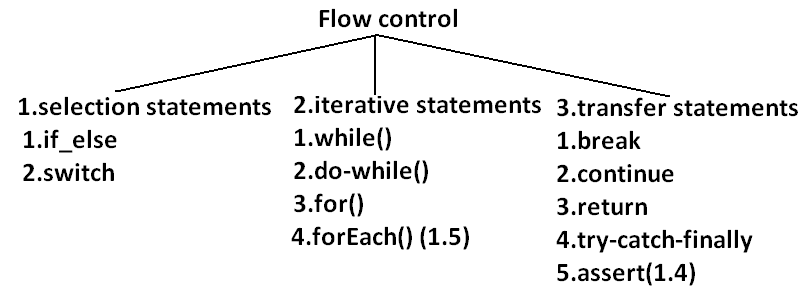
# Flow Control

* Flow control describes in which order the statements need to be executed



**Selection Statements:**

* They are of two types :
  + If conditionals
  + Switch
* **If:**
* If statement is used to test the condition.
* It checks Boolean condition – true or false.
* Various types of if statement are:
  + Simple if
  + If-else
  + If-else ladder
  + Nested if
* **Simple If:**
  + *Syntax*:

if(condition) {

//block of code to be executed

}

* + *Example*:

class SimpleIf {

public static void main(String[] args) {

int age = 18;

if(age >= 18) {

System.out.println(“You are eligible to vote”);

}

}

}

* **If-Else:**
  + *Syntax***:**

if(condition) {

//block of code to be executed if condition is true

} else {

//block of code to be executed if condition is false

}

* + *Example:*

class IfElseExample {

public static void main(String[] args) {

int age = 18;

if(age >= 18) {

System.out.println(“You are eligible to vote”);

} else{

System.out.println(“You are not eligible to vote”);

}

}

}

* **If-else ladder:**
  + *Syntax*:

if(condition1) {

//block of code to be executed if condition 1 is true

} else if(condition2) {

//block to be executed if condition 1 is false and condition2 is true

} else {

//block of code to be executed if both conditions are false

}

* + *Example*:

class IfElseLadder {

public static void main(String[] args) {

int number = 0;

if(number < 0) {

System.out.println(“Negative number”);

} else if(number > 0) {

System.out.println(“Positive number”);

} else {

System.out.println(“ZERO”);

}

}

}

* **Nested If:**
  + *Syntax*:

if(condition1) {

//block of code to be executed if condition 1 is true

if(condition2) {

//block to be executed if condition 1 and 2 are true

}

} else {

//block of code to be executed if condition 1 is false

}

* + *Example*:

class NestedIf {

public static void main(String[] args) {

int age = 18, weight = 50;

if(age > 18) {

if(weight >=50 ) {

System.out.println(“Eligible to donate blood”);

} else {

System.out.println(“Weight is not sufficient”);

}

} else {

System.out.println(“Age is not sufficient”);

}

}

}

* **Switch:**
* Switch statement executes one statement from multiple conditions.
* It is like if-else ladder.
* It tests equality against multiple values.
* It works with byte, short, int, long, enum types & some wrapper types.
* After Java 7, Strings can be used in switch statements
* There can be *1 or N* number of case values in switch statement
* Case value must be of switch expression type value.
* Case value should be *literal* or *constant* and cannot be a variable.
* Case values must be *unique*
* Case value can *default* and it’s optional.
* To break the flow after evaluating expressions use *break* after statements;
* *Syntax*:

switch(expression) {

case value1:

//code to be executed

break; //optional

case value2:

//code to be executed

break; //optional

default:

//default code to be executed if no value is found

}

* *Example*:

class Switch {

public static void main(String[] args) {

int year = 1;

switch(year){

case 1:

System.out.println(“1 year”);

break;

case 2:

System.out.println(“2 year”);

break;

case 3:

System.out.println(“3 year”);

break;

case 4:

System.out.println(“4 year”);

break;

default:

System.out.println(“Invalid year”);

break;

}

}

}

**Iterative/Loop Statements :**

* They are of 4 types :
  + while
  + do-while
  + for
  + foreach
* **while:**
* while statement executes a block of statements when a particular condition is true.
* It executes block of statements until the specified condition is false.
* It condition is true every time then loop run into infinite.
* *Syntax*:

while(condition) {

//block of code to be executed if condition is true

}

* *Example*:

class WhileExample {

public static void main(String[] args) {

int number = 10;

while(number > 0) {

System.out.println(“Number :” + number);

number--;

}

}

}

* **do-while:**
* do-while evaluates its expression at the bottom of the loop instead of the top.
* Statements within do block executes at least once.
* *Syntax*:

do {

//block of code to be executed

}

while(condition);

* *Example*:

class DoWhileExample {

public static void main(String[] args) {

int number = 10;

do {

System.out.println(“Number :” + number);

number--;

} while(number > 0);

}

}

* **for:**
* for statement provides a compact way to iterate over range of values.
* ‘*for loop’* repeatedly loops until a specific condition is satisfied.
* *‘for loop’* has 3 parts:
  + *Initialization –* initializes the loop & executes only once at beginning
  + *Termination condition –* loops until this condition is false
  + *Iteration –* invoked after each iteration through the loop. Increament/decrement a value.
* *‘for loop’* can be used over lists/arrays whose size is known.
* *Syntax*:

for(initialization; termination; iteration) {

//block of code to be executed

}

* *Example*:

class DoWhileExample {

public static void main(String[] args) {

int[] numbers = {1,2,3,4,5,6,7,8,9,10};

for(int i=0;i< numbers.length; i++) {

System.out.println(“Number :” + numbers[i]);

}

}

}

* **forEach / Enhanced for loop:**
* forEach statement is available from Java 1.5
* ‘*for each’* is primarily used to access elements from arrays/collections even of unknown size.
* *Syntax*:

for(DataType var: varCollection) {

//block of code to be executed

}

* *Example*:

class ForEachExample {

public static void main(String[] args) {

int[] numbers = {1,2,3,4,5,6,7,8,9,10};

for(int i:numbers) {

System.out.println(“Number :” + i);

}

}

}

**Transfer Statements :**

* They are of 3 types :
  + break
  + continue
  + return
* **break:**
* break statement is used to break loop loop or switch statement.
* It breaks the current flow of the program at specified condition
* *Syntax*:

jumpStatement;

break;

* *Example*:

class ForEachExample {

public static void main(String[] args) {

int[] numbers = {1,2,3,4,5,6,7,8,9,10};

for(int i:numbers) {

if(i==5) break;

System.out.println(“Number :” + i);

}

}

}

* **continue:**
* continue statement is used to jump to next iteration of loop immediately.
* It skips remaining statement in block.
* *Syntax*:

jumpStatement;

continue;

* *Example*:

class ForEachExample {

public static void main(String[] args) {

int[] numbers = {1,2,3,4,5,6,7,8,9,10};

for(int i:numbers) {

if(i==5) continue;

System.out.println(“Number :” + i);

}

}

}

* **return:**
* return statement is used to exit from the method.
* It skips remaining statements in method.
* *Syntax*:

jumpStatement;

return;

* *Example*:

class ForEachExample {

public static void main(String[] args) {

int[] numbers = {1,2,3,4,5,6,7,8,9,10};

for(int i:numbers) {

if(i==5) return;

System.out.println(“Number :” + i);

}

}

}

**\*\*try-catch-finally & assert will be learnt in future sections\*\***