Agenda

Mon Jan 1
No class

Constructor
Types of constructor
Deep copy vs shallow copy
Inheritance
Method overloading

Currently
Contest scaleup

Constructor

```
class Student {
    String name
    int age
    double psp
    public Student ( int age ) {
        this.age = age
    }
    public Student ( Student s ) {
        this.name = s.name
        this.age = s.age
        this.psp = s.psp
    }
}
```

this → current object

Copy Constructor

Student S_1 = new Student (25)

Another independent new student object with same data values as another object

Student S_2 = new Student (S_1)

1) Student s_2 = new Student (s_1) } Deep copy

2) Student s_2 = s_1 } Shallow copy

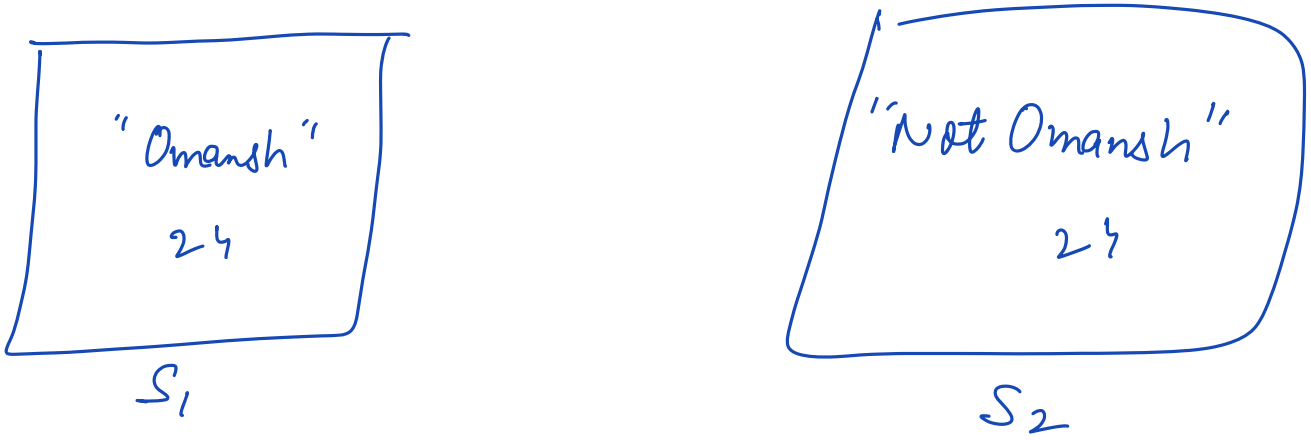
DIFFERENT

⇒ Deep copy vs Shallow Copy

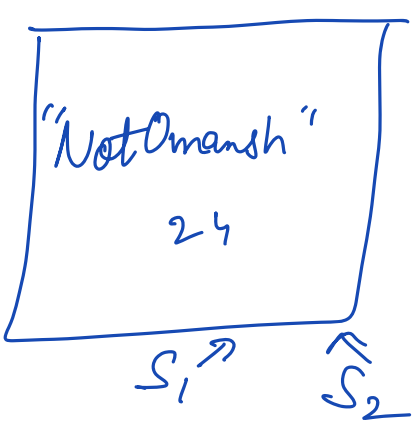
Deep copy vs shallow copy

Deep copy : Create new object & copy all data
Shallow copy : Copy the objects reference, not the object data

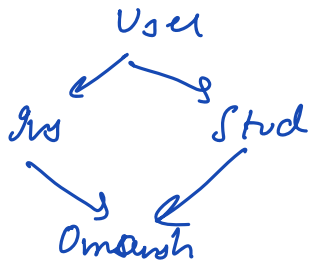
```
Student s1 = new Student (24, "Omansh")
Student s2 = new Student (s1)
s2.name = "Not Omansh"
print (s1.name)  ⇒ Omansh
```

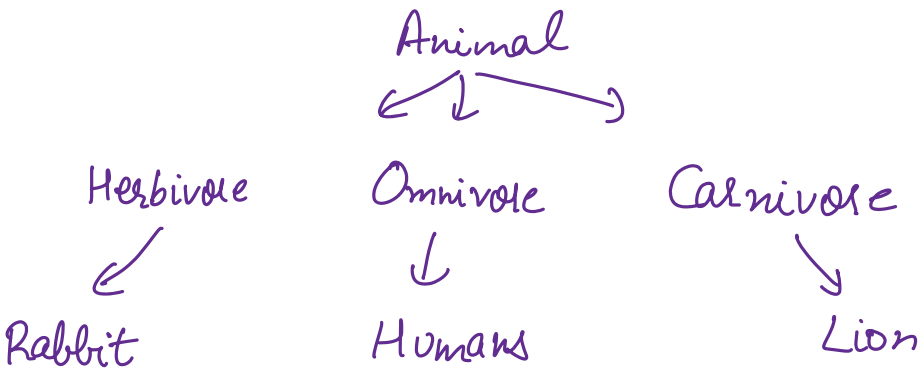



```
Student s1 = new Student (24, "Omansh")
Student s2 = s1
s2.name = "Not Omansh"
print (s1.name) ⇒ NotOmansh
```

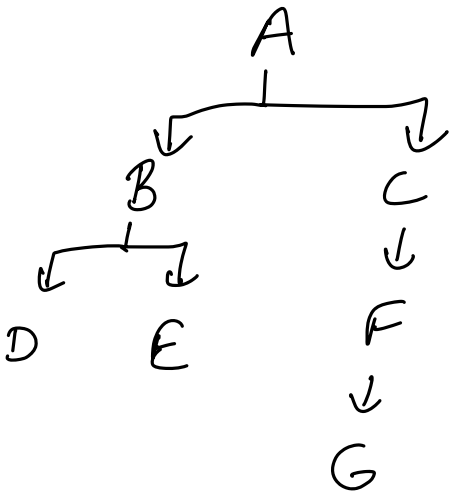


Inheritance





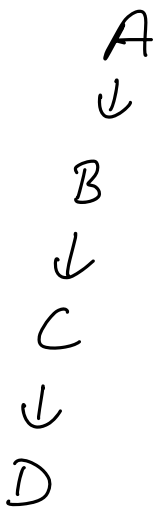

```
class Herbivore extends Animal {
    ⋮
}
class Rabbit extends Herbivore {
    ⋮
}
```



User

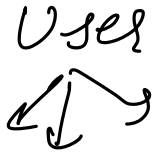
↓

Instructor



D obj = new D()

Order of constructors ⇒ A, B, C, D



Polymorphism — Ins TA Stud

→ Method overloading
→ Method overriding

- Method overloading

```
int add (int a, int b) {
    return a+b
}
```
(1)

```
int add (int a, int b, int c) {
    return a+b+c
}
```
(2)

```
int add (int[] arr) {
    sum = 0
    for (i=0; i< arr.length(); i++) {
        sum += arr[i]
    }
    return sum
}
```
(3)

print (add ({10, 20, 30, 40}))

Caveat → Corner case

```
void hello (string s)
string hello (string s)
```

This is NOT Allowed hello("Omonsh")

- Method overriding

parent find many x
&
you many y

```
class A {
    void doSomething () {
        print ("Hello")
    }
}
class B extends A {
    void doSomething () {
        print ("Bye")
    }
}
```

A
↓
B

User
↓
Student

Cast

```
A obj1 = new A()
obj1.doSomething ()        Hello

obj1 = new B()
obj1.doSomething ()        Bye
```

Depends on data type present at run time, not type of variable when defined.

A

B

C