03 Access Modifiers and Constructors

26 May 2024 05:51

Agenda :

1. Access Modifiers
2. Constructors
    a. Default
    b. Manual
    c. Copy
3. Deep Copy vs Shallow Copy
4. Value Vs Pass by Reference
5. Destructor

1. Access Modifiers

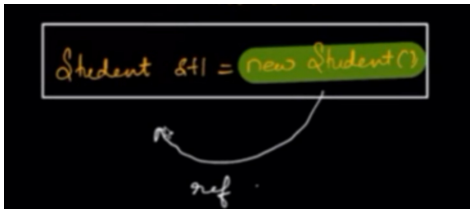
- Class will store attributes and methods together.
- Class should protect attributes and methods from any illegitimate access from outside.
- **Access Modifiers are the one which helps in controlling these attributes and method**s.

Types Of Access Modifiers :

1. **Public** : Can be accessed from anywhere.
2. **Default** : Can be accessed from only the same package.
3. **Protected** : Can be accessed from same package + any child class from another package.
4. **Private** : Can only be accessed from methods within the same class.

2. Constructors

```
Student {
        String name;
        Int age;
        Double psp;
        String userName;
}
```



- **new** creates a new object**.**
- **Student()** creates a reference to Student class using the default constructor.

Default Constructor

- If we don't create our own constructor of a class, a default constructor will be created.
- The default constructor creates a new object that sets the value os each attribute to default value of that data type.
  - **Takes no parameter and has no return type.**
  - **Same as class name**
  - **Created only when we don't have manual constructor.**

```
Student (){
        Name = null;
        Age = 0;
        Psp = 0.0;
        userName = null;
}
```

Manual Constructor

```
Student (String name, int age, double psp, String userName){
        this.name = name;
        this.age = age;
        this.psp = psp;
        this.userName = userName;
}

Student s1 = new Student("Vishal", 26, 92, "mahotpal1");
```

- Manual Constructors are created by user.
- **Initializes the object with the default values of every attribute.**
- **When constructor is called whatever values is passed from constructor is set.**
-

Copy Constructor

```
Student s1 = new Student("Vishal", 26, 92, "mahotpal1");
Student s2 = s1;

S2 and s1 points to the same memory location.
```

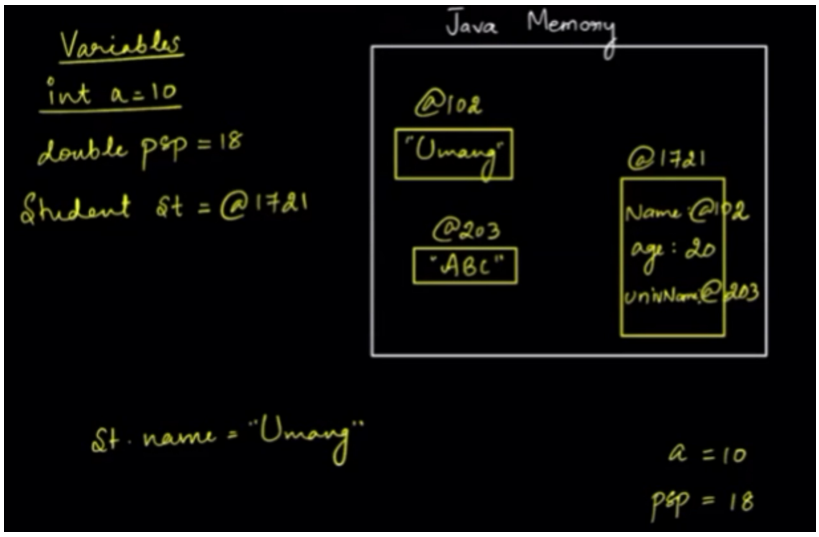
- **If we want to create a new object with the exact same values as older reference object is holding we use Copy Constructor.**

  Copy Constructor :

```
Student(Student s_ref){
        this.name = s_ref.name;
        this.age = s_ref.age;
        this.psp = s_ref.psp;
        this.userName = s_ref.userName;
}
```

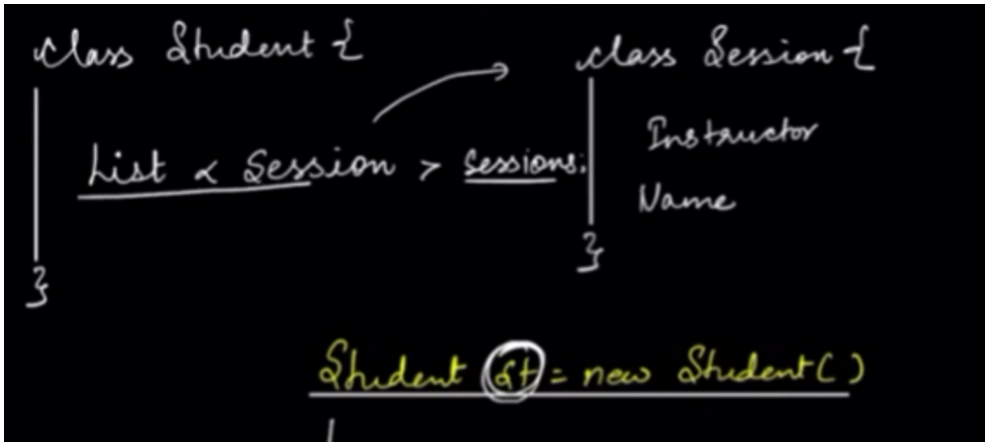
3. Deep Copy Vs Shallow Copy

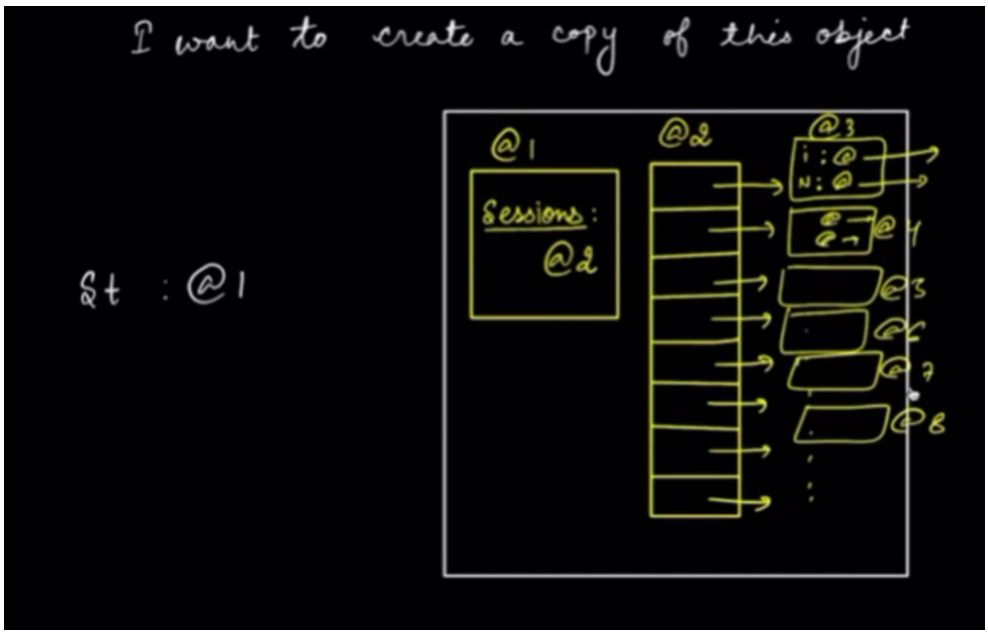
1. Primitive Types (int, float, double) : As variable itself is directly stored in the memory.
2. Objects : Objects are stored in memory variables just store reference/address to the object.



- **Deep Copy** is a copy of existing object which has a separate reference.
  Student s2 = new Student(s1);
- **Shallow copy** is a copy of existing object with same reference as of older object reference.
  Student s2 = s1;

Scenario For deep copy for complex object :





It's very difficult to create a deep copy for these kinds of scenario.

4. Pass By Value vs Pass By Reference

- When only value of the variable is passed, so that changing its value doesn't affect the original value. It is called pass by value.
- When reference of the variable is passed, so that changing its value affects the original one. It is known as pass by reference.

- Java is always pass by value.

5. Destructor

- Destructor is automatically called by java. We don't need to explicitly define destructor.
- **Garbage Collector** is the one which is responsible for cleaning and destroying all the objects in last.