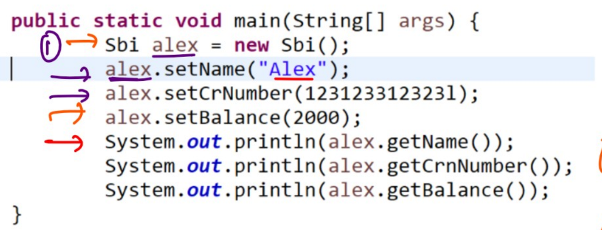
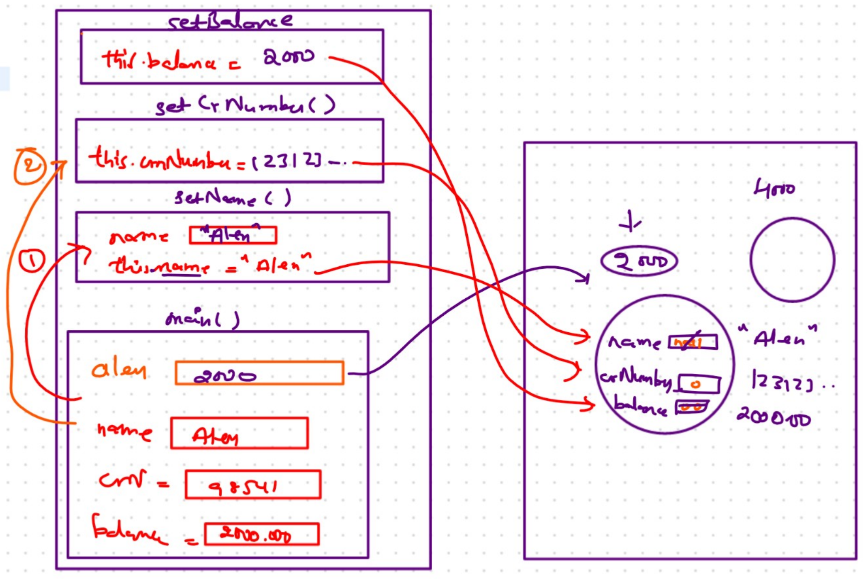
Oop . Encapsulation

– Continuation

Memory mapping



```
public static void main(String[] args) {
    Sbi alex = new Sbi();
    alex.setName("Alex");
    alex.setCrNumber(1231233123231);
    alex.setBalance(2000);
    System.out.println(alex.getName());
    System.out.println(alex.getCrnNumber());
    System.out.println(alex.getBalance());
}
```



sbi bob = new sbi( )

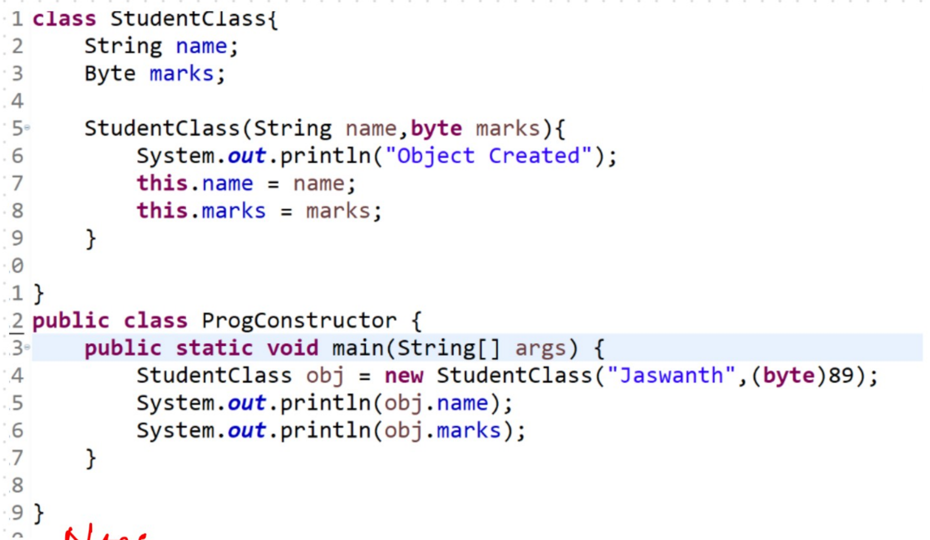
bob.setName("bob")

creates automatically a new stack region when method called

Constructor

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

* Constructor has the same name as the class
* Constructors doesn't have any return Type
* constructors are called during the time of object creation



```
class StudentClass{
    String name;
    Byte marks;

    StudentClass(String name,byte marks){
        System.out.println("Object Created");
        this.name = name;
        this.marks = marks;
    }

}
public class ProgConstructor {
    public static void main(String[] args) {
        StudentClass obj = new StudentClass("Jaswanth",(byte)89);
        System.out.println(obj.name);
        System.out.println(obj.marks);
    }

}
```

Note:

(1) constructors & setters are same but constructors create at one instance,

(2) if we want to change the values that created in object, we use setters

Constructor Overloading:

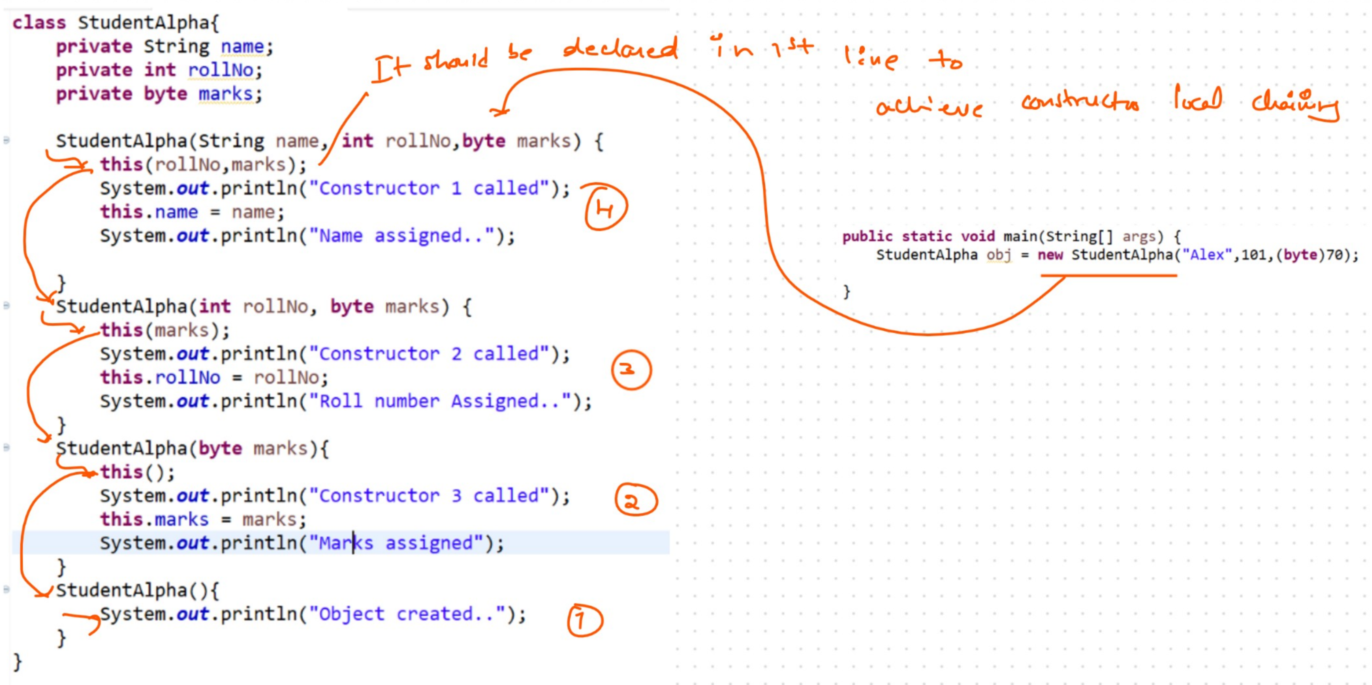
It is same method overloading having more than one constructor in a class following

(1) empty params

(2) different no. of params

(3) order of params

Local Chaining



```
class StudentAlpha{
    private String name;
    private int rollNo;
    private byte marks;

    StudentAlpha(String name, int rollNo,byte marks) {
        this(rollNo,marks);
        System.out.println("Constructor 1 called");
        this.name = name;
        System.out.println("Name assigned..");

    }
    StudentAlpha(int rollNo, byte marks) {
        this(marks);
        System.out.println("Constructor 2 called");
        this.rollNo = rollNo;
        System.out.println("Roll number Assigned..");
    }
    StudentAlpha(byte marks){
        this();
        System.out.println("Constructor 3 called");
        this.marks = marks;
        System.out.println("Marks assigned");
    }
    StudentAlpha(){
        System.out.println("Object created..");
    }
}
```

```
public static void main(String[] args) {
    StudentAlpha obj = new StudentAlpha("Alex",101,(byte)70);

}
```

It should be declared in 1st line to achieve constructor local chaining

Notes

→ 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

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

→ If we do not have any constructor inside a class then the "compiler" insert a constructor which accepts "zero parameters" while converting Java file to class file

→ the constructor inserted by the compiler is known as "default constructor"

Note:

(1) If a constructor of some parameters is created & if we try to create a default parameter object we get a error because default constructor won't be created by compiler since we created our own constructor

(2) We can have private, protected, default, public modifiers for constructors

(3) If constructor is made as private then we can creat object of that class only within the class not outside the class

Constructor Creation

Constructor Overloading

this importance [naming collision can be resolved]

local chaining & this ()

default constructor

access modifier for constructor [public & private]