Agenda → ① Access Modifiers
② constructors → Default
→ Manual / Parameter-ized
→ Copy → Shallow
→ Deep
↓
Destructor
↳ doesn't help a lot in OOP

⇒ Is Java pass by value or pass by reference ?

principle → Abstraction
Pillar → Encapsulation

① holding data & functions together
⇒ class

② protecting members from illegitimate access ⇒ Access Modifiers
(data + functions)

① Access Modifiers

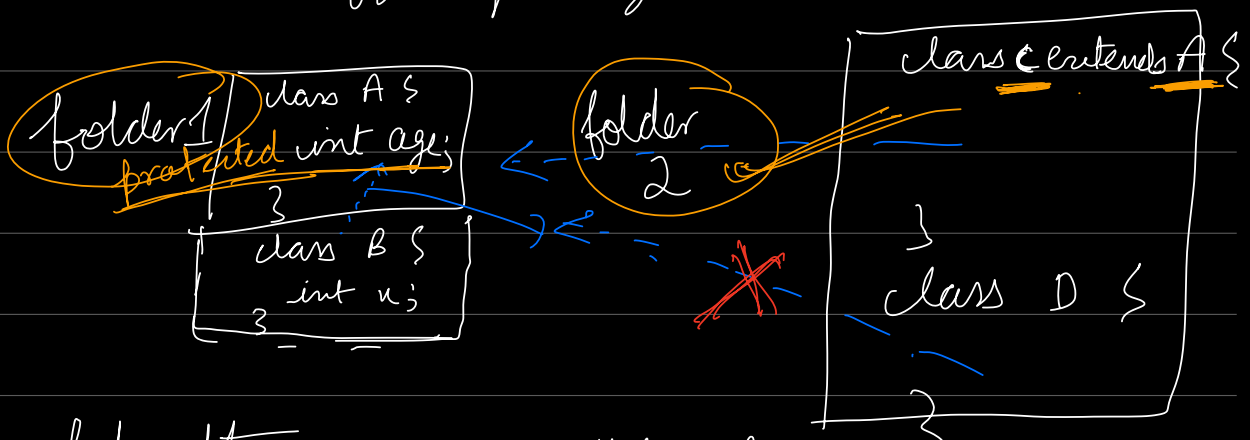
↳ 4 types of access modifiers

① public ② private ③ protected ④ default

public → member → accessible from anywhere

private → nobody from outside the class ⇒ it is still accessible within the class

protected → access it from the same package and also from child class in diff. package



default → accessible from within the same package.

anywhere

	same class	any other class in same package	child class in same pkg	child class in diff. package	anywhere
private	✓	✗	✗	✗	✗
default	✓	✓	✓	✗	✗
protected	✓	✓	✓	✓	✗

public

(2) Class — blueprint of an idea
object ⟶ turning your class into a reality

```
Student {
    string name;
    int age;
}
```

```
Student st = new Student();
```

```
int a = 5
```

Datatype variable name — default Constructor

```
Student ();
A ();
```

⇒ If we dont create our own constructor java provides us with with default constru

~ctor.

⇒ It creates a new object & then it initializes all attrs. with their default values

```
Student () {
    age = 0
    name = null
}
```

→ Same name as class
→ no return type
→ can have all types of access modifiers

Default Constructor → ① public
② only available when I/E dont create our own constructor
③ all attrs are initialized with default value for data type

```
Student () {
    name = "ABC";
}
```

```
Student st = new Student();
st.name = "Saurabh";
```

(2)

```
Student {
    int age;
    String name;
    Student(int studentAge, String studentName) {
        age = studentAge;
        name = studentName;
    }
}
```

~~Student st = new Student();~~ ✗

= new Student(21, "ABC");

```
private Student(int studentAge, String studentName) {
    age = studentAge;
    name = studentName;
}
```

If the parameterized constructor is private, default constructor becomes unavailable

Break till ~~10:36~~ ~~10:3~~ 10:40

Student st1 = new Student();

st2 ⟹ new object with same attrs as st1

st1 | age: 21 | name: ABC

st2 | age: 21 | name: ABC

Student st2 = new Student(st1);

Copy constructor

```
Student(Student st) {
    age = st.age
    name = st.name
}
```

⟹ st1, st2, st3, st4

```
Student st = new Student( );
st.name = "ABC";
st.age = 21;
```

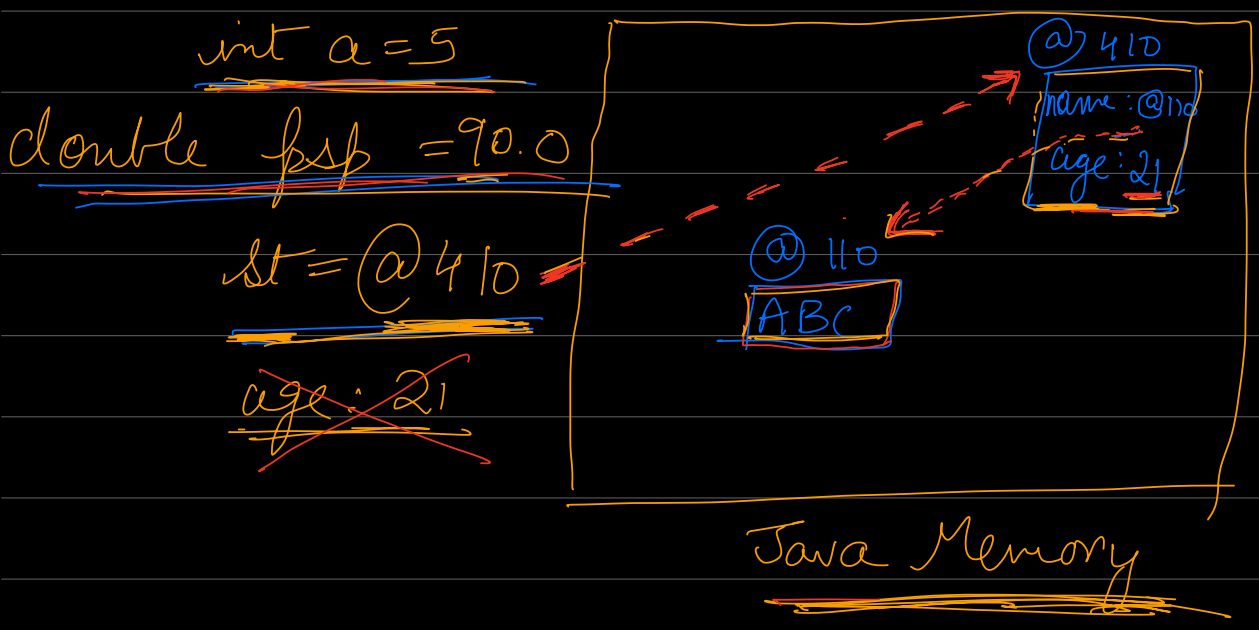
⇓

– 10 parameters

⇒ There are 2 types of data in Java

① Primitive data types ⇒ int, float, double
directly stored in variable itself

② Objects ⇒ stored in Java memory and variables store the address of object.

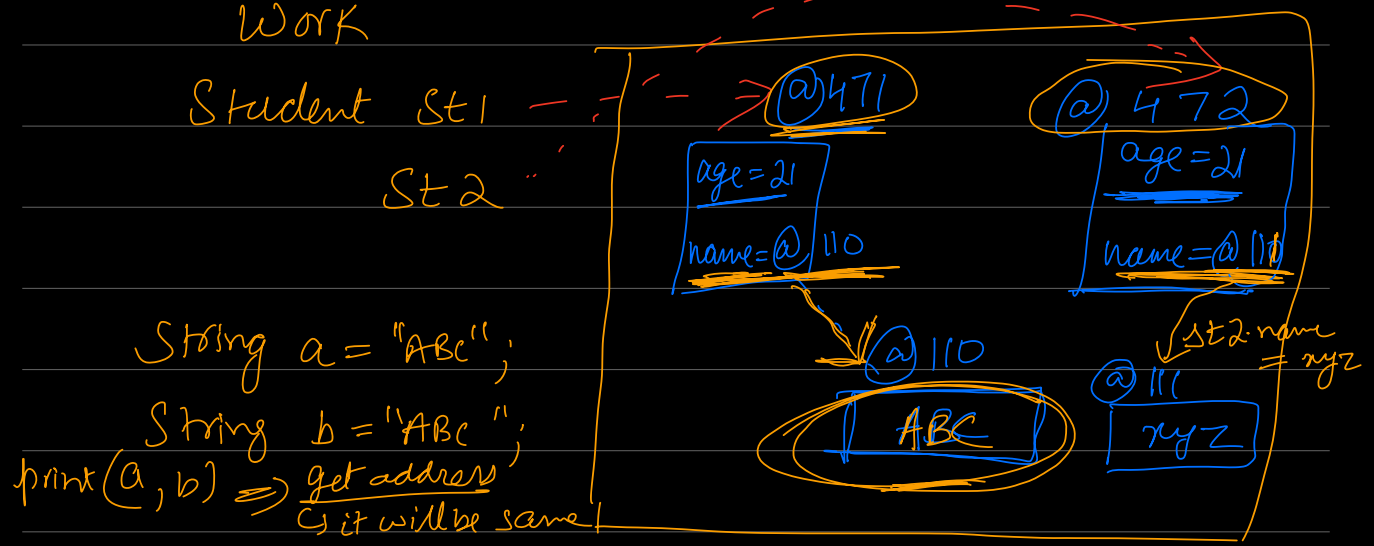


Student (st) = new Student();

String name = "Abhinav";

name ⇒ Abhinav

let us see how does the copy constructor work



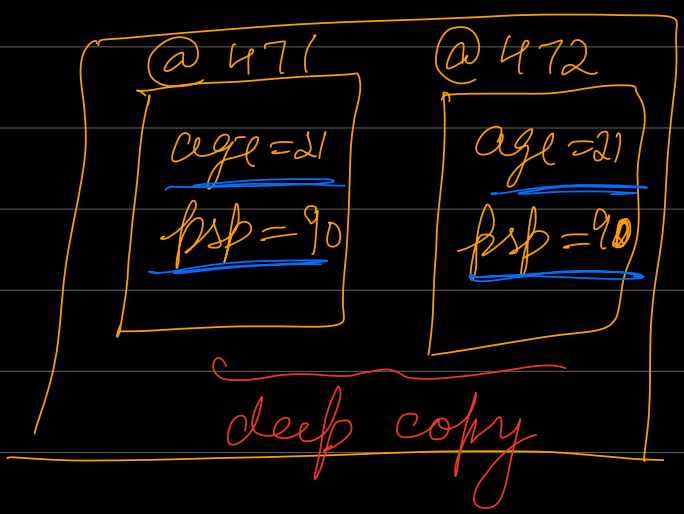
⇒ this type of copy where 2 objects share some attributes is called a shallow copy

⇒ Deep copy ⇒ where no attributes are shared between 2 objects.

⇒ In Java, deep copy would only be possible when object has only primitive data types.

Student {
int age;
int psp;
}

st1
st2

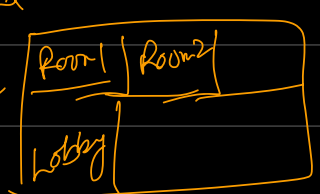


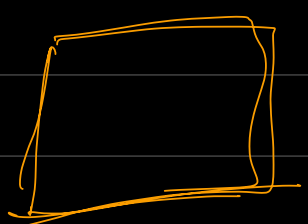
⇒ default
⇒ Parameterized
⇒ Copy ⟶ Shallow
⟶ Deep

Ques for rent class

Q. Java pass by value or pass by reference

Blueprint of House





Actual House

Object.