Constructors

def: It is a block of code that initialize the newly created object

- constructor has the same name as the class
- constructor doesn't have any return type
- constructor are called during the time of object creation

new Student( ) →

```
class Student {
    Student() {
        ≡
    }
}
```

** Characteristics **

(1) Constructor creation ✓
(2) Constructor overloading ✓
(3) this importance // naming collision can be resolved ✓
(4) Local chaining ✓ { this()
(5) Default Constructor ✓
(6) Apply Access Modifier for constructor ✓

eg:

Student
has →
name
rollno
marks

Note:

(1) constructors & setters are same, but constructor creates/called only once at object creation

(2) If we want to change the values that are created in objects we use the setters

Constructor Overloading

Studentt2 → key: name, marks

validity ↗

(1) no. of parameters
(2) data type
(3) order of parameters

① → Student(){
=
}

② Student(String name){
=
}

③ Student(byte marks){
=
}

④ String(String name, byte marks){
=
}

Note:-

(1) It is same method overloading only difference is we have to use Constructor name multiple times

Local Chaining

def: The process of calling one constructor from another constructor of the same class is called "Local Chaining"

* we can achieve local chaining with help of "this()" call

Rules

→ When ever we make this() call it should be in the first line of your constructor

```java
class Studentt{
    private String name;
    private int rollNum;
    private byte marks;
    Studentt(String name,int rollNum,byte marks){
        this(rollNum,marks);
        System.out.println("Constructor 1");
        this.name = name;
        System.out.println("Name is assigned");
    }

    Studentt(int rollNum,byte marks){
        this(marks);
        System.out.println("Constructor 2");
        this.rollNum  = rollNum;
        System.out.println("rollNum assigned");

    }
    Studentt(byte marks){
        this();
        System.out.println("Constructor 3");
        this.marks = marks;
        System.out.println("marks assigned");
    }
    Studentt(){
    } → Sop("Object Created");
}
```

Order: ① Studentt(), ② Studentt(byte marks), ③ Studentt(int rollNum, byte marks), ④ Studentt(String name, int rollNum, byte marks)

→ new Studentt("Alex", 101, 40)

this() → making call to another constructor

o/p

→ Object Created
Constructor 3
marks assigned
Constructor 2
rollNum assigned
Constructor 1
Name is assigned

Rules continuation

(2) If we do not have any constructor inside a class Then the "Compiler" will insert a constructor which accepts "zero" parameters

(3) The constructor inserted/created by the compiler is known as "default constructor"

Student

↓

Student(){

}

(4) If a constructor of some parameter is created {} if we try to create a object without any parameter it results in error because compiler will ignore creating default constructor as soon it see some user defined constructor is available

(5) If a constructor is made as private then we can create object of the class only within the class not outside class