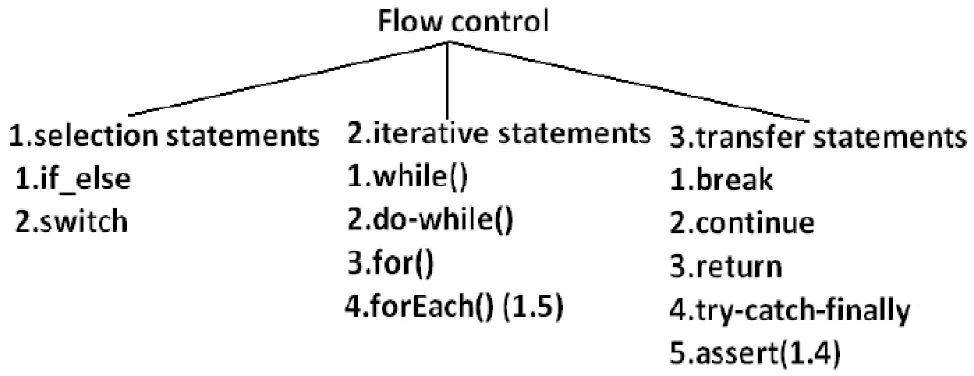
Flow Control Statements :

While programming, we need to manage the flow of the program execution. To describe the order in which program statements will be executed, we have flow control statements.

Flow control statements classification:

There are various types of flow control statements as follows:



Selection Statements:

Java Selection statements are used to executive a particular statement when some condition is fulfilled or not.

Selective statement can be further classified as :

1. if – else
2. switch

1) if-else:

If-else is used to executive a statement/statements when a particular condition is true or false.

Note : Note that while providing the condition for if else, it should be Boolean type always.

It should be Boolean always, even if it's some expression still it's net result should be of Boolean type

Syntax:
```
	if(condition){
	//if condition is true executive if block only
	}else{
	//if condition is false then executive else block only
	}
```

Example 1:
```
public class Demo {
	public static void main(String args[]) {
		int x = 0;
		if (x) {
			System.out.println("hello");
		} else {
			System.out.println("hi");
		}
	}
}
```
OUTPUT:
```
Compile time error:
Demo.java:4: incompatible types
found : int
required: Boolean if(x)
```

Example 2:
```
public class Demo {
	public static void main(String args[]) {
		boolean x = true;
		if (x) {
			System.out.println("hello");
		} else {
			System.out.println("hi");
		}
	}
}
```

OUTPUT:

```
hello
```

Note:

1) **Curly braces after if and else are optional. In such case, we can write only one statement after if and else , which should not be declarative statement.**

   For ex:

   ```
   if(true)
           System.out.println("in if block");
   else
           System.out.println("in else block");
   ```

   This is valid. Note there are no curely braces after if and else.But there must be only one statement which should not be declarative statement.

   Ex2.

   ```
   if(x)
       int y=10;
   else
       int z=100;
   ```

   This is invalid as the statement after if else without curly braces are declarative statements.

2) **If can exist without else. Else block is optional.**

   For ex.

   ```
   if(2<10){
           System.out.println("in if block");
           }
   ```

   Output:

   ```
   in if block
   ```

3) **Nesting of if else is possible. Nesting means writing if/else blocks within other if/else block.**

   For ex.

```
   if(x){
           if(y){
           //execute this
           }else{
           //execute this
           }
   }else{
           if(z){
           //execute this
           }
   }
```

4) **If we want to check multiple conditions out of which only one conditions should be executed then we use if-else-if ladder**

   For ex.

```
   int x=3;
           if(x==1){
                   System.out.println("Value of x is one");
           }else if(x==2){
                   System.out.println("Value of x is two");
           } else if(x==3){
                   System.out.println("Value of x is three");
           }
```

   Output:

```
   Value of x is three
```

In if-else-if ladder we can use a default condition which will execute if none of the condition is matched.

For ex.

```
int x=4;
        if(x==1){
              System.out.println("Value of x is one");
        }else if(x==2){
              System.out.println("Value of x is two");
        } else if(x==3){
              System.out.println("Value of x is three");
        }else{
              System.out.println("Value of x is not matched");
        }
```

Output:

```
   Value of x is not matched
```

2) switch:

If multiple options are available, then instead of using if-else-if ladder/if-else multiple times we should go for switch statement as it improves the readability of code.

Syntax:

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

For ex.

```
int x=3;
        switch(x){
        case 1:
        System.out.println("Executing case 1");
        case 2:
        System.out.println("Executing case 2");
        case 3:
        System.out.println("Executing case 3");
        default:
        System.out.println("Executing default case ");
        }
```

Output:

```
        Executing case 3
```

Note:

1) **If no case is matched then default case will be executed.**
2) **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.**
3) **Curly braces are mandatory.(except switch case in all remaining cases curly braces are optional)**
4) **Both case and default are optional.**
5) **Every statement inside switch must be under some case (or) default. Independent statements are not allowed.**
6) **Every case label should be within the range of switch argument type.**
7) **Duplicate case labels are not allowed.**
8) **Within 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 .

9) **Within the switch we can take the default only once**
10) **Within the switch we can take the default anywhere, but it is convention to take default as last case.**