Assignment3
Q1 ->
```
Invoice i1 = new Invoice();
Invoice i2 = new Invoice("","",0,0.0);
i1.set()
```
Q2 ->

```
package com.sunbeam.p1;
```

Access modifiers
private
default
protected
public

public
default

Types of methods
this(1,2);

this reference
//this -> to store address of current calling object

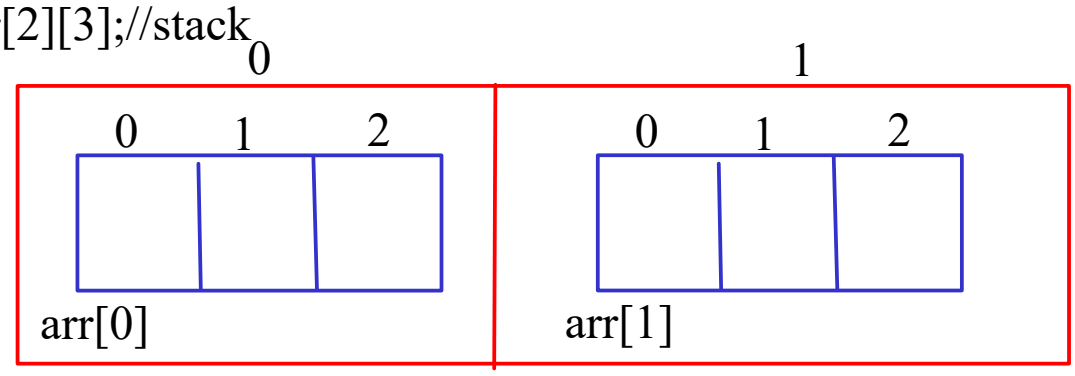
Array
- It is areference type in java

```
int arr[] =  new int[5];
```

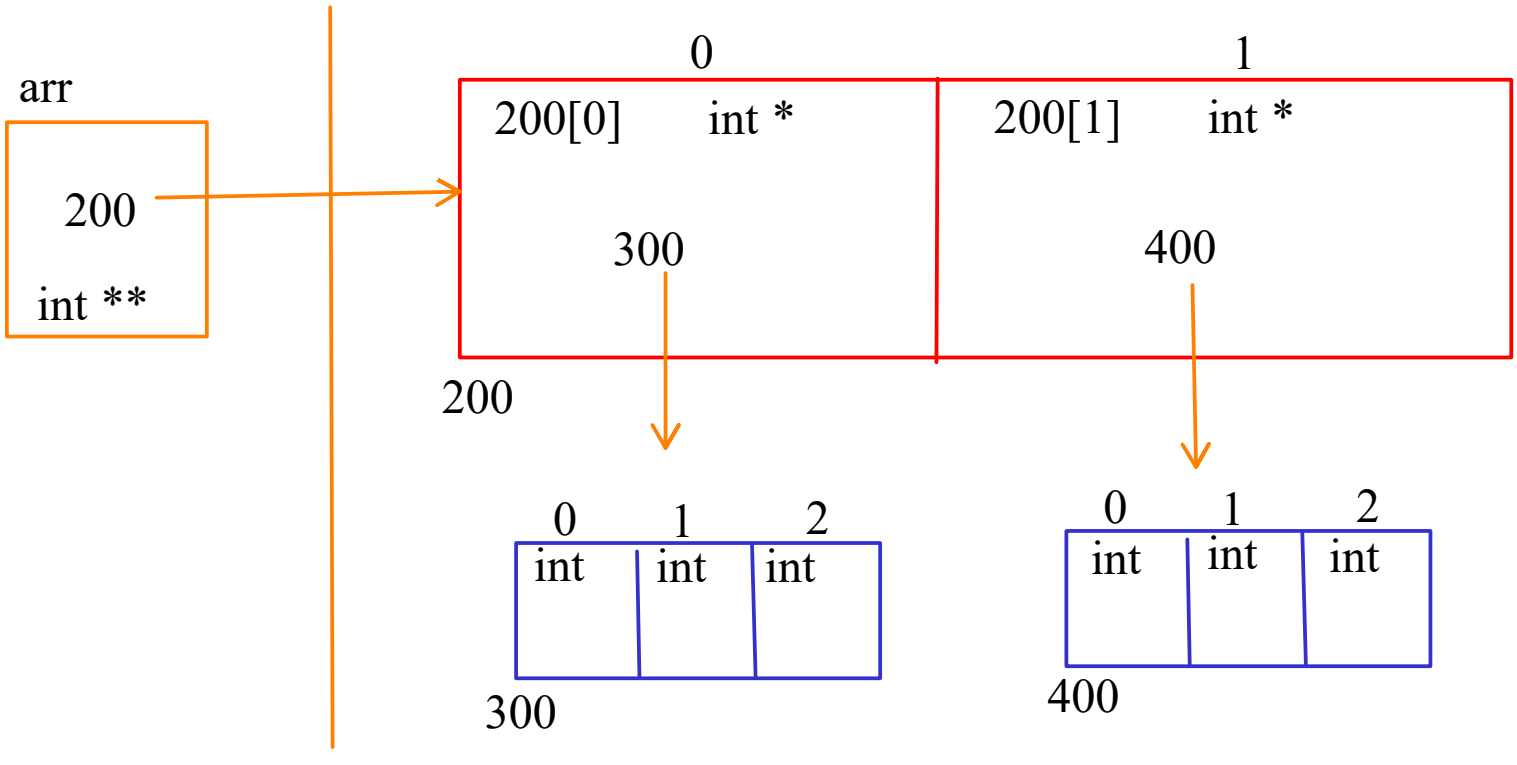
MultiDimension Array

In CPP,

```
int arr[2][3];//stack
```

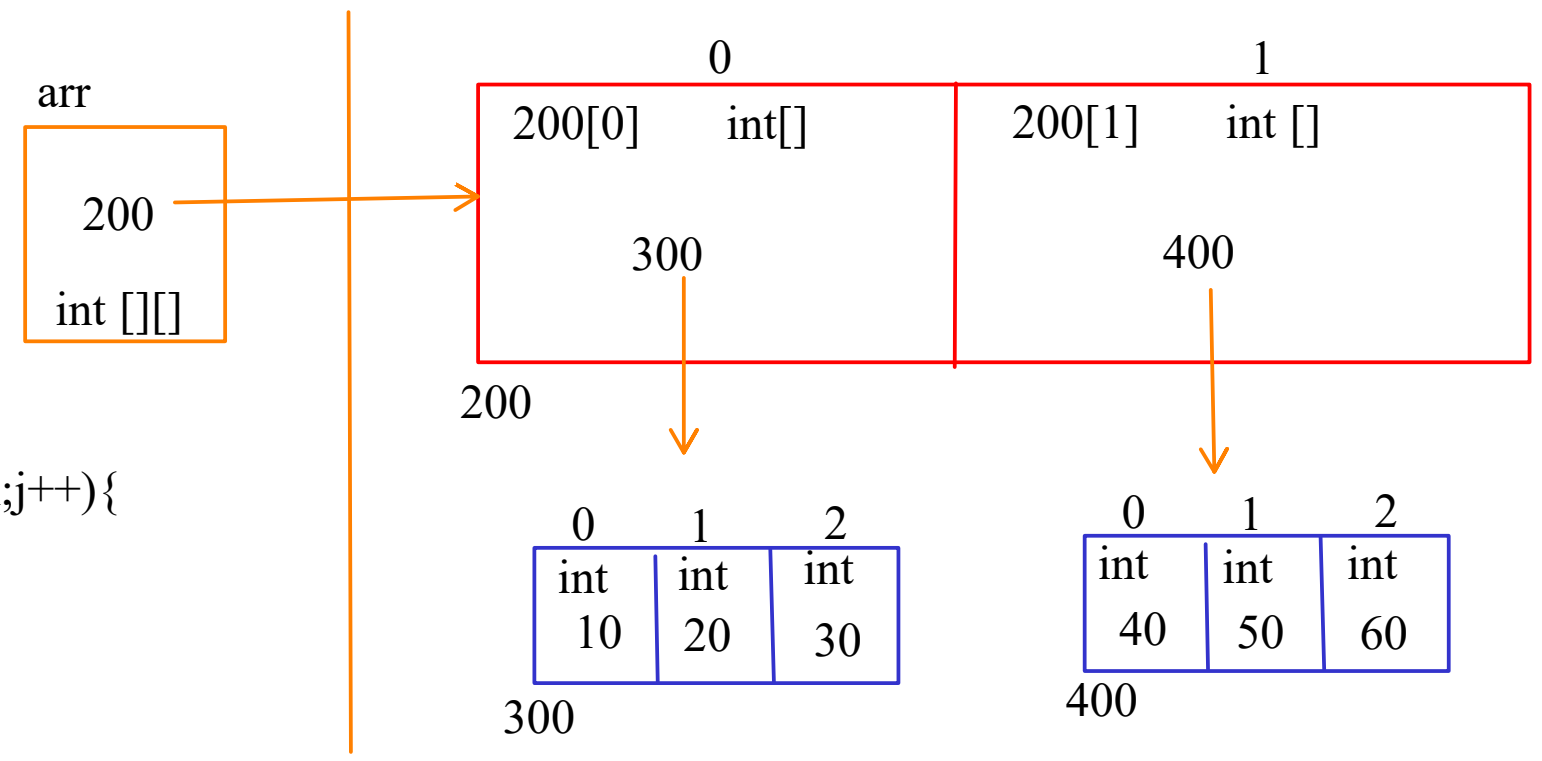



```
// heap
int **arr = new int*[2];
arr[0] = new int[3];
arr[1] = new int[3];
```

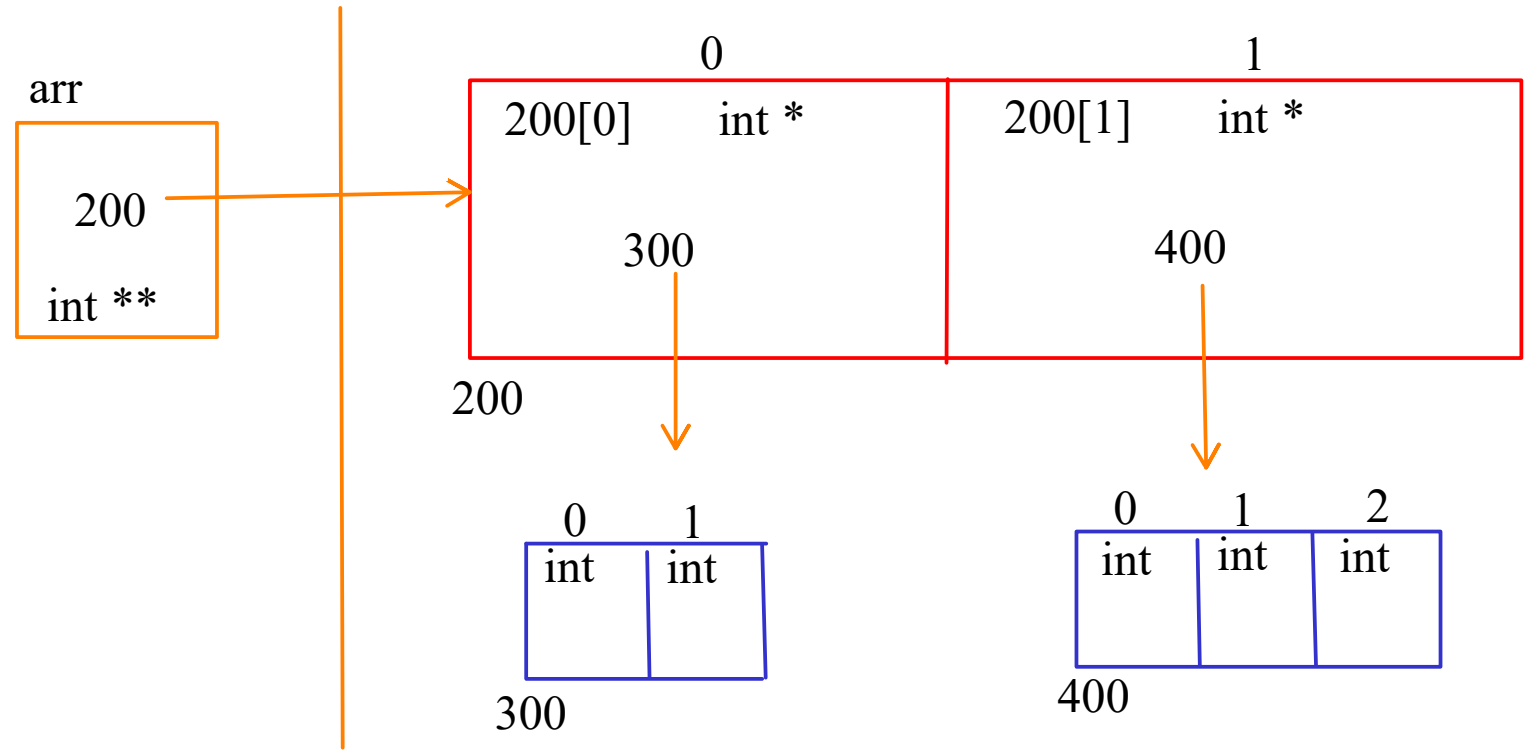


In Java

```
int [][] arr = new int[2][3];

300[0] = 10;
200[0][0] = 10;
arr[0][0] = 10;

for(int i=0;i<arr.length;i++){
        for(int j=0;j<arr[i].length;j++){
                sysout(arr[i][j]);
        }
}

for(int[] ele: arr)
        for(int e : ele)
                sysout(e);
```

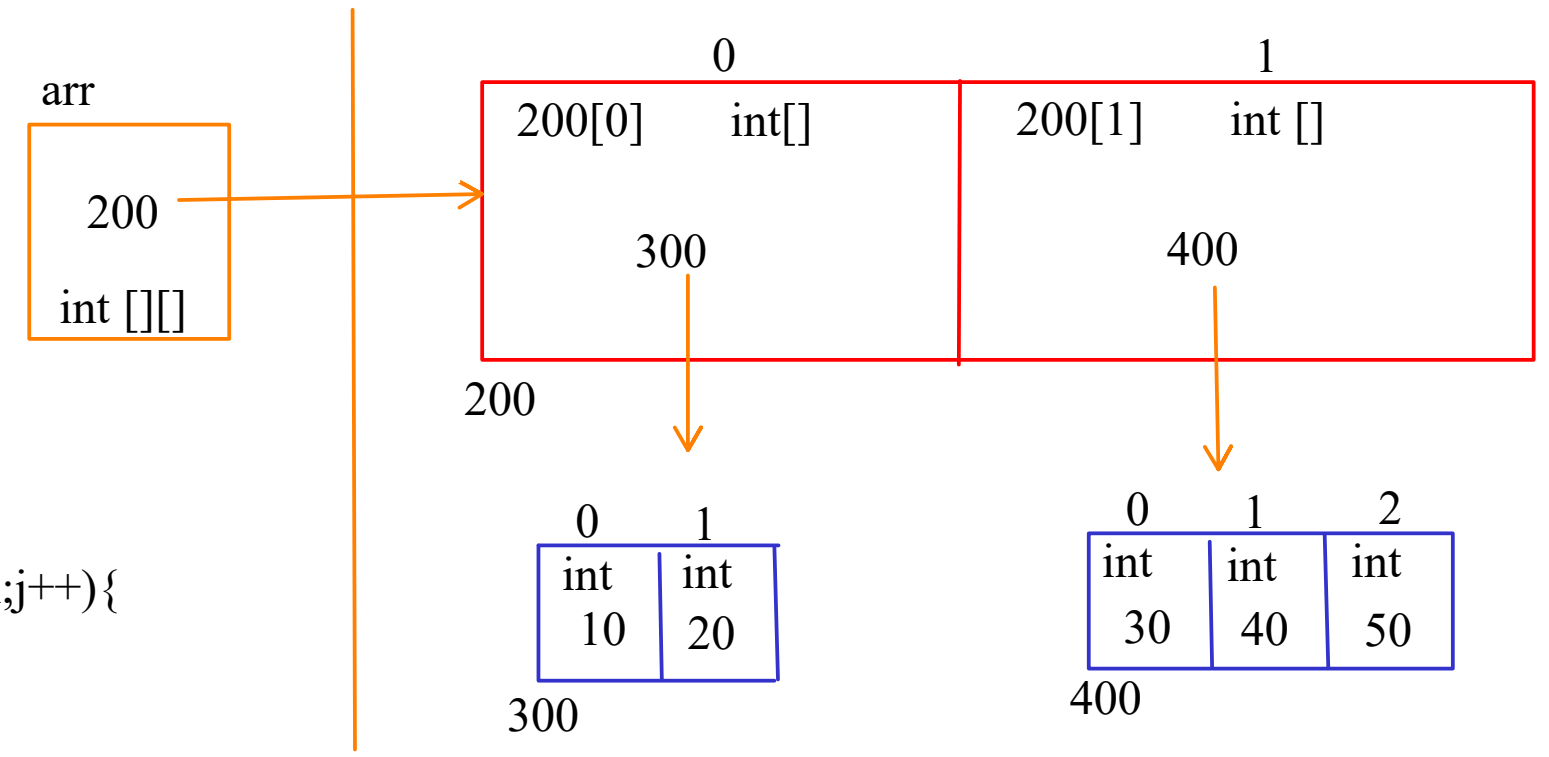


```
// heap
int **arr = new int*[2];
arr[0] = new int[2];
arr[1] = new int[3];
```

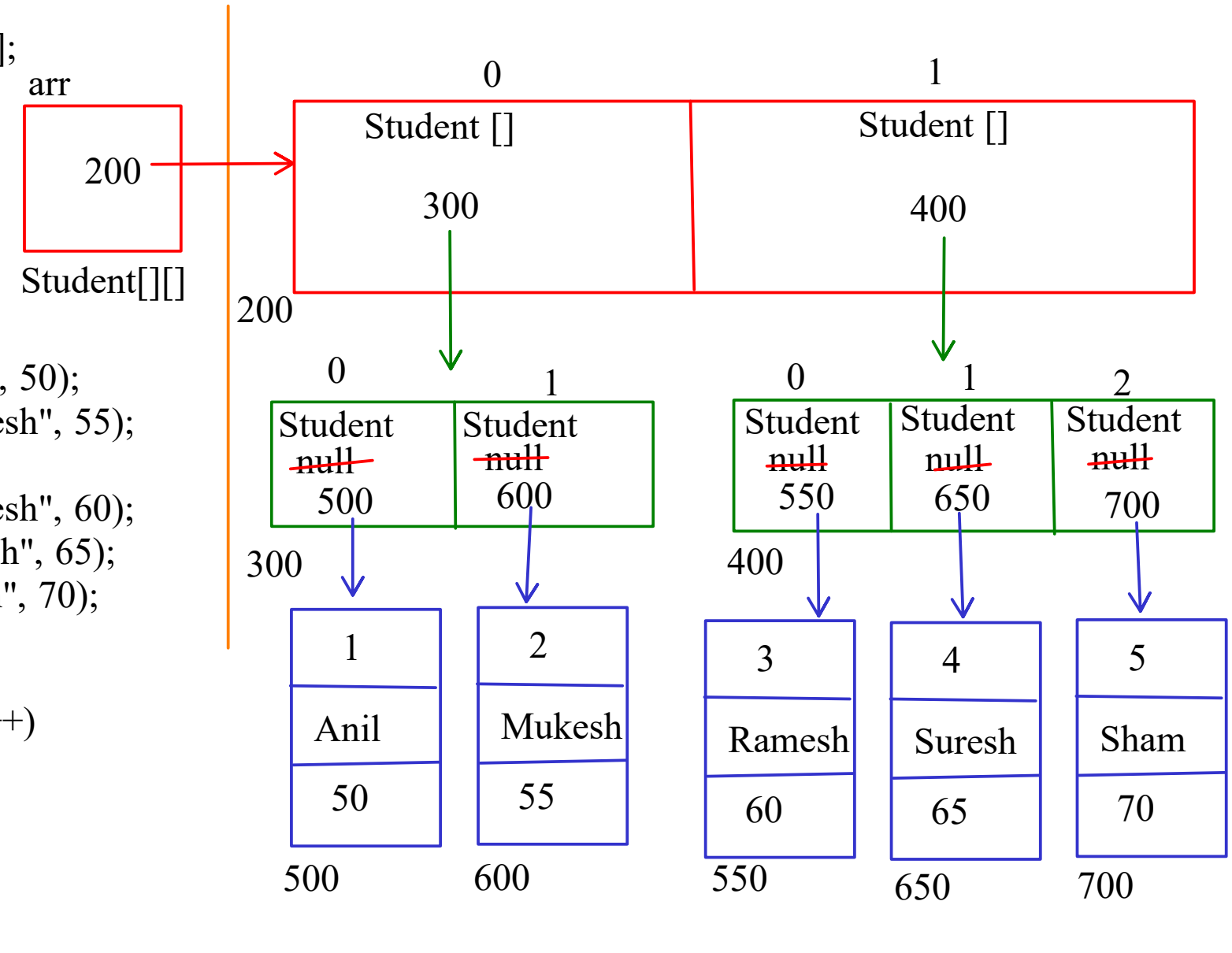


In Java
Raggged Array

```
int [][] arr = new int[2][];
arr[0] = new int[2];
arr[1] = new int[3];

300[0] = 10;
200[0][0] = 10;
arr[0][0] = 10;

for(int i=0;i<arr.length;i++){
        for(int j=0;j<arr[i].length;j++){
                sysout(arr[i][j]);
        }
}

for(int[] ele: arr)
        for(int e : ele)
                sysout(e);
```




```
Student[][] arr = new Student[2][];
arr[0] = new Student[2];
arr[1] = new Student[3];

300[0]= new Student();
200[0][0]= new Student();
arr[0][0]= new Student();

arr[0][0] = new Student(1, "Anil", 50);
arr[0][1] = new Student(2, "Mukesh", 55);

arr[1][0] = new Student(3, "Ramesh", 60);
arr[1][1] = new Student(4, "Suresh", 65);
arr[1][2] = new Student(5, "Sham", 70);

for(int i=0;i<arr.length;i++)
        for(int j=0;j<arr[i].length;j++)
                arr[i][j].display();

for(Student [] ele: arr)
        for(Student s : ele)
                s.display();
```



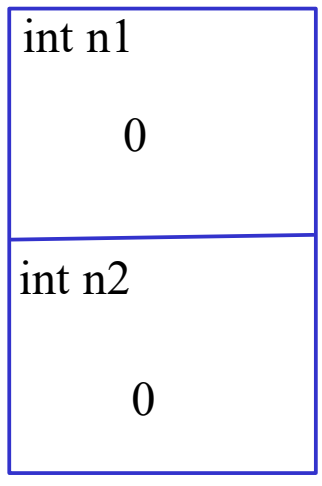
Initializer
1. Field Initializer
    - we can initialize the fields of the class at the declaration itself. this is called as field initializer
2. Object Initializer
    - we can put a block inside a class and can initialize the fields inside it. it is called as Object initializer
3. Constructor
    - It is a special method of a class

- Always the field initializer will get called first, then the object initializer and then the constructor.

Final
- In java we can make
1. Variable as a final
2. Field as a final
3. Method as a final
4. Class as a final

In cpp,
- we can make below members as constant
variable
pointer
data memebrs
member functions
object

```
class Account{
const int accno;

Account():accno(1001)
};

Accouont *ptr = new Account();
```




```
{
int num
}
        int num
```

```
int getMin() const
{
return min
}
```

new Test();

```
        class Test{

        void display() // Test *const this;
        {

        }
        };
```

```
        class Test{
void display()// final Test this
{

}
}
```

Final
- In java we can make
1. Variable as a final
    - Once initialized we cannot change the value inside it.
2. Field as a final
    - We can initialize it either in field initializer or in object initializer or in constructor
    - Once initialized we cannot change the value inside it.
3. Method as a final
    - We cannot override final methods
4. Class as a final
    - we cannot extend final classes

Static Fields
- The fields of the class that are desiged to be shared across multiple objects.
- these fields get the memory on the method area only once during class loading.
- the static fields can be initialized inside the field initializer or inside the static block
- static block is executed only once.

```
class Circle{
int radius;
static double PI;

Circle(int radius){
this->radius = radius;
}

void calculateArea(){

}
};
double Circle::PI = 3.14;
```

Circle c1(5); -> 12
Circle c2(7); -> 12
Circle c3(9); -> 12

8 bytes -> PI
Circle c1(5); -> 4
Circle c2(7); -> 4
Circle c3(9); -> 4

```
int val = Integer.parseInt(String s);
```

```
class System
static InputStream in
static PrintStream out
err
```

java.util.Arrays

Lab ->
1. Complete the pending assignment
2. initializers
3. Final
4. Static

```
//Tester
Point p1 = new Point(1,2);
p1.getDeatils();

Point p2 = new Point(3,4);
p2.getDeatils();

if(p1.isEqual(p2))

p1.accept();
p2.accept();
dist = p1.calculateDistance(p2);
p1.calculateDistance(p2);
```

```
class Point2D{
int x;
int y;

public String getDetails(){


}

public boolen isEqual(Point p2)//this-> p1
{

}

public double/void calculateDistance(Point p2)// this->p1
{

}

public void accept(){

}

}
```