(22) Encapsulation, Constructor, this Vs this()

↳ Hyder Abbas
↳ 9th Nov class

Agenda

1. This Keyword
2. 1-Program Encapsulation
3. Constructor
4. This() Method
5. This Vs This()

length - Keyword ↳ always
length() - Method ↳ String

→ There are 4 access specifiers in Java
- (i) Public - Entire package this Method can access.
- (ii) Protected
- (iii) default
- (iv) Private

→ getter Syntax : 'get' word followed Variablename

Eg:- getAge()

→ it follows Camel Care.

Recommandations :-

1. When Ever getter is returning boolean value then it is recommended to use 'is' followed by property.

Eg:-
```
Class student {
    Private int Age;
    Private String Name;
    Private boolean Married;

    // getter
    Public boolean isMarried()
    {
        return married;
    }
}
```

→ instead of 'get' use 'is', it is highly recommended for boolean return type.

→ Whenever there is naming Conflict between local Variable and instance Variable with in a setter is called Shadowing phenamenon.

Eg:
```
Class student {
    Private int age;
    Void setAge(int age) {      // local Variable
        age = age;              // instance Variable
    }
}
```

→ JVM at run time will not give Value

① This :-

→ In order resolve this issue we use 'this' keyword.

Eg:-

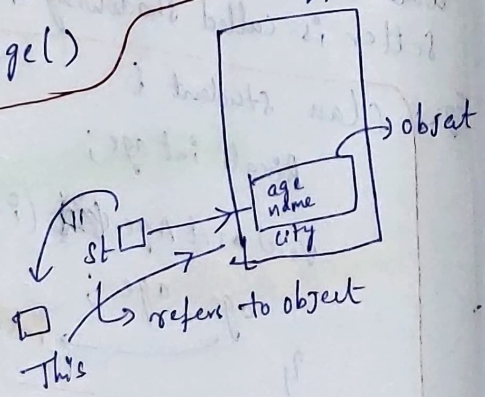
```
class Student {
    Private int age;

    // Setter
    Void SetAge(int age) {
        this.age = age;
    }
}
```

→ This refering to Currently running object.

→ we can Create multiple objects of one class.

```
Student st = new Student()
st.SetAge()
```



Eg:-

```
① Class Student {
    Private int age;

    //setter                          → local variable
    Void SetAge(int age) {
        this.age = age;   ②
    }
}
Public Class Encap {
    Public static Void main(String[] args) {
        Student st = new Student();
        st.SetAge(20);
    }
}
```

↳ Value of local Variable given to instance Variable

(i) st.setAge() Method is executed for st object

(ii) 'this' Keyword holds the address of st then this keyword also refering to same address.

(iii) 'This' Keyword refers to Currently running object

(iv) local Variables Values are given to instance Variable.

(v) 'This' keyword holds the address of object

GADDALA ASHISH
Linkedin

Eg ① :-

```
class Student {
    private int age;
    private String city;

    // getters and setters.
    public void setAge(int age){
        this.age = age;        → ⓐ
    }
    public int getAge() {
        Return age;
    }
    public void setCity (String city) {
        this.city = city;      — ⓑ
    }
    public String getCity() {
        return city
    }
}
```

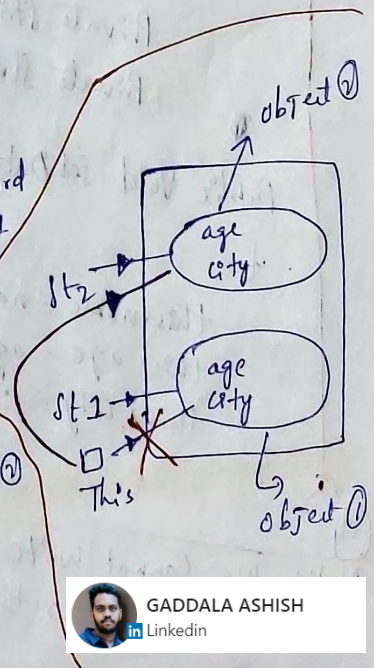
```
public class Encap {
    public static void (String[] args) {

object①⇒ Student st1 = new Student();
        st1.setAge(28)          — ①
        int age = st1.getAge();
        s.o.p(age);

object②⇒ Student st2 = new student();
        st2.setCity("Hyd");     — ②
        String city = st2.getCity();
        s.o.p(city);
    }
}
```

① After line ① 'this' keyword hold the address of st1

② When it comes @ line ① 'This' keyword holds Address of st2 Means it refers to object-②



GADDALA ASHISH
Linkedin

② Common Setter :-

Eg: class Student {

```
Private string name;
Private int age;
Private String City;

Public Void setData(string name, int age, String City)
{
this.name = name;
this.age = age;
this.City = City;
}
Public string getName(){
    return name;
}
Public int getAge(){
    return age;
}
Public int getCity(){
    return City;
}
}
Public class Encap{
    Public static Void main(String[] args){
Student std = new Student();
Std.setData("ashish", 23, "hyderabad");
S.O.P(std.getName());
S.O.P(std.getAge());
S.O.P(std.getCity());
}
```

Common setter

GADDALA ASHISH
Linkedin

③ Constructor :-

a Constructor is a block of code Similar to the method. it is Called when a instance of class is created.

→ Constructor name must be the same as its class name.

→ A Constructor must have no Explicit return type.

→ A Java Constructor Cannot be abstract, Static, final, and Synchronized.

→ Constructor is a kind of Method whose name is same that of class name.

→ Constructor is automatically invoked (or Called the Moment object is Created.

→ While Creating object arguments are Passed.

→ Whenever something should Executed at the time of instantiation (at the time of object Creation) keep that Particular thing in Constructor.

(1) Constructor is Executed When object is Created.

(2) In Case of Parameter whatever rules hold for Method it is also hold for Constructor.

Code :-

```
Class Student{
    Private String name;
    Private int age;
  // Constructor ↓
    Public Student(String name, int age){
        this.age = age;
        this.name = name;
    }
    Public int getAge(){
        return age;
    }
    Public String getName(){
        return name;
    }
}
Public Class Encap{
    Public static void main(String[] args){

        Student std = new Student("Ashish", 23);
                    ↳ The Moment object is created Constructor is Called (or invoked.

        S.o.p(std1.getName());
        S.o.p(std1.getAge());
    }
}
```

GADDALA ASHISH
Linkedin

important points

(3) Constructor will not have return type.

(4) no need Constructor explicitly.

(5) Constructor can have zero parameters

(6) if we call a Method which is not in class it shows Error

(7) if we call a Constructor which is not in class in that JVM (or) Java Compiler will include default Constructor behind the scene.

↳ since it is there that's the reason no Error is shown.

Eg:-

```
Class student {
    Private int age;
    Private String name;
}

Public class Encap {
    public s v M(String[] args) {
        Student std1 = new student();
    }
}
```

JVM include default Constructor behind the scene

↳ Constructor is called (or) invoked

(8) JVM will include default Constructor with zero parameters

Eg:-

```
Class Student {
    Private String name;
    Private int age;
}

Public class Encap {
    Public static void main (String[] args) {
        Student std = new student("Ashish", 23);
    }
}
```

↳ it give Error JVM include default Constructor with zero Parameters

Error

(9) if Programmer have Specified a Constructor irrespective of no. of Parameters. then JVM will not include any default Constructor.

→ if we called zero parameter Constructor it gives Error in this case.

```
Class student {
    Private String name;
    Private int age;

    Public student(String name, int age) {
        This.age = age;
        This.name = name;
    }
}
```

```
Public class Encap {
    P s V M(String[] args)
    {
        student std = new student
            ("Ashish", 23);

        Student std1 =
            new student();
    }
}
```

↳ it gives Error

GADDALA ASHISH
Linkedin

Constructor Overloading

→ Same name different Parameters

→ The Constructor overloading can be difined as the Concept of having More than one Constructor with different Parameters.

Eg:

Eg1

```
class Student{
    Private String name;
    Private int age;

Constructor ① ⇒ Student(String name, int age)   → Constructor with parameters
{
    this.name = name;
    this.age = age;
}

Constructor ② ⇒ Student()   → Constructor without parameters
{
    name = "Naveen";
    age = 26;
}}

Public class Encap{
    Public static void main(String[] args){
        Student std1 = new Student("Ashish", 23);   ↳ calling Constructor-①
        Student std2 = new Student();               ↳ calling Constructor-②
}
```

→ two Constructor with same parameters are not allowed in class.

→ Another Constructor with same parameter are not allowed in class.

→ Constructor: the Constructor who Constructs the object.

⑤ Super() Method

→ In Every Constructor behind the Seceans one Call of Method is made which is Called Super() method.

→ Super() Method Calls Parent class Constructor.

→ In Java if explicitly if we not specify any Parent. then the parent is object

→ object class

→ Super() Method :- it checks the order of Parameters

→ JVM includes Super() Method

→ if want we can write Super Method.

GADDALA ASHISH
Linkedin

Code Eg: Super() Method

```
Class Student {
    Public String name;
    Public int age;
    Public Student(String name, int age)
    {
(1) ⇒   super();
        this.name = name;
        this.age;
    }
}
```

behind Seceng JVM will include Super Method after Creating Constructor

⇒ In one Case Super Method will not be there.

↳ if there is this() Method.

↳ super(); this();

↳ Whatever write after semicolon Consider 2nd one

⇒ both Methods Should be 1st line when they written.

→ & only one Method is allowe to write.

→ if we don't write any then Super Method is included in 1st line by JVM

⇒ Super Method Calls Parent class Constructor.

(6) This() Method :-

(i) this() is used to call one Constructor from the other of Same Class.

(ii) this keyword is used to refer to the Current object. through which Method is called.

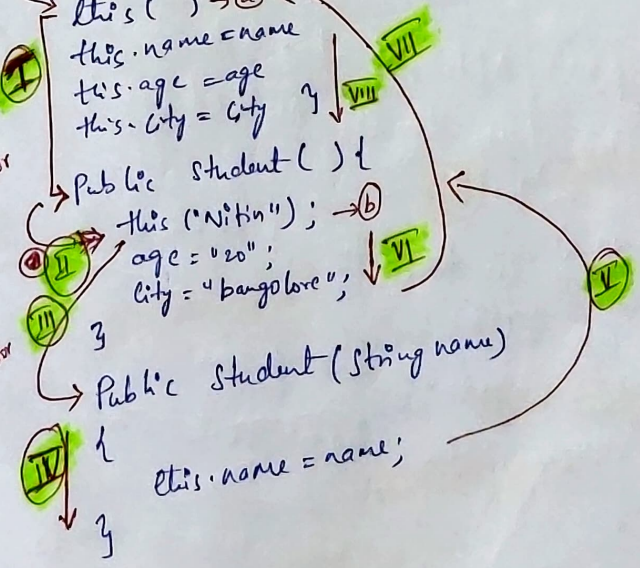
	default	1st	2nd
Name	null	Nithin	Ashish
Age	0	20	23
City	null	bangalore	Hyderabad

Code :-

```
Class Student {
Private String name;
Private int age;
Private String City;
Public Student(String name, int age, String City) {
    this();
    this.name = name
    this.age = age
    this.City = City
}
Public Student() {
    this("Nitin");
    age = "20";
    City = "bangalore";
}
Public Student(String name)
{
    this.name = name;
}
}
```

Ashish 23 Hyderabad

Constructor 1 — Constructor 2 — Constructor 3



Continuation Code ↓

GADDALA ASHISH
Linkedin

→ this() Method calls same class Constructor

```
Public class Encap{
    Public Static Void Main(String[] args){

        // Calling Constructor - I
        Student Std1 = new Student("ashish", 23, "Hyd");

        // Calling Constructor - II
        Student Std2 = new Student( )

        // Calling Constructor - III
        Student std3 = new Student("within");

    }
}
```

At ⓐ :- Arguments are given for Constructor (1)

① this() Methods Calls Same class Constructor which is zero parameterised.

② here, Control goes to Constructor (2) (refer (1))

Code :

```
Public Student(String name, int age, String city){
    this( )                          → ⓐ
    this.name = name;
    this.age = age;
    this.city = city;
}
Public Student( ){
    this("Within");                  → ⓑ
    age = 28;
    City = "bangolore";
}
```

At ⓑ :- Control comes to ⓑ from Constructor (2)

→ from this Method ⓑ it checks for Constructor with single parameterised (String as input)

→ then Control goes to Constructor (3)

→ name is given to Constructor - (3) as argument (see step (3))

→ The block under Constructor - (3) get executed.

GADDALA ASHISH
Linkedin

Step-IV Code under Construction get Executed.

Step-V After Code Execution Control again goes to this ("Within") Method.

Step-VI :- Code under this Method (b) get Executed.

```
Public student ( ) {
    This ("Within"); -(b)
    age = 28;
    City = "bangolore";
}
```

(VI)

Step-VII :- After Executing Code at Step IV Control goes this ( ) Method (a) in Constructor (1)

Step-VIII :-

→ Code under this ( ) Method (a) get Executed.

→ here arguments are given to Constructor during the object Creation.

→ This process we call as Constructor chaining.

(7) Constructor Chaining in JAVA

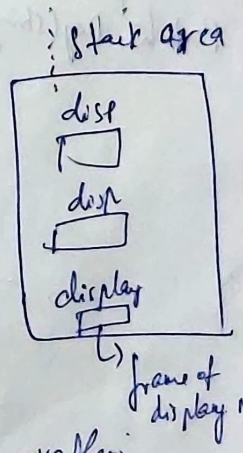
Constructor is sequence of invoking Constructors upon initializing an object. it is used when we want to invoke a number of Constructors, one after another by using only an instance.

(8) Stack overflow :-

A stack overflow is an undesirable condition in which a particular Computer program tries to use Memory Space than Call Stack has available.

→ In programming, the Call Stack is a buffer that stores request that need to be handled.

```
disp( ) {
    .
    .
    disp( );
}
```



frame of display Method

→ Once capality filled it will overflow.

GADDALA ASHISH
Linkedin

→ ~~the Moment~~

→ the logic responsible for Connected Java to Database is given to Constructor.

⇒ the moment we Create object data base will be Connected.

→ Inside Constructor there is need of another Constructor where we this() Method and Specify Paramets. (Constructor Chaining Example).

Eg

```
class Dog {

  Public String name;
  Public int Cost;

  Public Dog( ) {
    This ("sheru", "9999");
  }

  Public Dog(String name, int cost)
  {
    this.name = name;
    this.cost = Cost;
  }
}
```

```
Public Class Exeap {
  Public S V Main(Str Arg)
  {
    Dog d1 = new Dog( );

    Dog d2 = new Dog("robo", 10000)
  }
}
```

(Arrows numbered 1–8 show the order: `Dog d1 = new Dog( );` goes to `Public Dog( )`, then `This("sheru","9999");` goes to `Public Dog(String name, int cost)`, then to `this.name = name; this.cost = Cost;`, and back to `d1`; `Dog d2 = new Dog("robo",10000)` goes to `Public Dog(String name, int cost)`.)

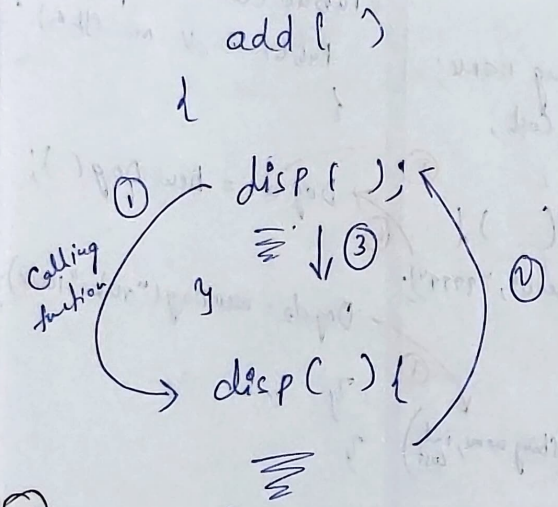
	Initial	d1	d2
name	null	Sheru	robo
cost	0	9999	1000

→ Inside a Constructor there is need to Call another Constructor we have to this() Method and Specify no. of Parameters.

GADDALA ASHISH
in Linkedin

④ Method chaining

→ We Cannot Create Inside a Method but we Can Call Method which is Created.



⑩ # Conclusion:

Constructor: ① Same name as Class name

② Constructor Parameters work like Method Parameters.

③ Constructor Called object Created (or) instantiated.

④ return statement invalid. it will not have return type explicity

⑤ Inside Constructor first statement is Super( ) (or) this( ) Method

→ Super( ) Method Calls parent Constructor

→ This( ) Calls Constructor of Same class

⑥ We Cannot have Both this( ) & Super( ) Method In Same Constructor.

⑦ We Cannot write this( ) (or) Super( ) Method apart from line.

→ we have to write in first line.

⑧ Purpose:- When Constructor is Called?

→ whenever object is Created (or) Initantiated Constructor is Called.

→ to Execute piece of Code the Moment we Create object of Class

→ if Some statement has to be Executed the Moment we Create a object.

→ we Can write Inside Constructor

⑨ this( ) :- Inside a one Constructor requirement to Call another body.

⑩ this( ) Method Calls Same class Constructor.

GADDALA ASHISH
Linkedin

(11) Difference between This and This()

This	This()
keyword ↓ refers to Current object	Method Call ↓ → it will call Same Class Constructor → it refers to the Constructor of the Same Class whose parameters matches with the Parameters passed to this (Parameters.

(12) Use Constructor Overloading

We can have More than one Constructor with different parameters..

(13) Difference between Method and Constructor

Method	Constructor
① Call explicity by Calling name.	① When object is Created Constructor is Called.
② it will have written type explicitly (Void, int, String)	② no Explicit return type.
③ return statement is allowed	③ no return Statement.

GADDALA ASHISH
Linkedin