Constructors, Inheritance & Polymorphism.

AGENDA:

- Constructor
- Types of Constructor
- Deep copy & shallow copy
- Inheritance
- Polymorphism
- Method overloading & overriding.

Rajat Sharma
Vishal Mosa
Naval Oli
Murali Mudigonda
Subhranil Kundu
Shivansh Agarwal
Saurabh Ruikar
Madhan Kumar M S
Shrikanth
Sarthak
amit khandelwal
Divyanshu
soumya hubballi
sharath r
Pankaj
Khushi Raj
Akansh Nirmal
Rajendra
Purusharth A
Vasanth
Sumit Adwani
Bhaveshkumar
Gowtham Sankaran
suyash gupta
Sanket Giri
GAGAN KUMAR S
Vimal
Shani
Venkata Sribhavana Nandiraju
Ankit Mishra
Sneha L
Shradha Srivastava
Sushant Patil
Abhishek Sharma
Rajeeb Kumar Nath
Sridhar Hissaria
Nikhil Pandey

65% ⟶ 70%.

Constructors

Class ⟶ Blueprint

Object ⟶ Instance.

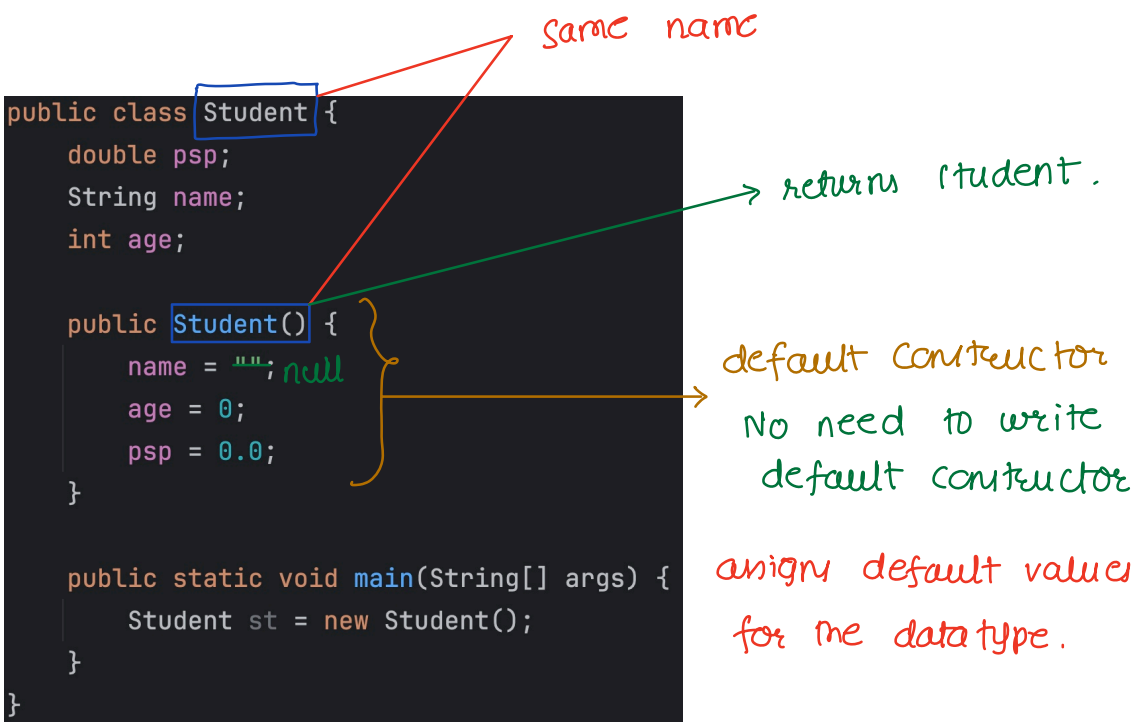
```java
public class Student {
    double psp;
    private String name;
    private int age;

    public static void main(String[] args) {
        Student st = new Student();
    }
}
```

default constructor.

int a = 12

allocate memory in heap

Default Constructor



QUIZ: In what case the default constructor is not created ?

If we define our own constructor. Default constructor is NOT created.

Summary of default constructor

- takes no parameter
- Sets every attribute to its default value
- Created only if we dont write our constructor
- Its public
- return type in not void

Manual Constructor

Defined by the user.

```java
public class Student {
    double psp;
    String name;
    int age;

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

    public static void main(String[] args) {
        Student st = new Student();
    }
}
```

if manual constructor is present, then no default constructor is created.

How to call custom constructor ?

```java
public static void main(String[] args) {
    Student st = new Student( psp: 3.5, name: "John", age: 20);
}
```

How does a custom constructor work ?

① — Initializes all attributes to its default values.

② — Then the custom constructor executes line by line

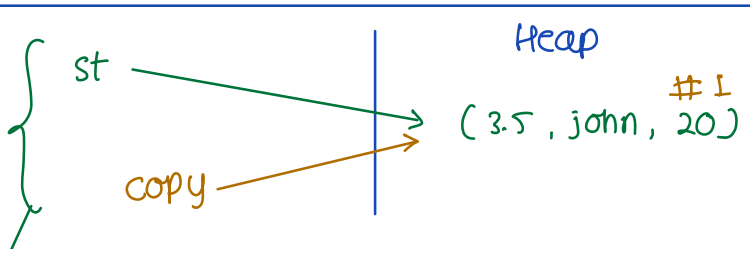
Copy Constructor

I need to make exact student with same attributes as a student st

```java
public class Student {
    double psp;
    String name;
    int age;

    public Student(double psp,
                   String name,
                   int age) { // manual constructor
        this.psp = psp;
        this.name = name;
        this.age = age;
    }

    public Student(Student st) { // copy constructor
        this.psp = st.psp;
        this.name = st.name;
        this.age = st.age;
    }

    public static void main(String[] args) {
        Student st = new Student( psp: 3.5,  name: "John",  age: 20);
        Student copy = new Student(st);
    }
}
```

QUIZ: Is copy constructor same as doing ?

Student copy = st ;



Shallow copy → you think its a copy but its just a reference.

```
public static void main(String[] args) {   args: []
    Student st = new Student( psp: 3.5,  name: "John",  age: 20);   st: Student@756
    Student shallowCopy = st;   shallowCopy: Student@756
    Student deepCopy = new Student(st);   st: Student@756   deepCopy: Student@757
}
```

A = [1, 2, 3]

copy = A

→ A[0] = 10

print(copy[0]) → 10

A = [1, 2, 3]

copy = copy.deepcopy(A) → a function to create a copy

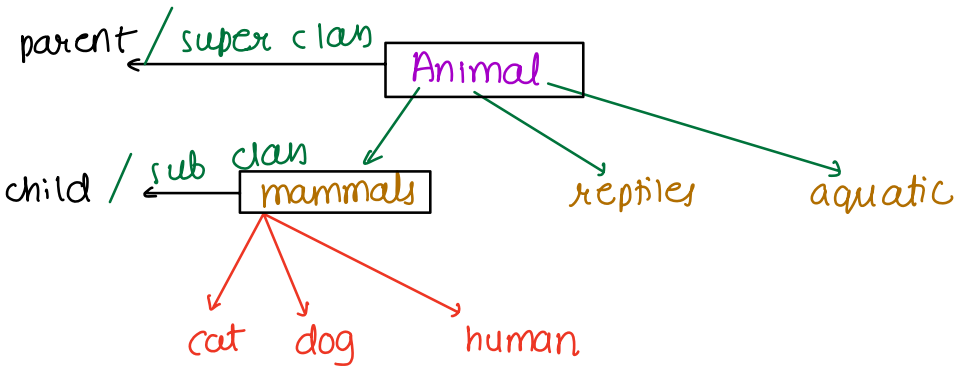
→ A[0] = 10

print(copy[0]) → 1

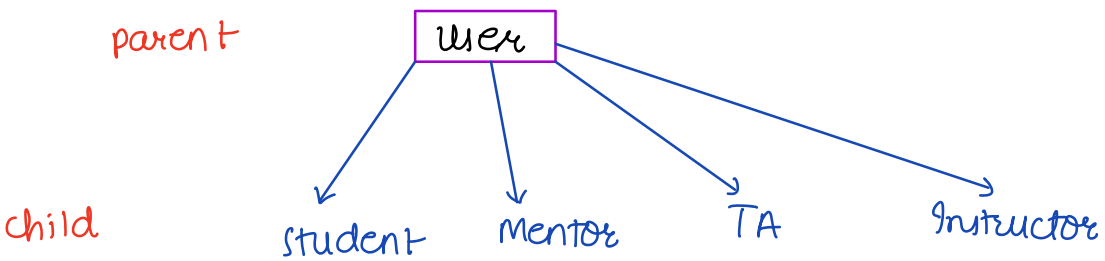
→ A copy constructor creates deep copy.

Inheritance

⟶ You can define relationship b/w different entities.



If animal can walk, human can walk ?
humans can walk because animals can walk.



User has a username and user can login
⟶ student can login ? ✓
mentor can login

Child class inherits all attributes & methods from parent.

Parent Child Relationship in code

```java
class User {
    String username;

    void login() {
        System.out.println("Logging in... " + username);
    }
}
```

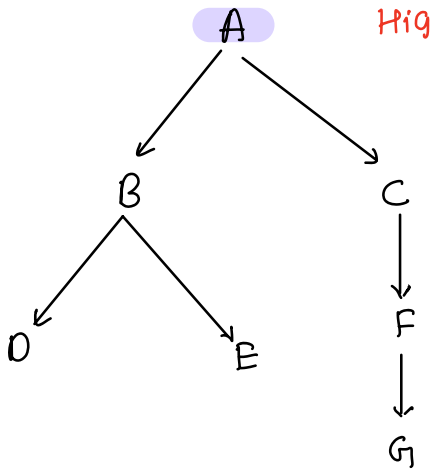
Assume Instructor is a child/subclass of User ?
How to write it in code ?

class Instructor extends User

```java
class Instructor extends User {
    String batchName;
    double avgRating;

    void teach() {
        System.out.println("Teaching... OOPS 2");
    }
}
```

Instructor i = new Instructor();
i.username ✓
i.login() ✓

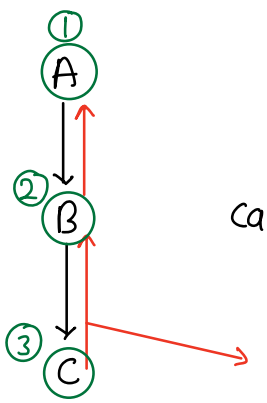
default constructor



Highest level of abstraction

Do child class contain all attributes of parent. ?

Example in ds



C c = new C()

Can a child exist without its parents ?

C asks B ⟶ B asks A

which object will be created first ? A

which object will be created last ? C

Super Keyword

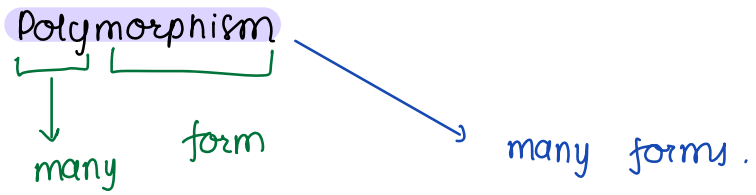
```java
class User {
    String username;

    public User(String username) {
        this.username = username;
    }

    void login() {
        System.out.println("Logging in... " + username);
    }
}

class Instructor extends User {
    String batchName;
    double avgRating;

    public Instructor(String username,
                      String batchName,
                      double avgRating) {
        super(username);
        this.batchName = batchName;
        this.avgRating = avgRating;
    }

    void teach() {
        System.out.println("Teaching... OOPS 2");
    }
}
```

super calls constructor of parent.

super should be called before

super() is always the first line in child's constructor. else Error

Break : 22:51




```
Animal a = new Dog()   ✓
Dog d = new Animal()   ✗
```

```
class A {
    int age
    String name
}

class B extends A {
    String univ
}

class C extends A {
    String company
}
```

HW
Try various scenarios.

```
A a = new C();
a.company
```

Throw an error. Since, class A doesn't contain company. CTE

Method Overloading

Method overloading is compile time polymorphism.

Same method name, but different method signature.

```java
class A {
    void hello() {
        System.out.println("Hello");
    }

    void hello(String name) {
        System.out.println("Hello " + name);
    }
}
```

```
A a = new A();
a.hello();  // Hello
a.hello("Ganga")  // Hello Ganga
```

Q1> `void hello();` `vlod hello(String s);` } ✓

Q2> `void hello(int i);` `void hello(String s);` } ✓

Q3> `void hello(String s);` `String hello(String s);` } ✗

Method Overriding

```
class A {
    void hello(String s) {
        .....
    }
}
```

```
class B extends A {
    String hello(String s) {
        .....
    }
}
```

Is this allowed ?

NO

{equivalent class}

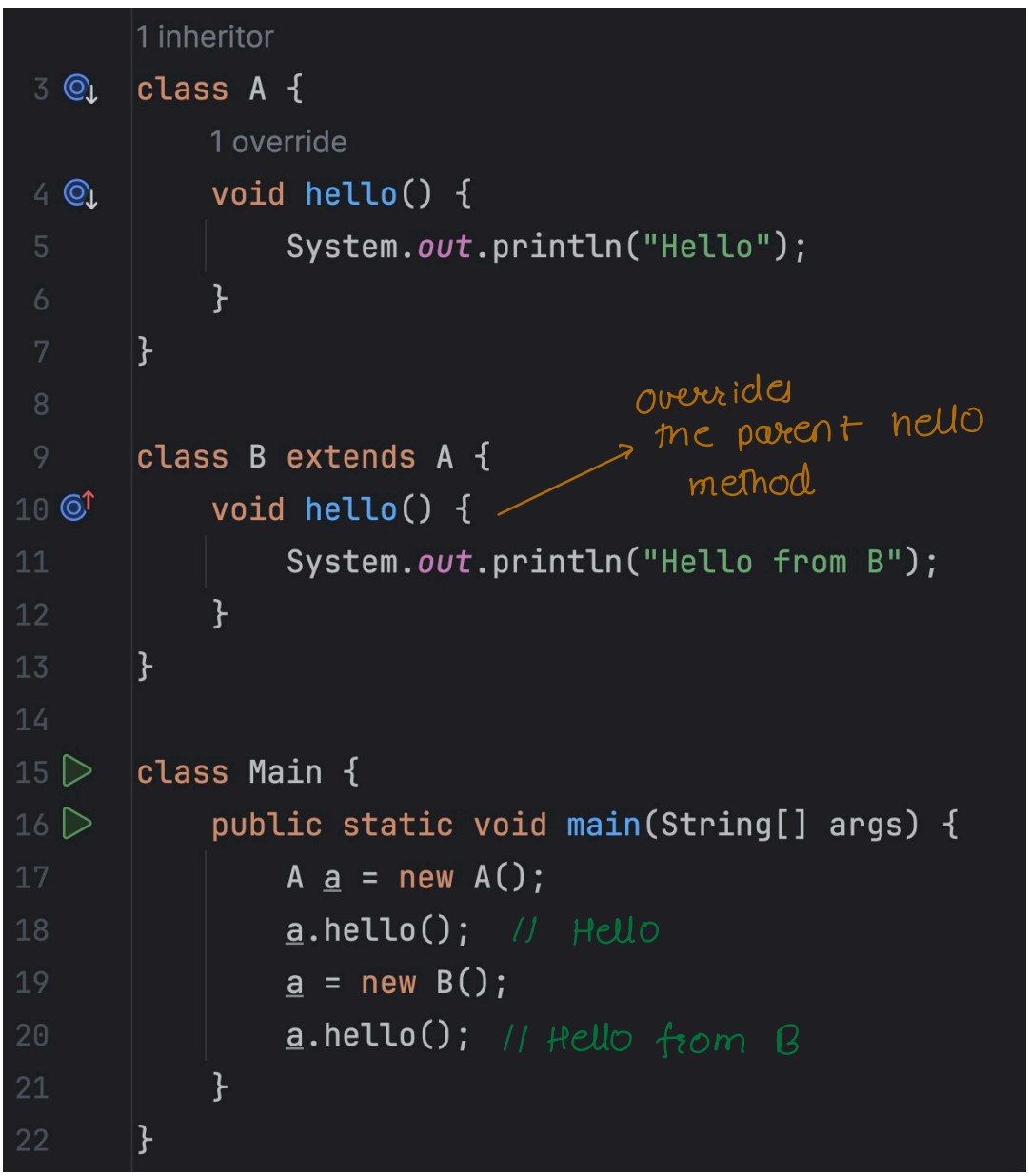
```
class B extends A {
    String hello(String s) {
        .....
    }
    void hello(String s) {
        .....
    }
}
```

```
1 inheritor
class A {
    1 override
    void hello() {
        System.out.println("Hello");
    }
}

class B extends A {
    String hello() {
        System.out.println("Hello from B");
        return "Hello from B";
    }
}
```

CTE ∵ method signature is same but return type of hello method is different.

Method overriding is runtime polymorphism

hello() method in B overrides hello method in A.



Doubt senion