Interfaces & Abstract classes

Interfaces

$$\begin{tikzcd}
& \text{Animal} \arrow[dl] \arrow[dr] & & \\
\text{Aquatic} & & \text{mammals} \arrow[dl] \arrow[dr] & \\
& \text{Dog} & & \text{Cat}
\end{tikzcd}$$

$\longrightarrow$ categorize entities on the basis of logical / physical relationships

bark()
run()

$$\begin{tikzcd}
\text{Robotic Dog} \arrow[d] \\
\underline{\underline{\text{run}}}
\end{tikzcd}$$

organise a race $\longrightarrow$ <u>run</u>

List < ? > participants

Animal $\longrightarrow$ ✗

Dog $\longrightarrow$ ✗

amphibians

categorize on the basis of a $\boxed{\text{behaviour}}$

$\downarrow$

<u>Interface</u>

```
interface Runner {
      abstract
        ^ void run();      ——→  no defⁿ is required.

  }

class RoboticDog implements Runner {

        void run() {

               ≡

        };

}
```

class implementing an interface has to ensure that it defines all the methods of the interface.

$$\begin{tikzcd} & \text{array} \\ \text{Stack} \arrow[r, dash] & \text{linkedlist} \\ & \text{Queue.} \arrow[from=2-1, to=1-2] \arrow[from=2-1, to=3-2] \end{tikzcd}$$

push()	:	inserting element
pop()	:	remove last inserted ele.
top()	:	top-most.

```
interface Stack {
    void push( elem );
    void pop();
    int top();
}
```

[by default public]

→ Abstraction

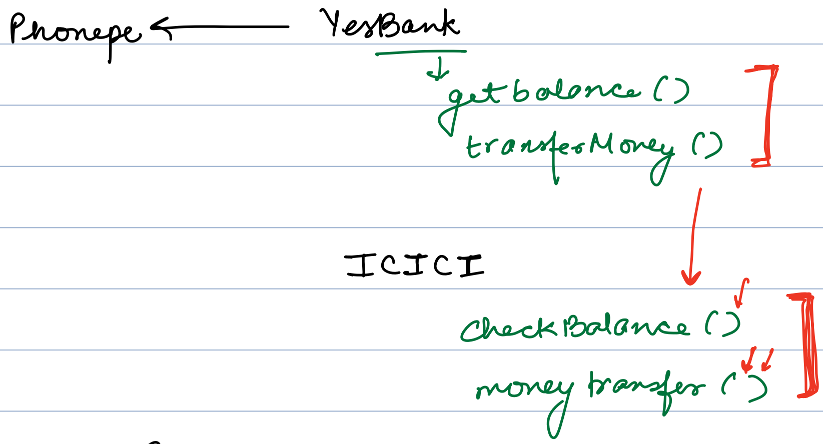
```
class ArrayStack implements Stack {
    push() {
        =
    }
    ⋮
}
```

$\boxed{\text{default / static}}$ ⇑ java 8 +

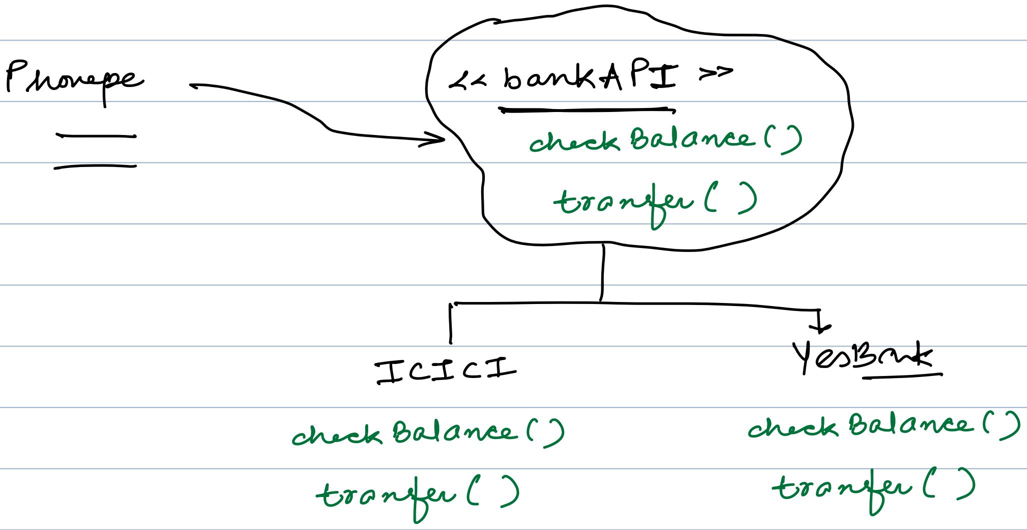
↓

public static final

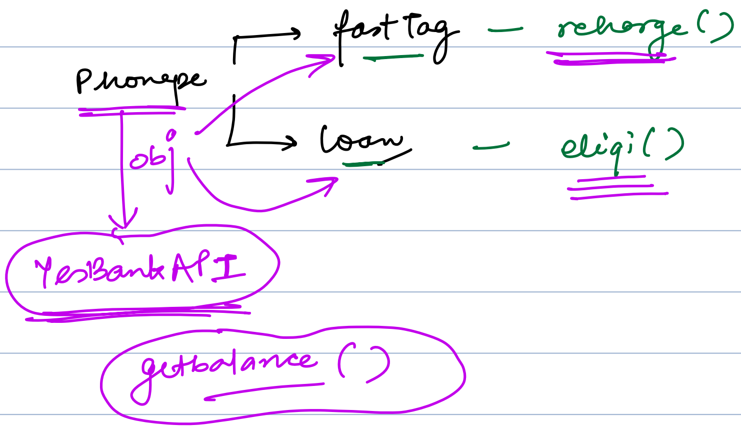
↓ constant



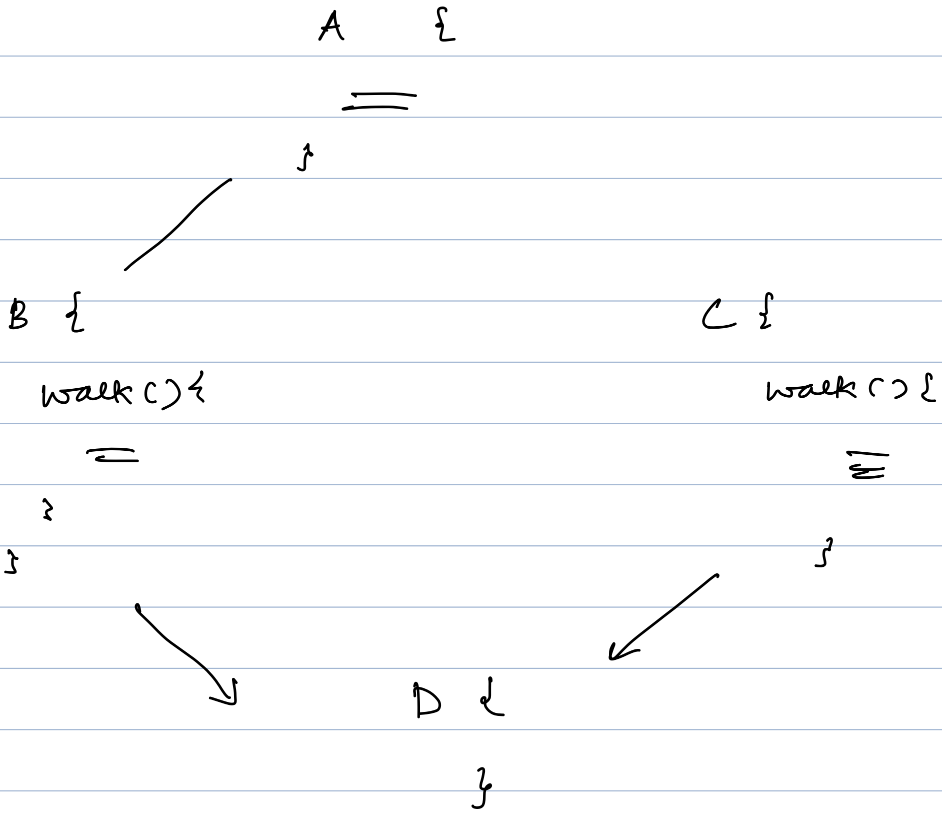
<u>RBI</u>
↳ UPI provider



" code to interfaces not implementation "



A class can implement multiple interfaces.



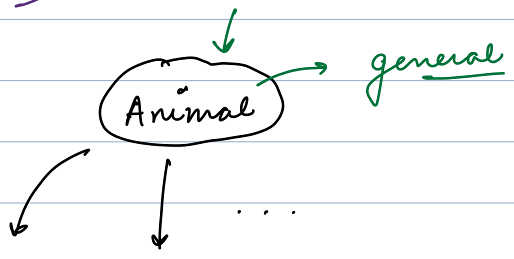
D d = new D();
d.walk();

```
interface B {
    walk();
}

C {
    walk();
}

D implent B, C {
    walk() {
        =
    }
}
```

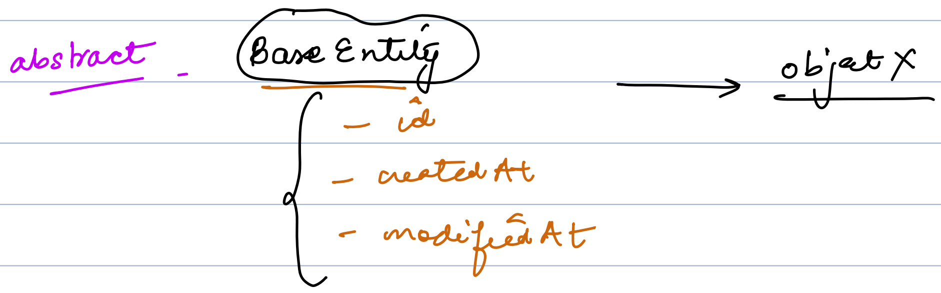
$\boxed{10:30 \text{ pm}}$

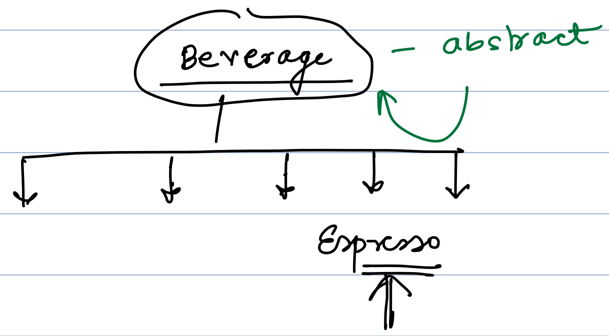
Abstract classes

vague
overview



(I) Restriction of creation of objects.





II Abstract

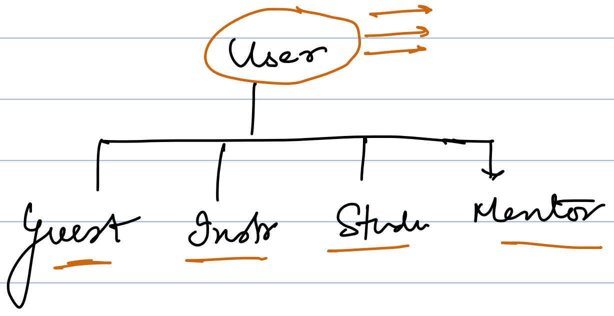
- normal methods/ fields.
- abstract methods

  ↓

  no def^n, only declarations.

abstract class ⟶ X might not have abstract method.

if there is abstract method, class has to be abstract.



— abstract methods.

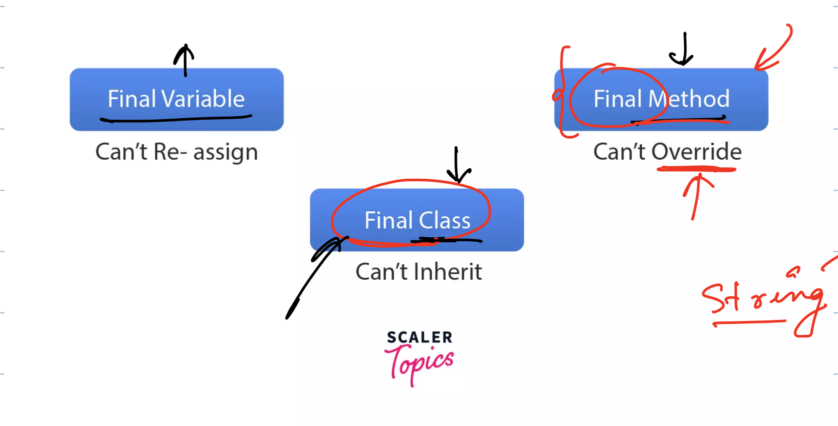
abstract Animal {

{ abstract void makeSound();

}

X { Interfaces ⟶ fields X

Final

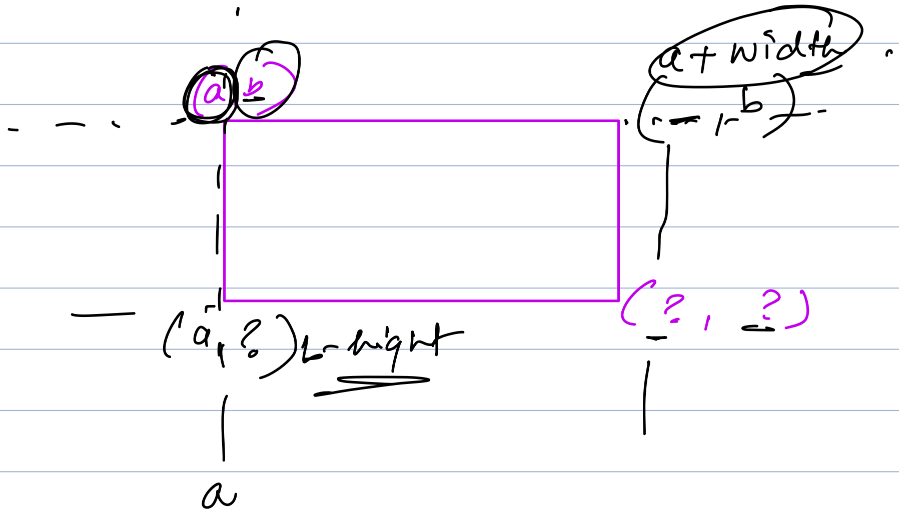


String — final

```
final class ——— {
    final int x;
    final void y() {
    }
}
```

HW : I read the article

II OOPS - 1, 2, 3, 4


a
b
a + width
b
(a, ?) ← height
(?, ?)
a