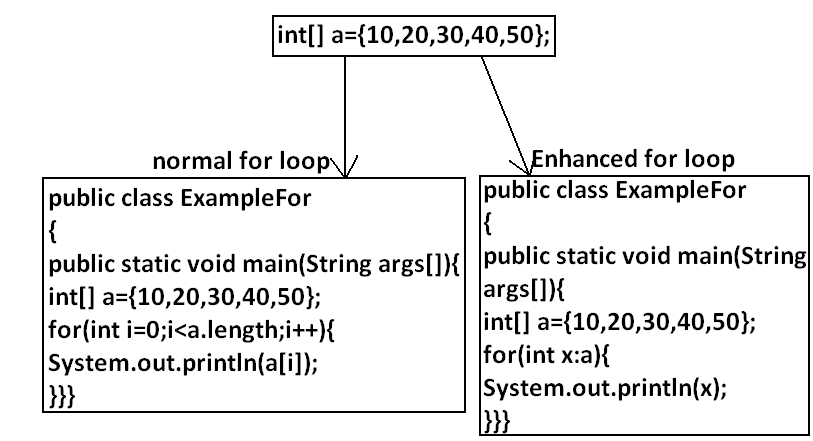
System.out.println("hi");

**For each:**

* 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:**



**Output:**

D:\Java>javac ExampleFor.java

D:\Java>java ExampleFor

10

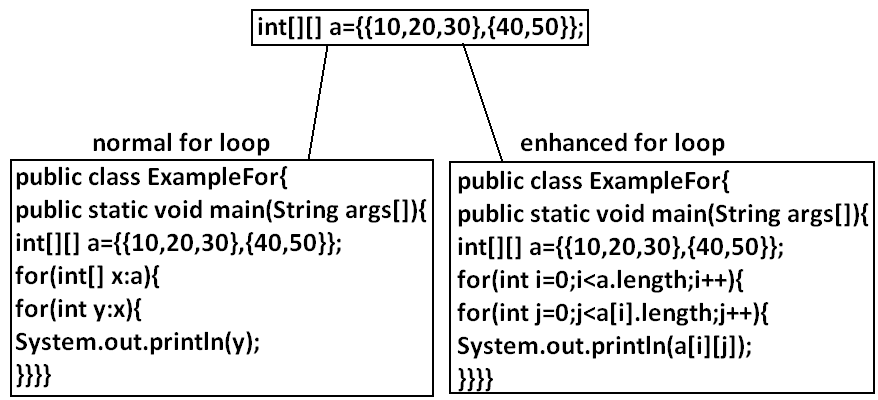
20

30

40

50

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



**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.

**Transfer statements:**

**Break statement:**

* We can use break statement in the following cases.

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

**Example 1:**

class Test{

public static void main(String args[]){

int x=10;

l1:

{

System.out.println("hello");

if(x==10)

break l1;

System.out.println("hi");

}

System.out.println("end");

}}

**Output:**

D:\Java>javac Test.java

D:\Java>java Test

Hello

End

* 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

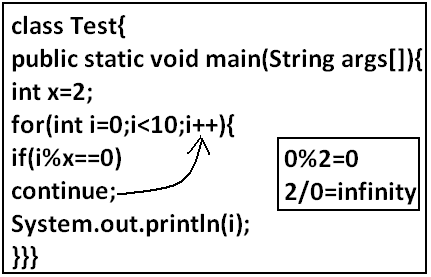
Test.java:5: break outside switch or loop

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

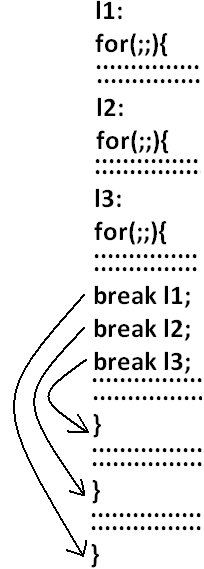
Test.java:6: continue outside of loop

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.

**Syntax:**



**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

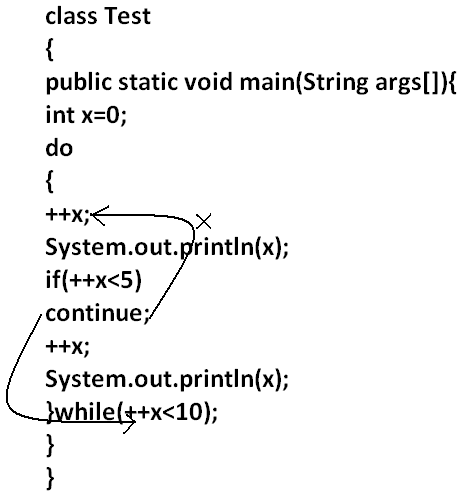
**Continue l1:**

1.........0

2.........0

2.........1

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

****

**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