Lecture :- OOPS-2

Agenda

- Constructor
- Types of constructor
- Deep copy and shallow copy.
- Inheritance
- Polymorphism
- Method overloading and method overriding.

Constructors

Class :- blueprint

Object :- instance of class

Example:

```
class Student {
    String name;
    int age;
    double pCp;
}
```

Object :- `Student st = new Student();`

* Declare a variable :- `int a = 12;` (Data type; a → var name)

* Declare an object :- `Student st = new Student();` (Data type; st → obj name; new → initializer)

Qu. What does new Student() part of `new Student()` part of statement do here?

Student() ⟶ constructor
(creates the object of a class)

```
class Student {
    String name;
    int age;
    double pCp;
}
```

⟶ Student() → default constructor.

Default constructor

- if we don't create our own constructor in a class, a default constructor is created.

→ Creates a new object of the class and sets the value of each attribute as the default value of that type.

Data types	Default values
integer	0
String	null / ""
float / double	0.0

```
class Student {
    String name;
    double psp;
    String univName;
    int age;
}
```

```
Student s = new Student();
```

s.name = null
s.psp = 0.0
s.univName = null
s.age = 0

```
class Student {
    String name;
    double psp;
    String univName;
    int age;

    Student() {
        name = null;
        psp = 10.00
        univName = null
        age = 0;
    }
}
```

s.psp = 10.0

```
Student s = new Student();
```

Qu In what conditions a default constructor is not created?

Ans The moment we create our own constructor.

```
class Student {
    String name;
    double pop;
    String univName;
    int age;

    Student() {
        name = "Ayush"
        pop = 18.0;
    }
}
```

```
Student st = new Student();
st.name = Ayush
st.pop = 18.0
```

Default constructor

- Takes no parameter
- Sets every attribute of class to its default val
- Created only if we don't write our own constructor
- Its public, i.e. can access from anywhere.

Manual constructor

```
class student {
    String name;
    private int age = 21;
    String univName;
    double psp;
    student(String studentName, String univerSity Name) {
        name = studentName;
        univName = univerSity Name;
    }
}
```

```
public class Client {
    public static void main(String[] args) {
        Student st = new student(); // Compilation error.
    }
}
```

Qu Why is it throwing error ?

student() is needed

```
class student {
    String name;
    private int age = 21;
    String univName;
    double psp;
    student(String studentName, String univerSity Name) {
        name = studentName;
        univName = univerSity Name;
    }
    student() {
    }
```

student s = new student();

```
public class Client {
    public static void main(String[] args) {
        Student st = new student("Ayush", "IP");
    }
}
```

st.name = Ayush
st.univName = IP.

Copy constructor

Student s1 = new Student();
- age = 25;
- name = "Ayush".

s1.age = 25
s2.name = Ayush

* Create a object that has exact same values of attributes as older objects.

Student s2 = s1; || s2.age = 25
s2.name = Ayush

* Create a new object that has exact same values of attributes older objects.

```
class student {
    string name;
    private int age;
    student() {
        name = null;
        age = 0;
    }
    student(student st) {
        name = st.name;
        age = st.age;
    }
}
```

```
student st = new student();
st.age = 25;
st.name = Ayush;

student s2 = new student(st);
    s2.name = Ayush
    s2.age = 25
```

Quiz! Is copy constructor same as doing

Student s2 = s1;

a) Yes

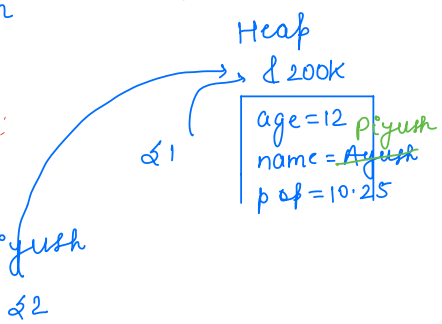
b) In some cases only

c) ~~No~~

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

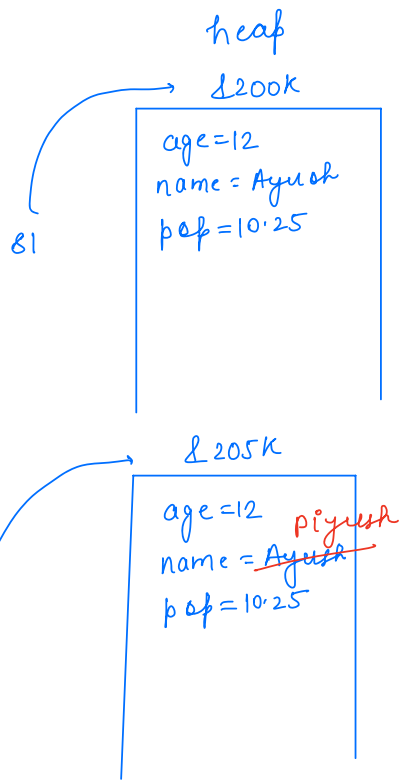
Shallow copy

```
Student s1 = new Student(12, Ayush, 10.25);
s1.name = Ayush
Student s2 = s1;
s2.name = "Piyush";
print(s1.age); // 12
print(s1.name); // Piyush
print(s1.pop);
    // 10.25
```



Deep copy

```
Student s1 = new Student(12, Ayush, 10.25);
Student s2 = new Student(s1);
s2.name = "Piyush";
print(s1.age); // 12
print(s1.name); // Ayush
print(s1.pop); // 10.25
```

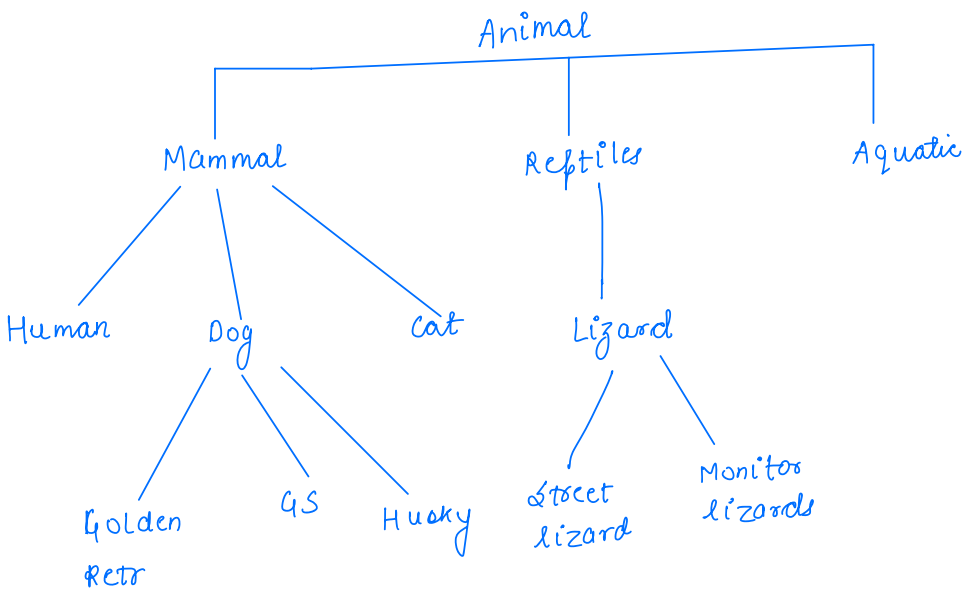


Inheritance

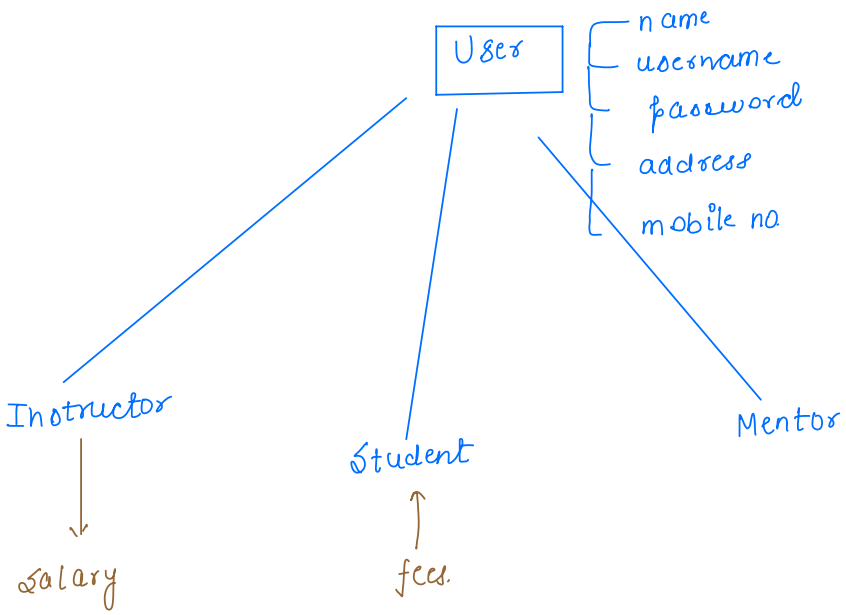
Qu why had OOPs taken over procedural programming ? what was the idea behind it ?

Ans Entity - activity relationship

Examples Classification of animals

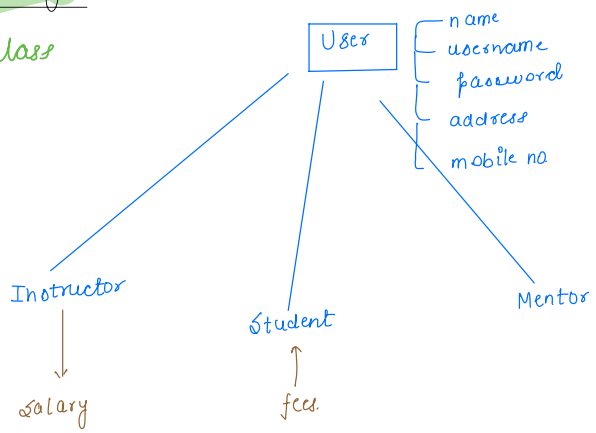


Scaler heirarchy of representation



Inheritance in coding

```
class User {          ⟶ Parent class
    String userName;
    String password;
    void login() {
    }
}
```




```
class Instructor {
    int salary;
}
```

java ⟶ class Instructor extends User

python ⟶ class Instructor (User):

C++ ⟶ class Instructor: public User

C# ⟶ class Instructor: User

```
class Instructor extends User {

}
```

```
Instructor i = new Instructor();
i.username = valid
```

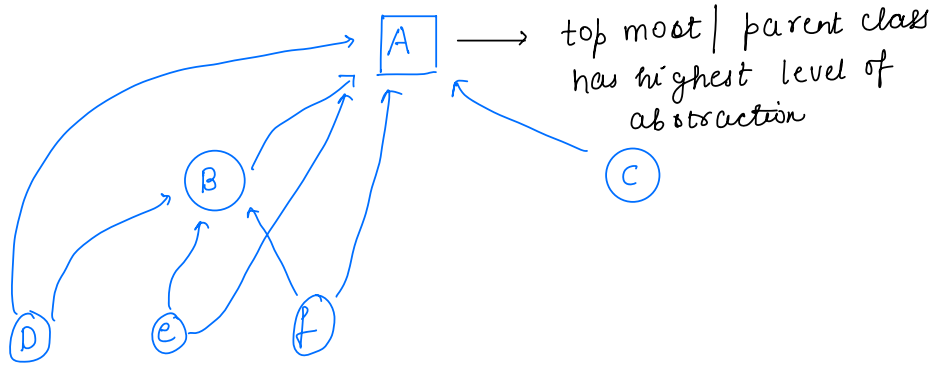
Constructor chaining

```
Instructor i = new Instructor();
```

default

```
i.userName = "Ayush"
i.login()
i.rating = 4
```

```
class Instructor extends User {
    int rating;
}
```

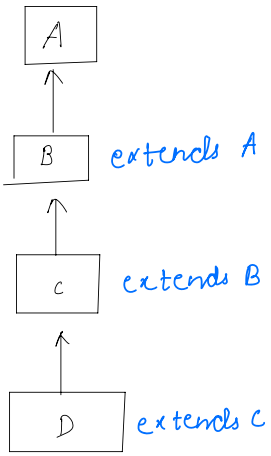


Steps to create an object of child

B is a child of A

C " " " " B.

D " " " " C.



Object of D: `D d = new D();`

What really happens when we call D()?

1. Constructor of D is called
2. Since D is child of C. so before its execution, constructor of C is called.
3. Illy, constructor of C will B
4. " " " B " A

```
D d = new D();
      |
      A
      B
      C
      D.
```

Who's constructor will be finished first? (A)

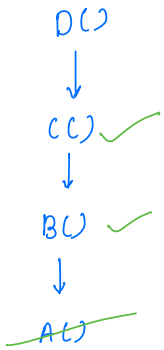
```
* class A {
      A() {
        print(" constructor of A ");
      }
  }

  class B extends A {
      B() {
        print(" constructor of B ");
      }
  }

  class C extends B {
      C() {
        print(" constructor of C ");
      }
  }

  class D extends C {
      D() {
        print(" constructor of D ");
      }
  }

  class main {
      main() {
        D d = new D();
      }
  }
```



Constructor of A
" " B
" " C
" " D

```
* class A {
      A() {
        print("constructor of A");
      }
  }

  class B extends A {
      B() {
        print("constructor of B");
      }
  }

  class C extends B {
      C(String message) {
          print(message);
      }
      C() {
      }
  }

  class D extends C {
      D() {
        super("hello");
        print("constructor of D");
      }
  }

  class main {
      main() {
        D d = new D();
      }
  }
```

constructor of immediate parent

```
D d = new D();
  ↓
C();
  ↓
b()
  ↓
a() → const of A
      " " B
      hello
      " " D
```

Break: 8:36 – 8:46 AM

Polymorphism

Poly means many and morphism mean form

Qu In today's class, have you learned about something which had multiple forms ?

User had multiple forms
- instructor
- TA
- student
- Mentor

Another example

Animal
- Aquatic
- Amphibians
- Mammals
- Reptiles

✓ `Animal a = new Dog();` — Creating an obj of Dog.

Dog → Mammal → Animal

`Dog d = new Animal();` ✗

Animal() → Croc

```
class A {
    int age;
    String name;
}
```

```
B extends A {
    String univ;
}
```

```
C extends A {
    String company;
}
```

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

✗ compilation error

```
C c = new A();
```

✗

```
C c = new C();
```

✓

Method overloading / overriding

```
void print(int a){
    print(a);
}
void print(int a, int b){
    print(a + " " + b)
}
void print(a, b, c){
    print(a + " " + b + " " + c);
}
void main(){
    print(5);   (1)
    print(3,4,5);   (3)
}
```

Compile time polymorphism

void print(String str)

String print(String str)

not allowed at all

```
String x = print("hello");
print("hello");
```

Method overriding — h/w

Thankyou ☺