```
com.gemotery
Point2D{
int x
int y

boolean isEqual(){

}

String getDeatils(){
return x+","+y
}
}
```

Class,Object,Reference

Types of Methods
Ctor ()
- Parameterless
- Parameterized
Settters
Getters
Facilitators

```
Date(){
this(1,1,1900) // ctor chaining
}

Date(int d, int m, int y){
this.d = d;
this.m = m;
this.y = y;
}
```

```
void setDay(int day){
this.day = day;
}
```

```
int getDay(){
return day;
}
```

Initializers
1. Field
2. Object
3. Ctor

```
class Test{
int num1 = 10; // Field Initializer
int num2;
int num3;

// Object Initializer
{
num2 = 20;
}

// Object Initializer
{
num2 = 30;
}

// Ctor
public Test(){
        num3=30;
}

}
```

```
Test t1 = new Test();
```

Array
- Array is reference type in java
1. Single Dimension
2. Multi Dimension
3. Ragged

```
int arr[]; // reference

arr = new int[5];
```

0	1	2	3	4
int 10	int 20	int 30	int 40	int 50

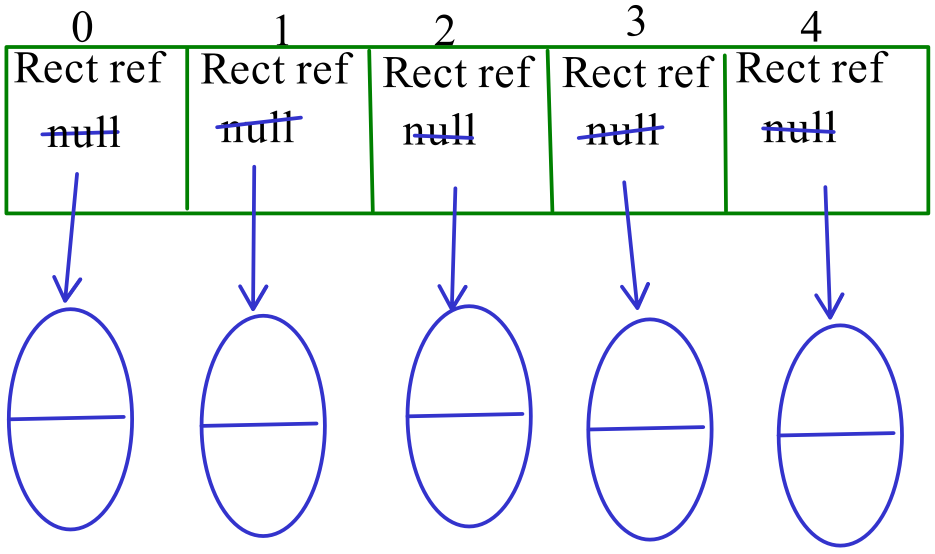
arr

```
arr[0]=10
arr[1]=20
arr[2]=30
```

```
for(int element: arr)
    sop(element)
```

```
class Rectangle{
int length;
int breadth;

void display(){

}
}
```

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

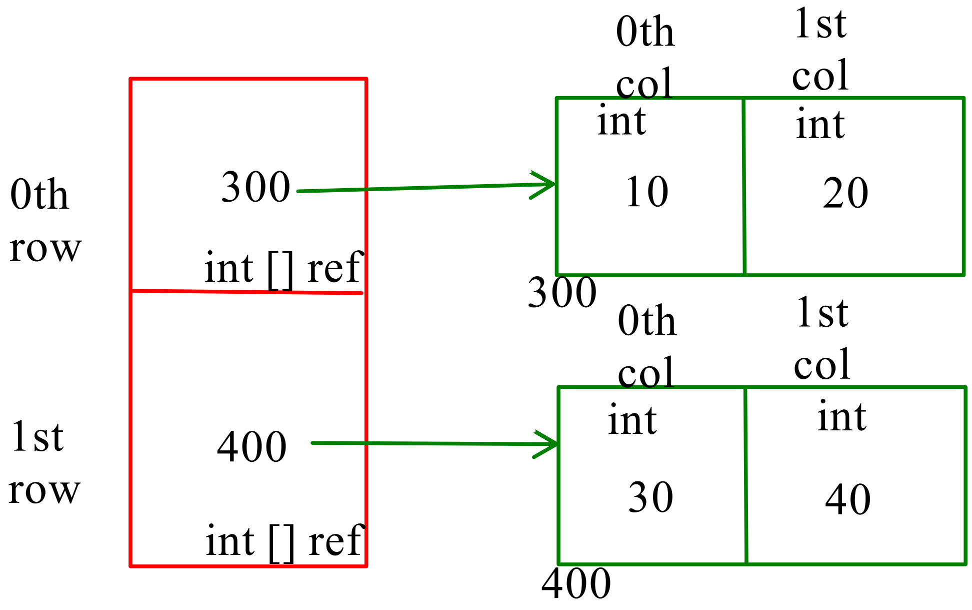


```
arr[0] = new Rectangle(10,20);
arr[1] = new Rectangle(11,21);
arr[2] = new Rectangle(12,22);
arr[3] = new Rectangle(13,23);
arr[4] = new Rectangle(14,24);
```

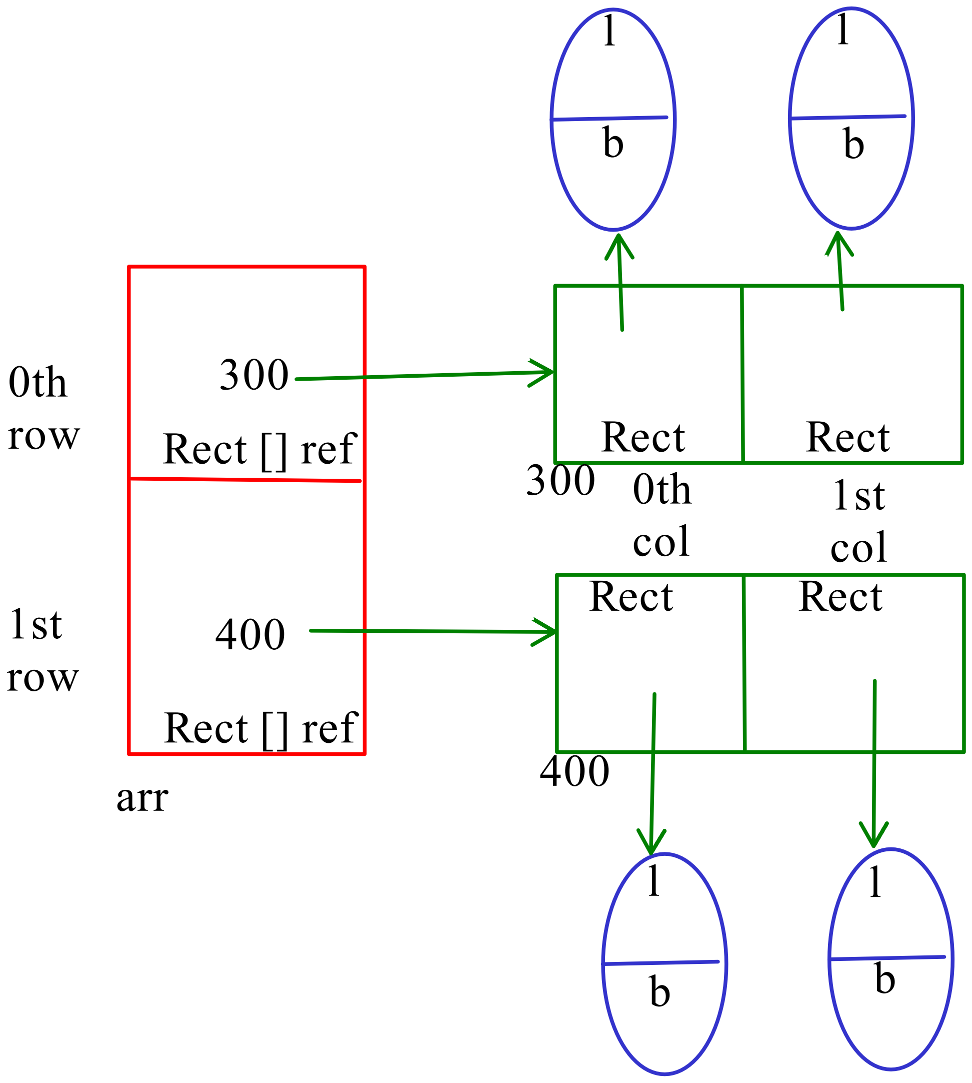
```
for (Rectangle ref : arr )
    ref.display();
```

Multidimensional Arrary -> Primitive type

```
int[][] arr; //reference
arr = new int[2][2];
//arr = new int[rows][cols]
```

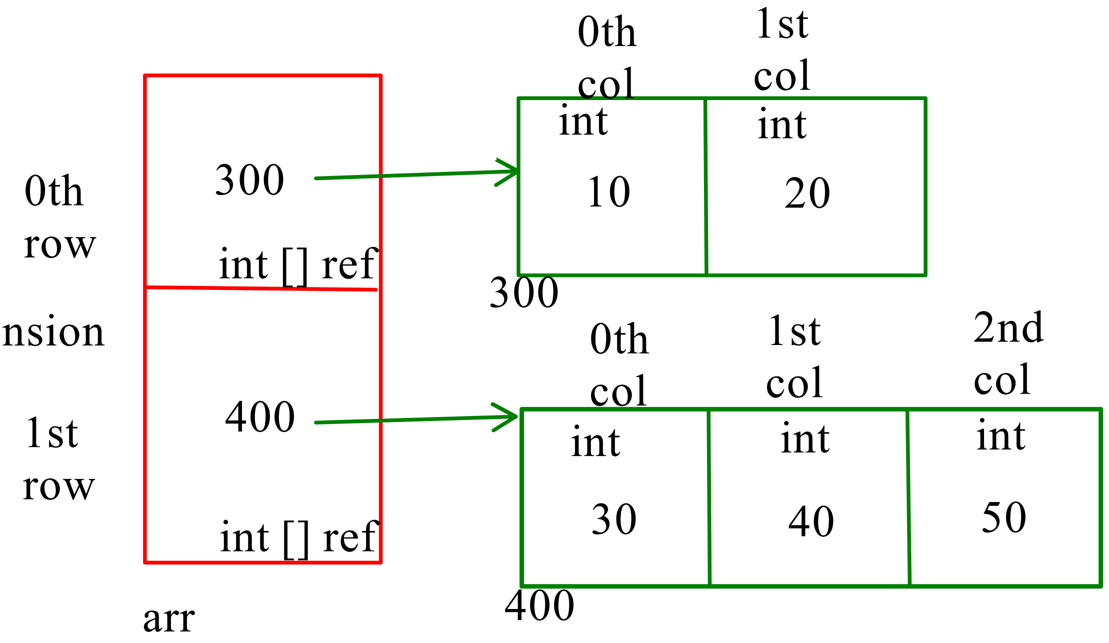


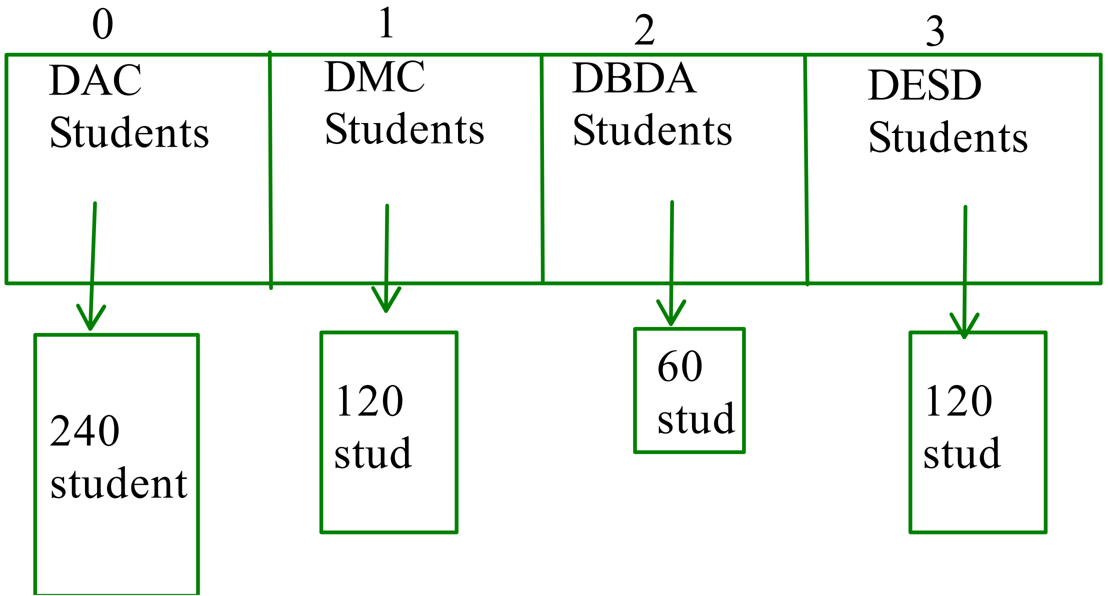
```
Rectangle[][] arr; //reference
arr = new Rectangle[2][2];
//arr = new Rectangle[rows][cols]

arr[0][0] = new Rectangle();
arr[0][1]= new Rectangle();
arr[1][0] = new Rectangle();
arr[1][1]= new Rectangle();


 for (Rectangle [] ref : arr)
        for(Rectangle ele : ref)
                ele.display();
```



3. Ragged Array

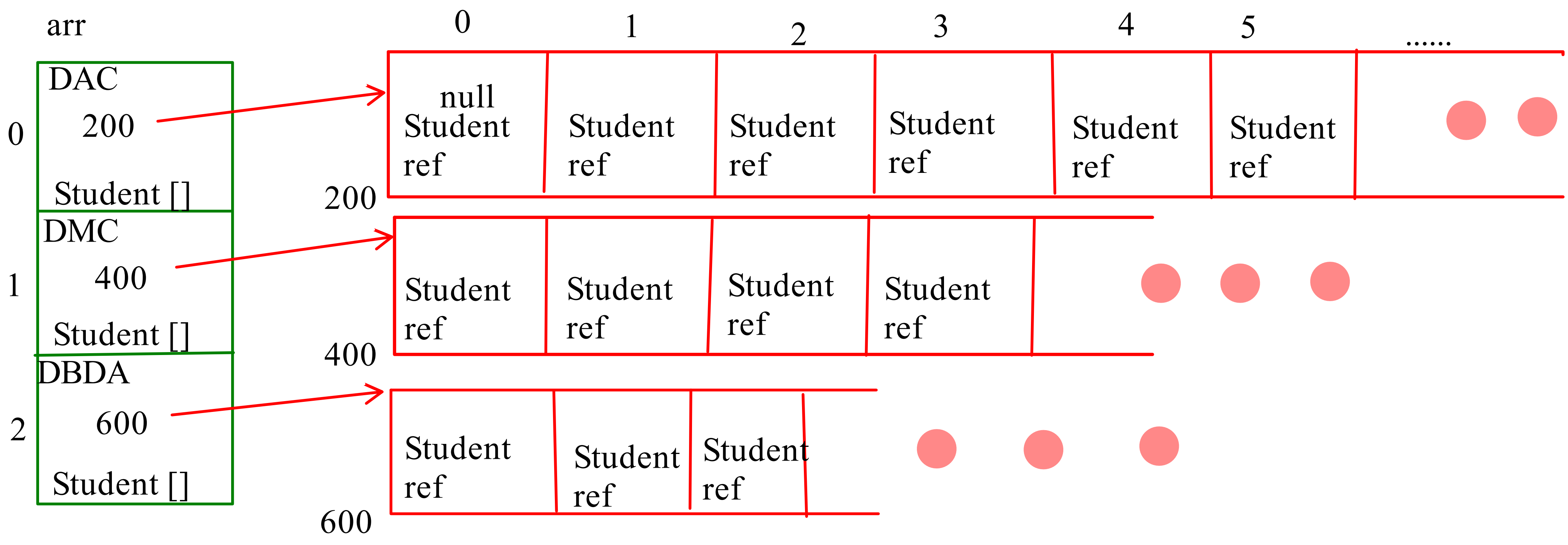
```
int[][] arr;
arr = new int[2][];
// Step to create array on second dimension
arr[0] = new int[2];
arr[1]=new int[2];

arr[0][0] = 10;
```



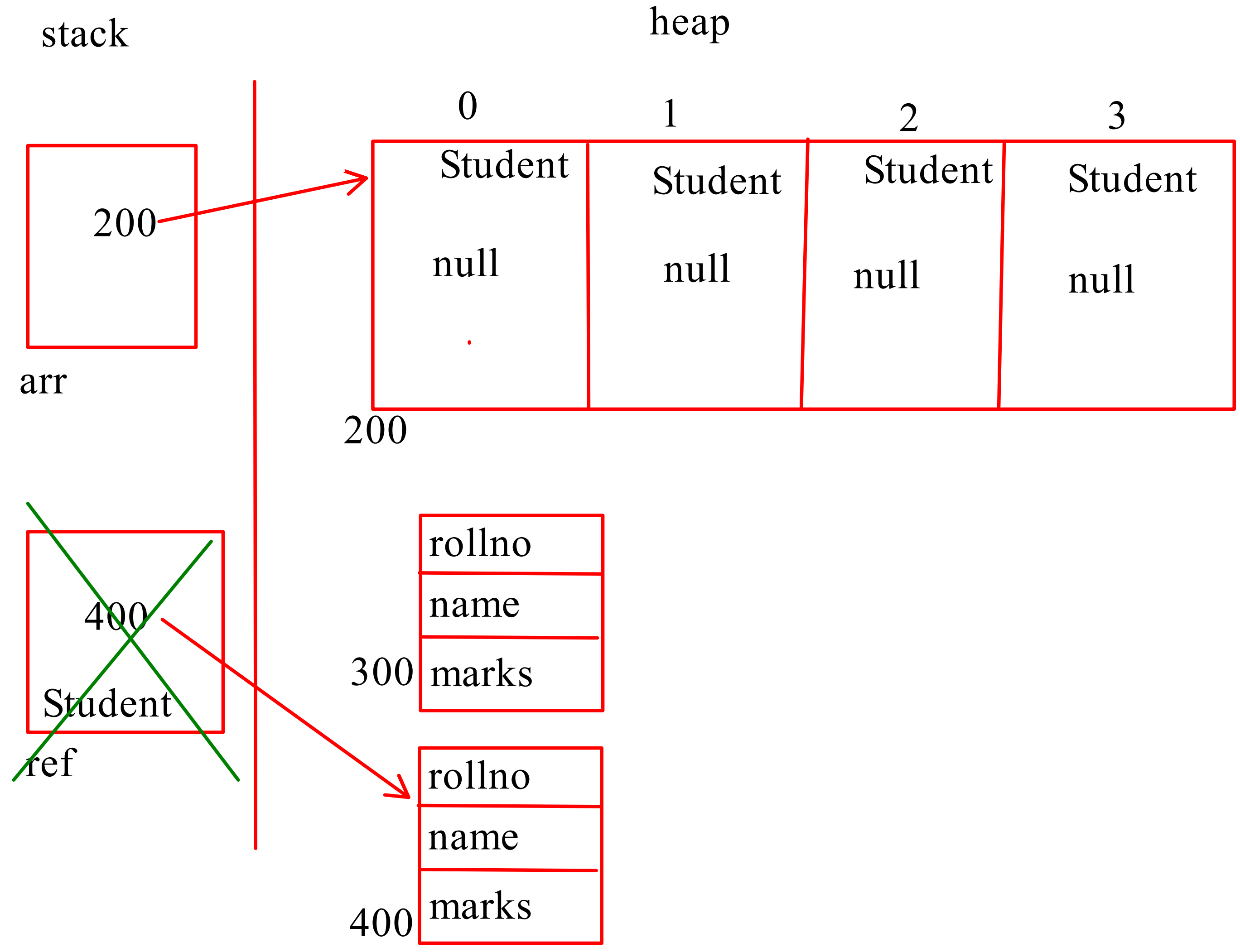


Ragged Array

```
Student [][] arr = new Student[3][];
arr[0] = new Student[240]; // DAC course
arr[1] = new Student[120]; // DMC course
arr[2] = new Student[60]; // DBDA course
```



```
arr[0][0]= new Student();

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

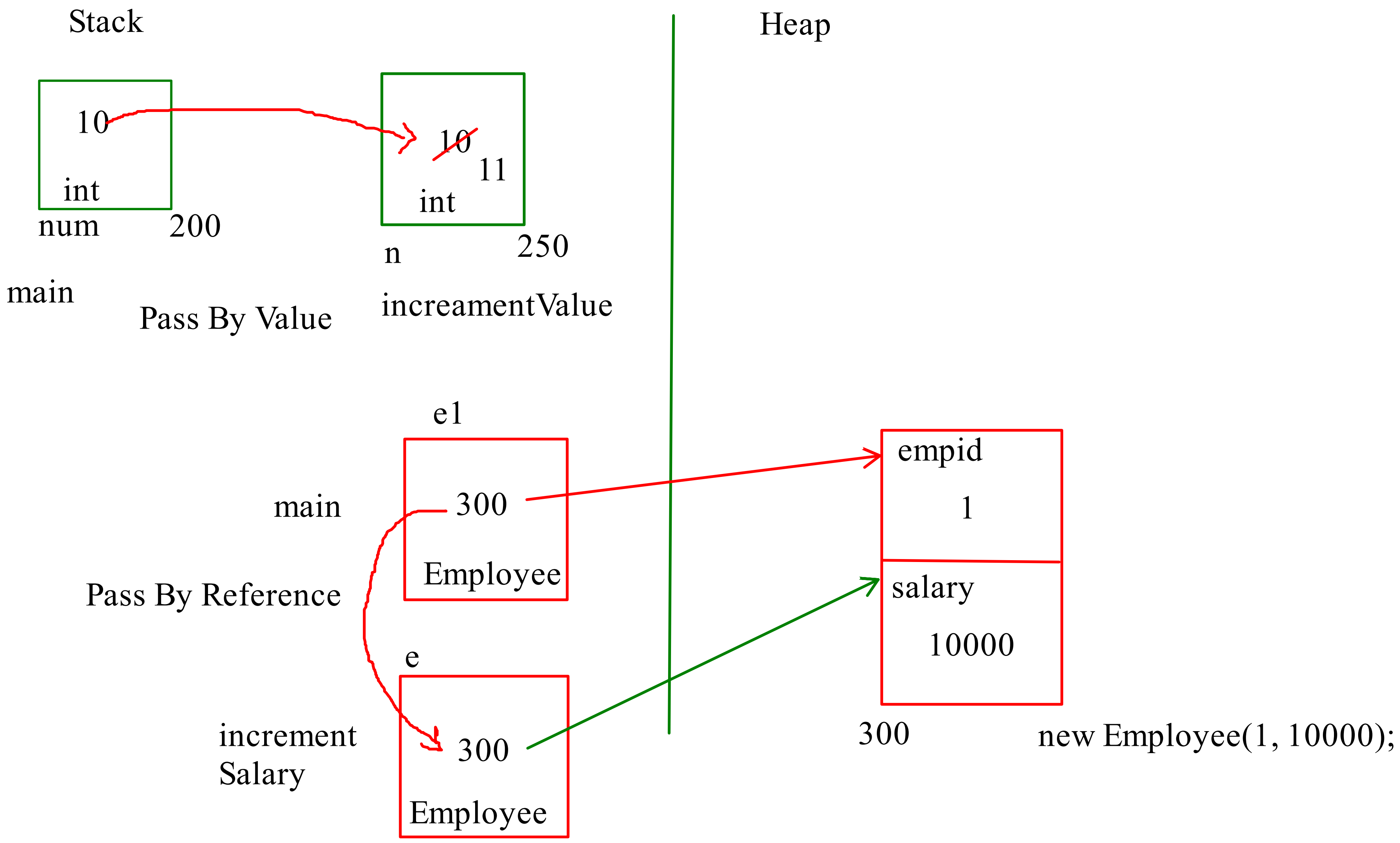


```
Student [] arr = new Student[4];


    for(Student ref : arr){
          ref = new Student();
    }


// arr[0] = new Student();
// arr[1] = new Student();
```

Method Overloading
- Defining the same method multiple time with same name but different signature is called as method overloading
- It can be done by changing the signature of the methods
- The signature can be changed by
    1. changing the no of paramaters
    2. by changing the type of parameters
    3. if no and type of parameters are same then their order of declaration can be changed
- Method overloading is an example of compile time polymorphism

```
void method(dataype parameter){
// B.L
}

method(arguments);
```

Arguments that is passed to a method can be of two types
1. Primitive - pass by value
2. Non Primitive - pass by reference



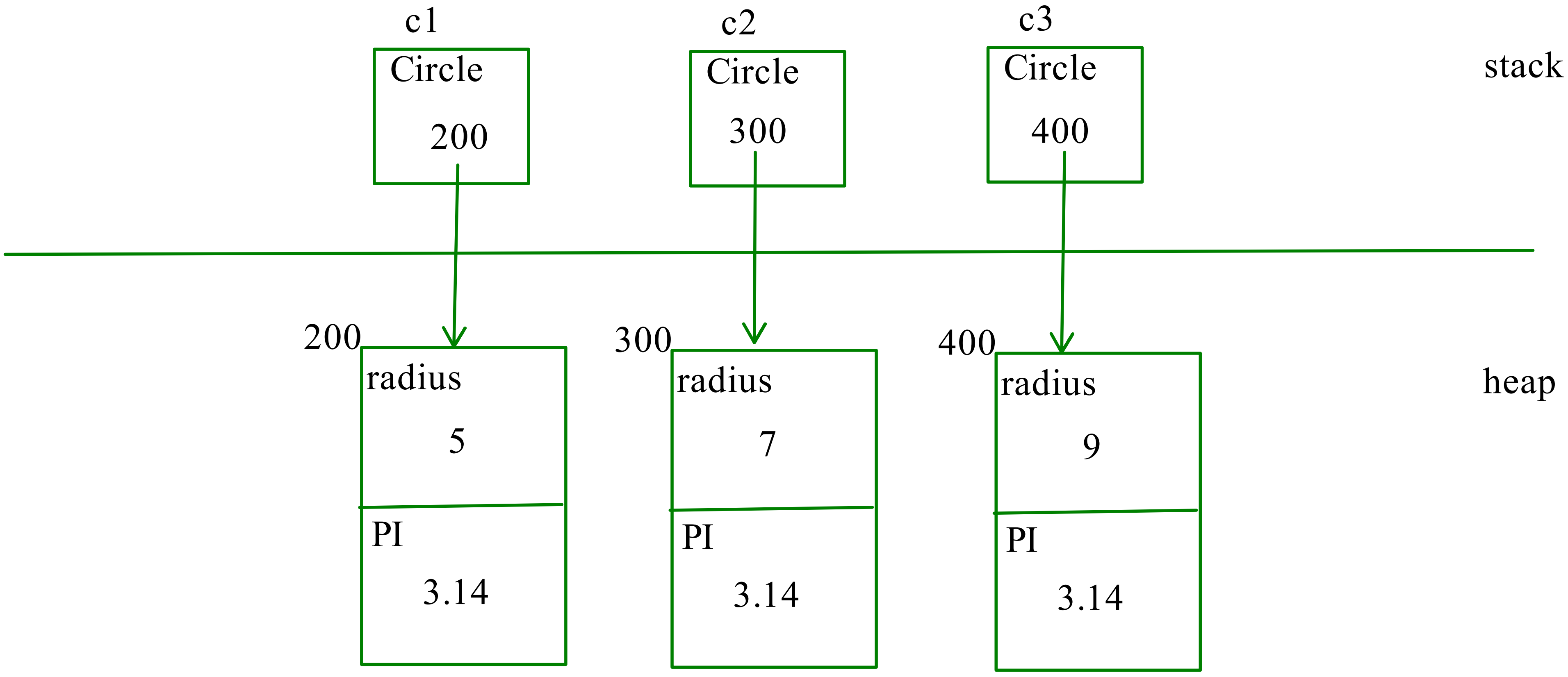
Final
- In java we can make,
1. Varibale as a final
2. Field as a final

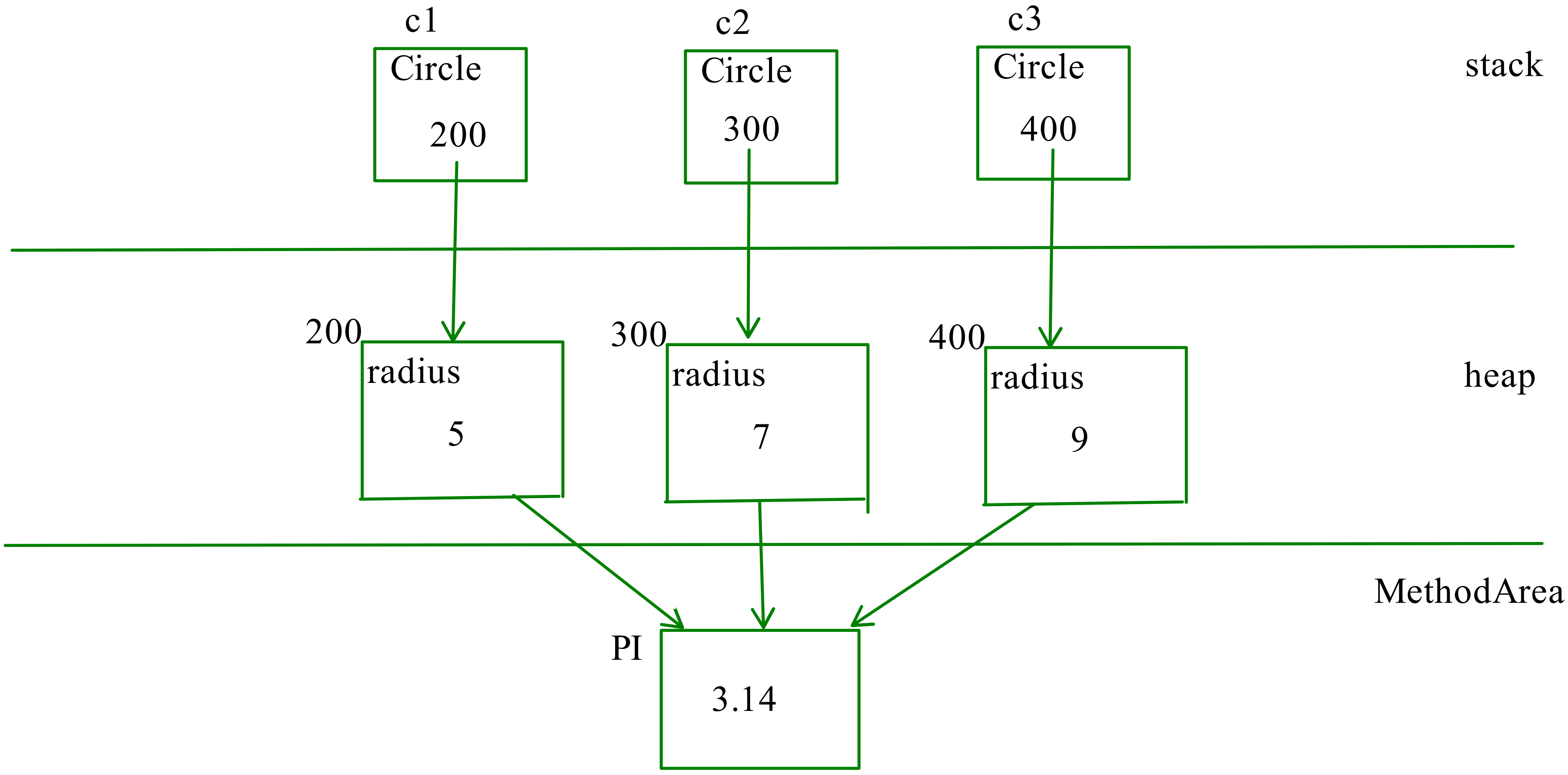
3. method as a final
    - cannot be overriden into the subclass
4. class as a final
    - cannot be extended by the other classes

Final Field
- we can even declare the field of a class as final
- final fields can be initializeed in
    1. field initializer
    2. Object initializer
    3. Constructor
- once initialized it cannot be changed.

Static
- Access modifier which is used for sharing the members.
- static members are considerd as class level members and not object level members
- We can declare members of the class as static
    - field
    - method
- We cannot declare local varaibles as static

Static Field
- Class fields can be declared as static.
- static fields get the memeory on method area only once during classs loading
- static fields are desiged to be shared in multiple objects.


c1
Circle
200
c2
Circle
300
c3
Circle
400
stack
200
radius
5
PI
3.14
300
radius
7
PI
3.14
400
radius
9
PI
3.14
heap


c1
Circle
200
c2
Circle
300
c3
Circle
400
stack
200
radius
5
300
radius
7
400
radius
9
heap
MethodArea
PI
3.14