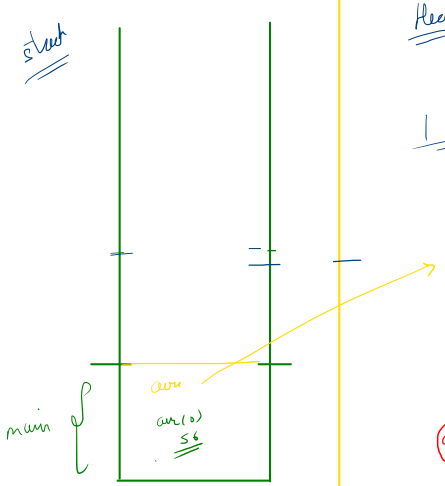
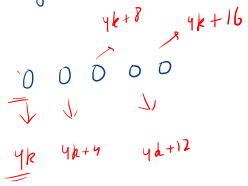
Memory Mapping



Heap

1 int → 4 bytes

20 bytes



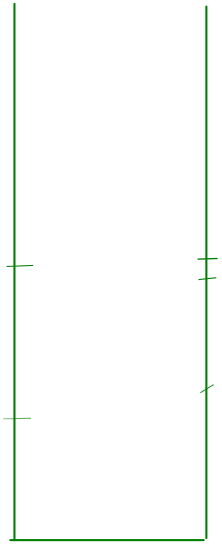
arr → address of 0th variable

arr[idx]

arr[3] = 4k+12

arr[i] → base address + 5*4

```
public static void swap(int a, int b){
        int temp = a;
        a = b;
        b = temp;
}

public static void increaseArray(int[] arr){
        arr[0] = 56;
}

public static void main(String[] args){
        int[] arr = new int[5];
        increaseArray(arr);

        //int a = 2;
        //int b = 5;

        //swap(a,b);
        //System.out.println(a+" "+b);
}
```



garbage - collected

OOPs → Object Oriented Programming

Machine language → assembly languages → BASIC → COBOL

Compiler

Compiler

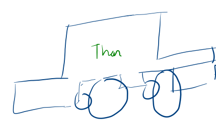
C → purely functional language

C++ → C with OOP features

Java → Object Oriented

Car

The car
- color = blue
- engines → 2L engine
- wheels → alloy wheels
- lights → UV
- seats → 2 seats



customer

1 page

→ real entity

car

blueprints
- lights
- wheels
- color
- seats
- engine

use → Thar

↳ taking some

one

automatic side mirror

↳ (ele___ motors)

automatic side mirrors

blueprint

Thar 1

Thar 2

Thar1

close mirror

⇒ Class → a blueprint

properties / attributes
methods / functions

```
class Car {
    → String color;
    → int seats;
    → String engine;
}
```

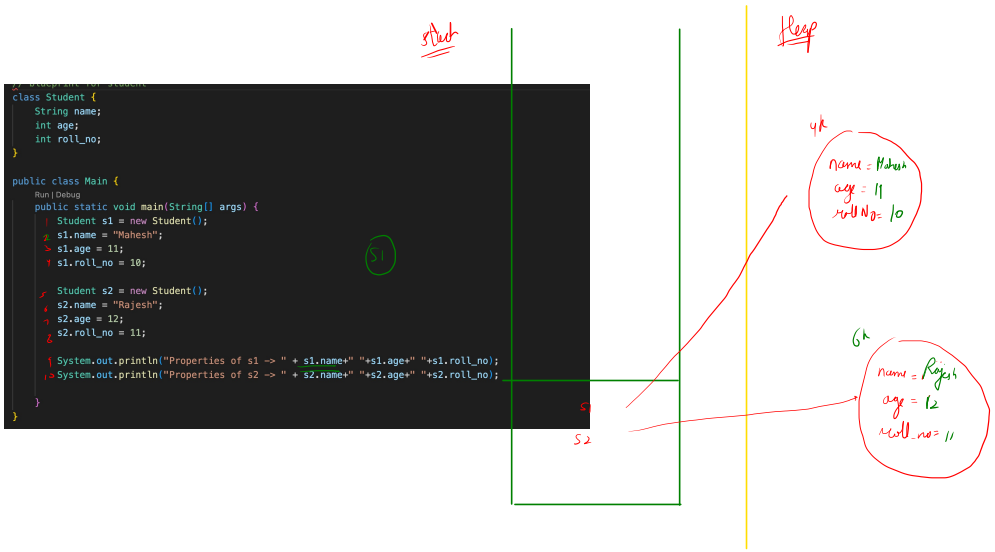
name_of_class name_of_object = new Name_of_class

```
class Student {
    String name;
    int rollNo;
    int age
}
```

Objects
↓
Object

```
Car c1 = new Car();
c1.color = "Red";
c1.seats = 3;
c1.engine = "v4";
System.out.println(c1.color)
```

```
Student s1 = new Student();
s1.name = Mahesh
s1.rollNo = 1
s1.age = 11;

Student s2 = new Student();
s2.name = "Rakesh"
s1.rollNo = 2;
s2.age = 12;
```



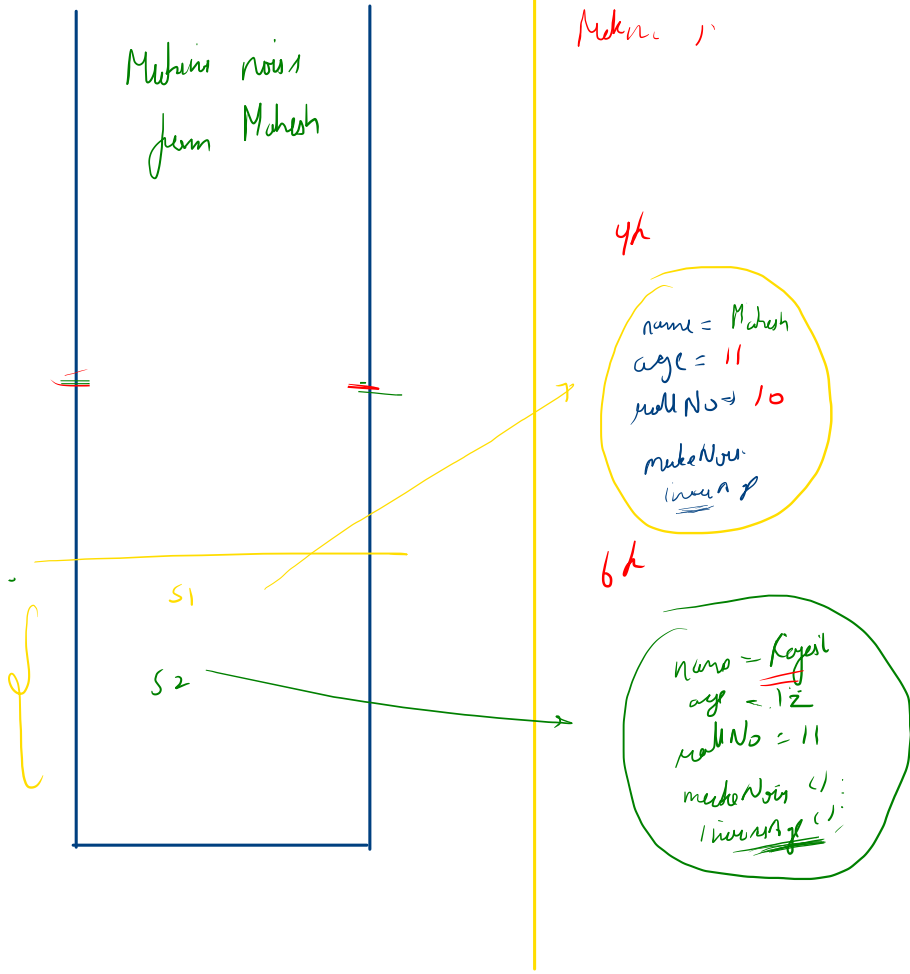
Arraylist <Integer> al = new Arraylist<>();

Arraylist < String > al = new Arraylist <>

Arraylist< E > al =

```java
class Student {
    String name;
    int age;
    int roll_no;

    public void makeNoise(){
        System.out.println("Making noise from "+name);
    }

    public void increaseAge(){
        age++;
    }
}

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Mahesh";
        s1.age = 11;
        s1.roll_no = 10;

        Student s2 = new Student();
        s2.name = "Rajesh";
        s2.age = 12;
        s2.roll_no = 11;

        // System.out.println("Properties of s1 -> " + s1.name+" "+s1.age+" "+s1.roll_no);
        // System.out.println("Properties of s2 -> " + s2.name+" "+s2.age+" "+s2.roll_no);

        s1.makeNoise();
        s2.makeNoise();
    }
}
```

→ Making noise from Rajesh

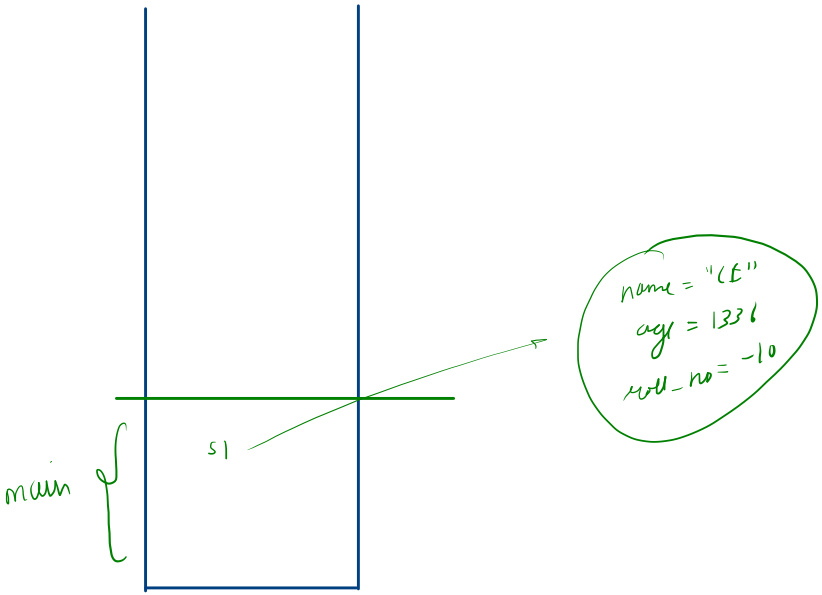


→ DRY

↳ don't repeat yourself.

Constructors

↳ function inside your class with same name as 'Class' and no return type.




```java
class Student {
    String name = "CE";
    int age = 1336;
    int roll_no = -10;
}

public class ConstructorExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
    }
}
```

As soon as we create a constructor, default constructor is deleted.

Types of constructors

default

no input arguments

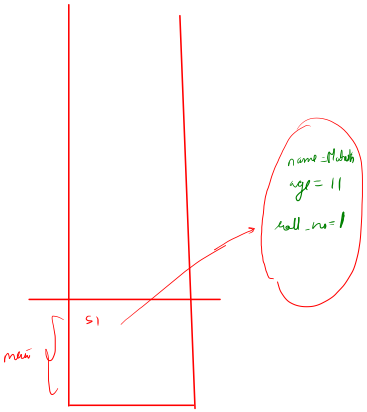
```
public Student() {
    // logic
}
```

Parameterized

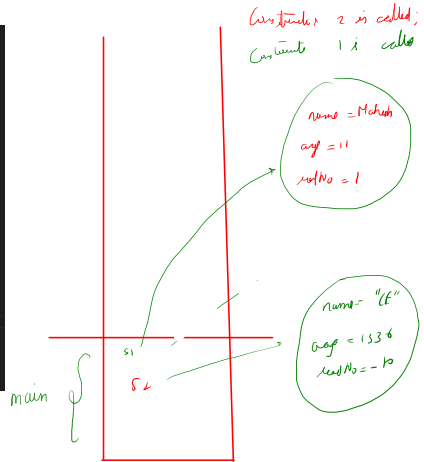
arguments / parameter

```
public Student(_ _ _ _) {

}
```

```java
class Student {
    String name = "CE";
    int age = 1336;
    int roll_no = -10;

    // default constructor
    public Student(){
        System.out.println(x: "My constructor is called");
    }

    // parametrized constructor
    public Student(String newName, int newAge, int newRoll_no){
        name = newName;
        age = newAge;
        roll_no = newRoll_no;
    }
}

public class ConstructorExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student(ne[redacted]e: "Mahesh",n[redacted]: 11,n[redacted]: 1);
        System.out.println("Student Object with properties "+s1.name+" "+s1.age+" "+s1.roll_no);
    }
}
```




```java
class Student {
    String name = "CE";
    int age = 1336;
    int roll_no = -10;

    // default constructor
    public Student(){
        System.out.println(x: "Constructor 1 is called");
    }

    // parametrized constructor
    public Student(String newName, int newAge, int newRoll_no){
        System.out.println(x: "Constructor 2 is called");
        name = newName;
        age = newAge;
        roll_no = newRoll_no;
    }
}

public class ConstructorExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student([redacted]e: "Mahesh",[redacted]: 11,[redacted]: 1);
        Student s2 = new Student();

        System.out.println("Student Object with properties "+s1.name+" "+s1.age+" "+s1.roll_no);
        System.out.println("Student Object with properties "+s2.name+" "+s2.age+" "+s2.roll_no);
    }
}
```



this

Class Student {
String name;
int age;

public void fun (int age) {
age++;
}

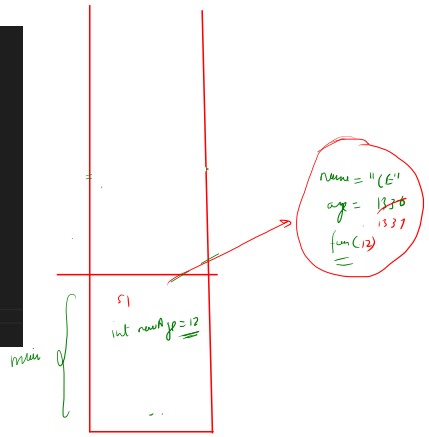
}

Scope of variable

```java
class Student {
    String name = "CE";
    int age = 1336;

    public void fun(int age){
        age++;
        System.out.println(age);
    }
}
public class ThisKeywordExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();

        s1.fun(age: 123);

        System.out.println(s1.age);
    }
}
```

```java
class Student {
    String name = "CE";
    int age = 1336;

    public void fun(int newAge){
        age++;
    }
}
public class ThisKeywordExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();

        s1.fun(newAge: 123);

        System.out.println(s1.age);
    }
}
```

```java
class Student {
    String name = "CE";
    int age = 1336;

    public void fun(int newAge){
        newAge++;
        age++;
    }
}
public class ThisKeywordExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();

        int newAge = 12;
        s1.fun(newAge);

        System.out.println(newAge);
        System.out.println(s1.age);
    }
}
```




```java
class Student {
    String name = "CE";
    int age = 1336;

    public void fun(int age){
        this.age++;
        System.out.println(age);
        System.out.println(this.age);
    }

    public Student(String name, int age){
        this.name = name;
        this.age = age;
    }
}
public class ThisKeywordExample {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student(name: "Accio",age: 23);

        // s1.fun(-100);

        // System.out.println(s1.age);
        System.out.println(s1.name+" "+s1.age);
    }
}
```