

# Day-3.1

# Agenda

**Operators**

**Type Casting**

**Array**

**Flow Control**

# Operators

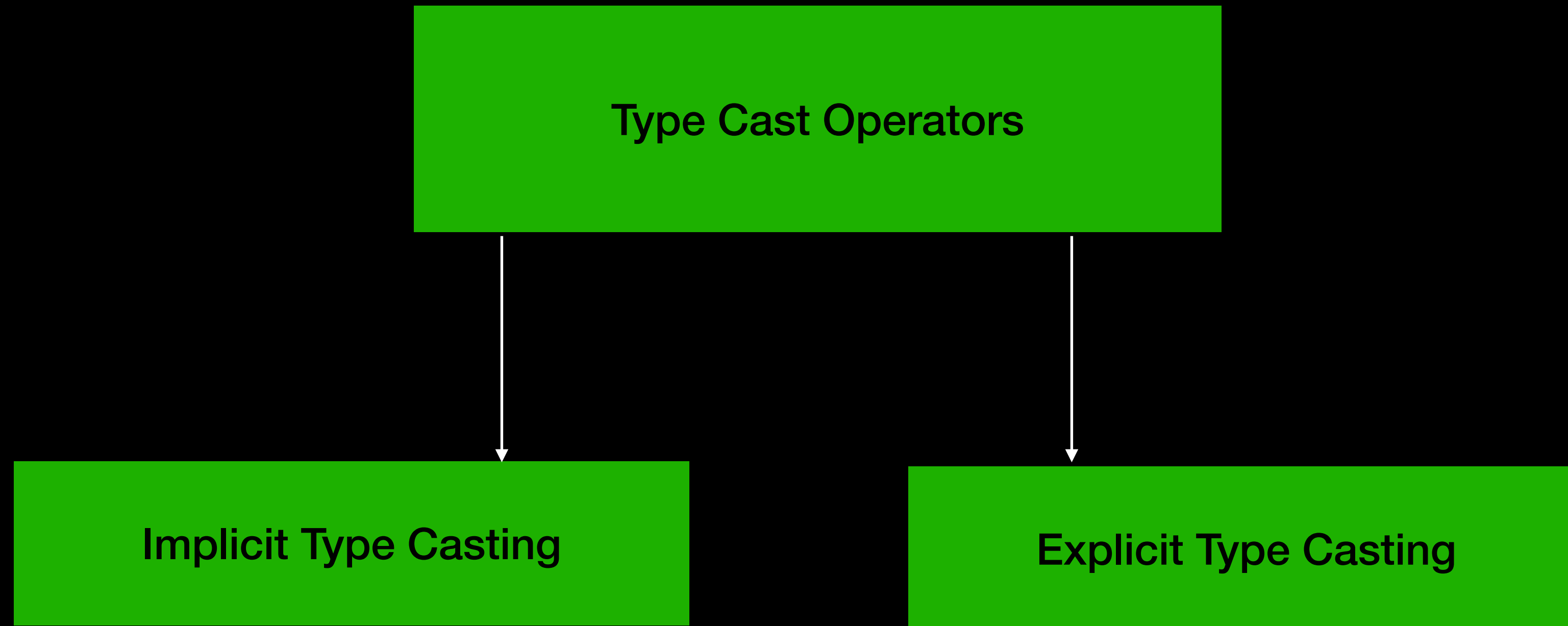
## Bit-wise Operators [ & , | , ^ ]

Operator	Meaning	Definition
&	AND	If both operands are True then result is True
	OR	If at least 1 operand is True then result is True
^	X-OR	If both operands are different then result is true

A	B	A & b
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A   b
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A ^ b
0	0	0
0	1	1
1	0	1
1	1	0



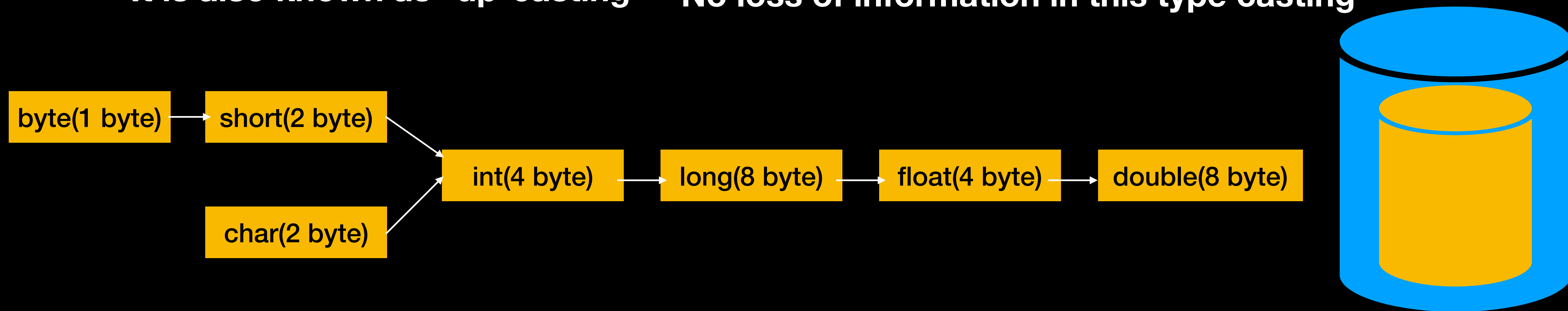
# Type Casting

## Implicit Type Casting

**Compiler is the responsible to perform this type casting .**

**This typecasting is required when ever we are assigning smaller data types value to bigger  
To the bigger data type variables.**

**It is also known as “up-casting ” No loss of information in this type casting**

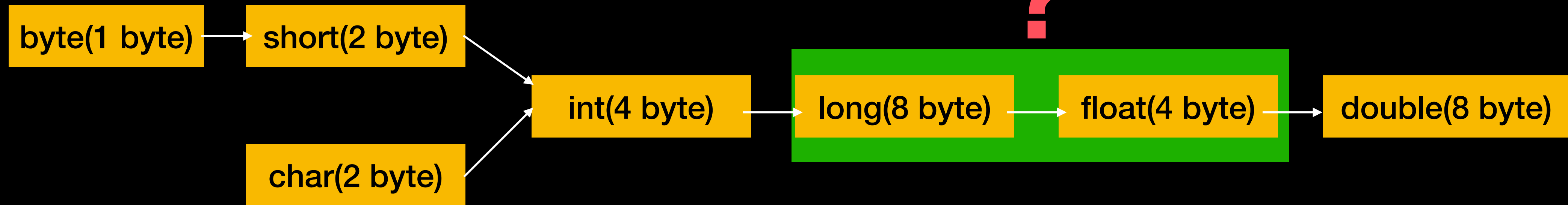


# Type Casting

## Implicit Type Casting

# WHY

# ?

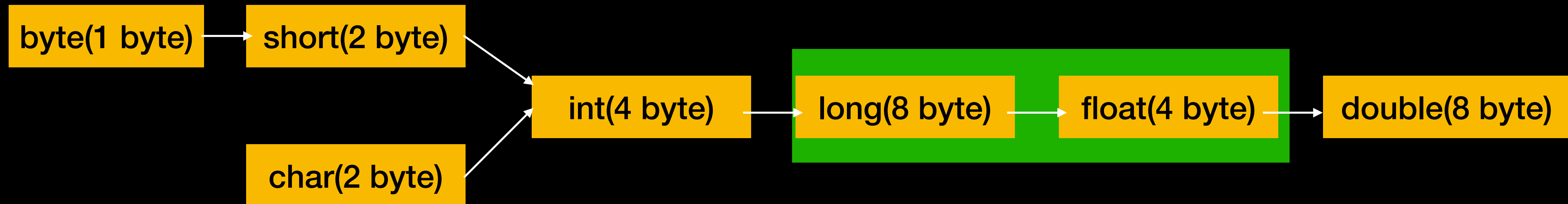


# Type Casting

## Implicit Type Casting

**Although long use more bytes , but it has a smaller rang , max size is  $2^{63}$**

**Where as float can got up to  $2^{127}$**



# Type Casting

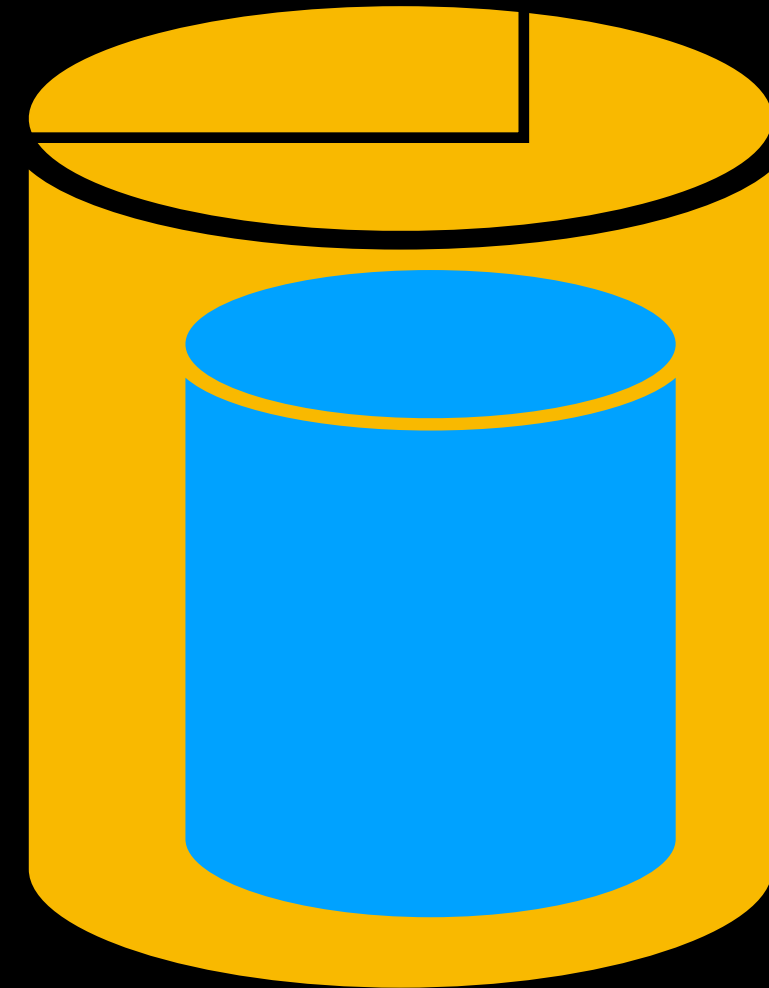
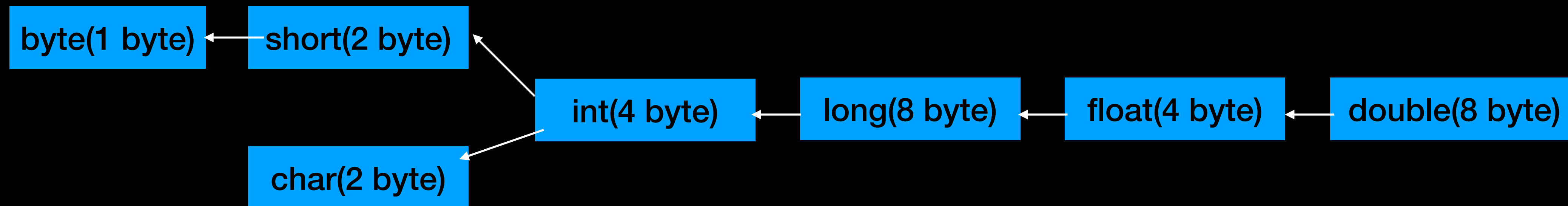
## Explicit Type Casting

**Programmer is responsible to perform the type casting**

**This typecasting is required when ever we are assigning bigger data types value to the smaller data type variables.**

**It is also known as “down-casting ”**

**There may be a chance of loss of information in this type casting**





**An array is an indexed collection of fixed no of homogeneous data element**

**The main advantage of array is we can represent multiple values under the same name.**

**But the main limitations of array is once we can create an array there is no chance of increasing/decreasing size based on our requirements .**

### Types of Array

Single Dimension Array

Double Dimensions Array or 2D Array

Triple Dimensions Array or 3D Array

Single Dimension Array

1. `int[] a;`

2. `int a [];`

3. `int []a;`

Double Dimensions Array or 2D Array

1. `int[][] a;`

2. `int [][] a;`

3. `int a[][];`

4. `int[] a[];`

5. `int[] []a ;`

6. `int []a [];`

Triple Dimensions Array or 3D Array

1. `int[][][] a;`

2. `int a[][][];`

3. `int [][][] a;`

4. `int[] [][]a ;`

5. `int[] a[][];`

6. `int[] [] a [];`

7. `int[][] [] a;`

8. `int[][] a[];`

9. `int [][]a[];`

10. `int [] a [][];`

## Array Construction

### Single Dimension Array

```
int[ ] a = new int[SIZE];
```

### Double Dimensions Array or 2D Array

```
int[ ][ ] a = new int[ ] [ ];
```

### Triple Dimensions Array or 3D Array

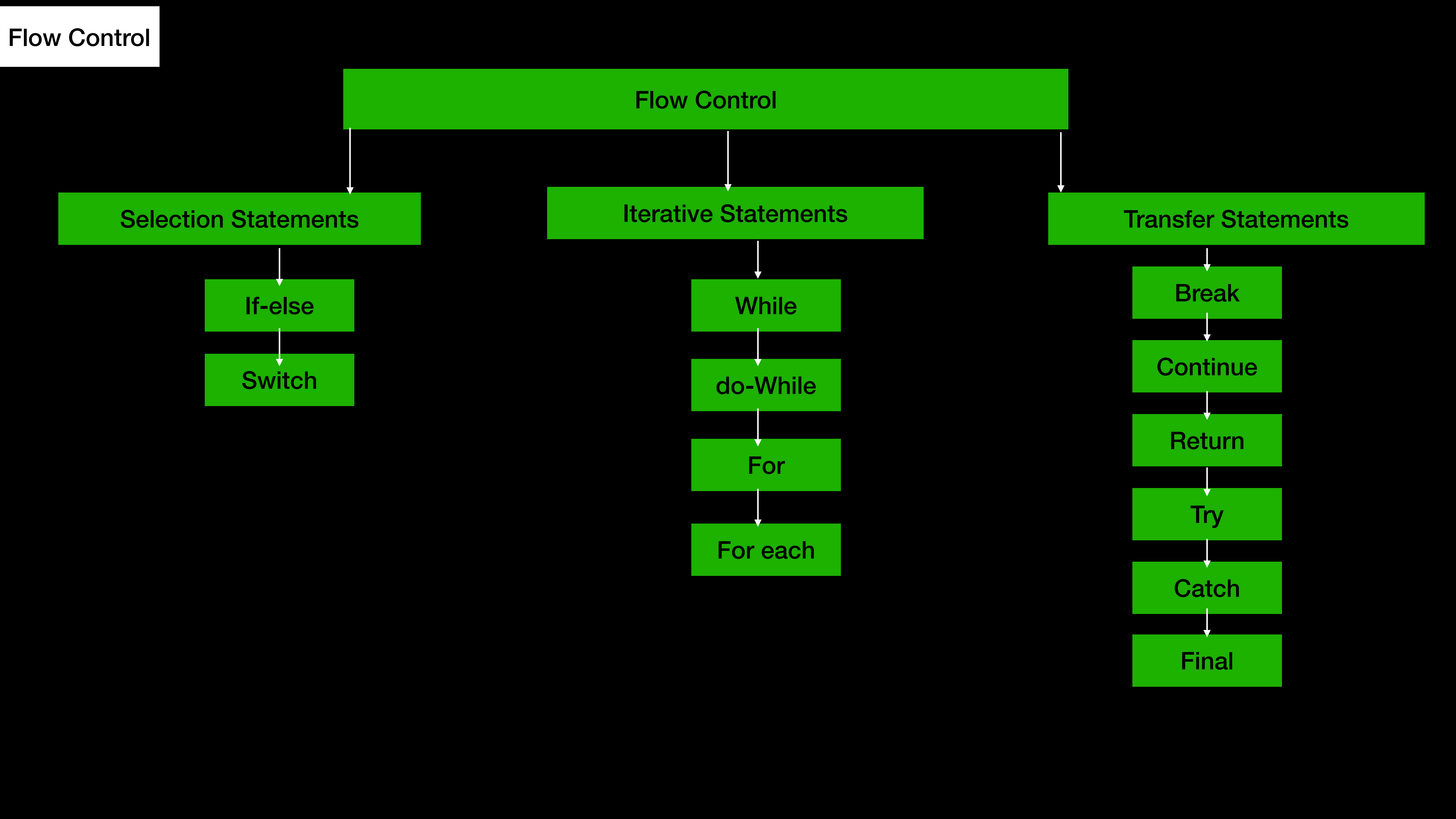
```
int[ ][ ] [ ] a = new int[ ] [ ] [ ];
```

## Array Operation

1. Insert

2. Search/Find

3. Update



Flow Control

Selection Statements

If-else

Switch

Iterative Statements

While

do-While

For

For each

Transfer Statements

Break

Continue

Return

Try

Catch

Final

## If-else

The argument to the if statement should be boolean type , if we are providing any other we will get compilation error

### Syntax

```
if(b){  
    //Action if b is true  
}else{  
    // Action if b is false  
}
```

## Switch Statement

If several option are possible then it is never recommended to use if-else , we should go for switch

### Syntax

```
Switch(x){  
  Case 1 :  
      //Action ;  
  Case 2 :  
      //Action ;  
  Case 3 :  
      //Action ;  
  .....  
  .....  
  Default :  
      Default Action ;  
}
```

### Before JAVA : 5

**byte**

**short**

**int**

**char**

### In JAVA : 7

**String**

### After JAVA : 5

**Byte**

**Short**

**Int**

**Char**

**enum**

### while

**If we do not know the no of statement in advance then the best suitable loop is while loop**

### Syntax

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



### do while

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

### Syntax

```
do{  
    //Logic  
}while(b);
```

for loop

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

**Syntax**

```
do{  
    //Logic  
}while(b);
```