

Doubts

- 1 Finalize (Garbage Collection) VS Final VS Finally.
- 2 Synchronized Methods Multi-Threading →
(wait & stop)
- * 3 Static Keyword
- 4 Abstraction VS Encapsulation
- 5 .toString()
- 6 (==) VS equals()
- 7 Overloading VS Overriding (enum & ~~comp~~) (Static VS Dynamic Binding) [Polymorphism]
- 8 Friend Functions
- 9 JDBC
- 10 String Buffer VS Builder
- 11 Downloader
- 12 Copy Constructor VS Virtual Function
- ✓ 13 Serialization
- 14 Visualization

15 Composition VS Aggregation VS Association.

16 Linear VS Non-linear D.S.

→ 17 Hierarchy of Collections

18 linked list VS Arraylist

19 This Key Word.

20 Run VS Start

21 Process VS Thread

22 Heap VS Stack Memory.

23 Generics

24 Comparator Interface. VS Comparable

25 Abstract Keyword → Interface Also

26 Type Casting → (2)

27 Scanner VS Buffered Reader] File Handling

28 Exception → Inheritance

34 char to int is Casting?

29 Countdownlatch *

35 Access Specifier

30 Tokenizer String

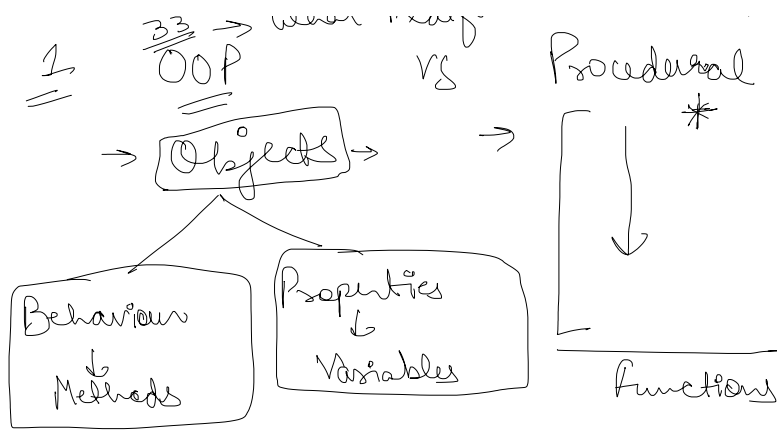
36 Super in Interfaces

31 Why OOPS VS Procedural

32 Bitwise operators

33 → What Platform JOL is build upon? [C?]

1 OOP VS Procedural



Divided into Objects
Bottom-up Approach

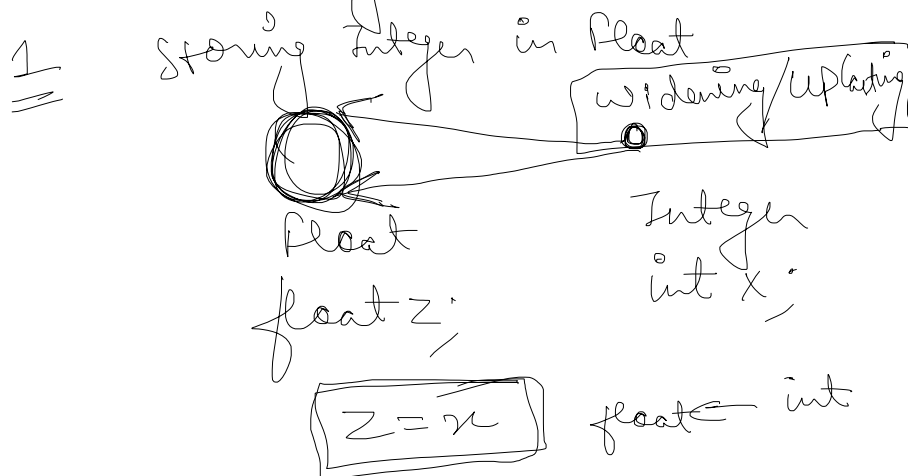
Purely Real World

Top-Down Approach

~~Not~~ ~~Easy~~
to add
No Polymorph
X Real world

2 Type-Casting

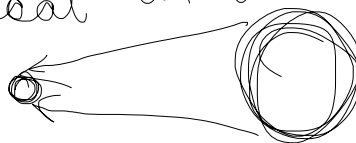
\hookrightarrow Converting one value from one type to other



Implicit

2

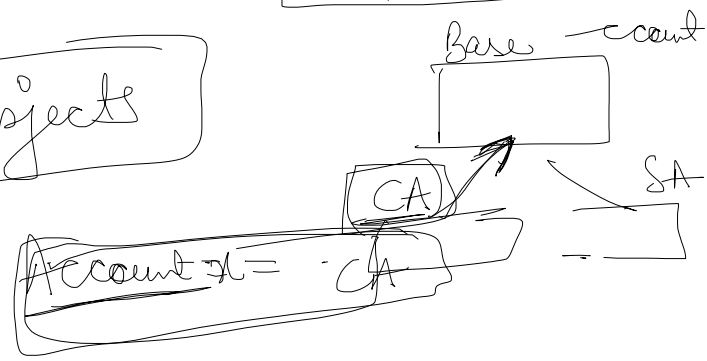
Storing float in int



Narrowing/Down Casting

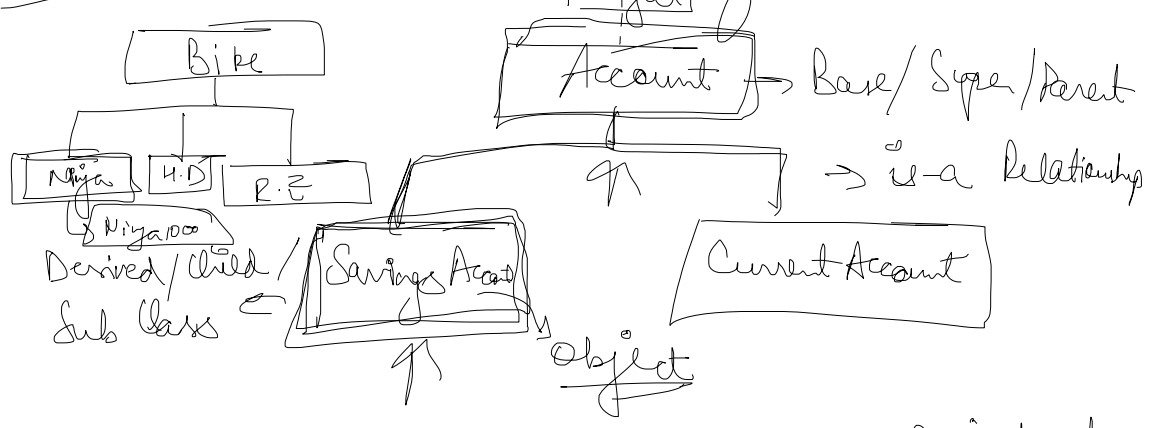
$\text{float } z$ $\text{int } x$
 $x = (\text{int}) z$
Explicit

Objects



3

Inheritance & Interfaces

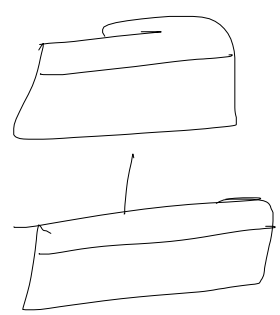


Object of Derived Class - is an Object of Parent Class.

SavingsAccount x = new SavingsAccount();

extends Keyword Inheritance
Syntax
 Class Derived extends Base

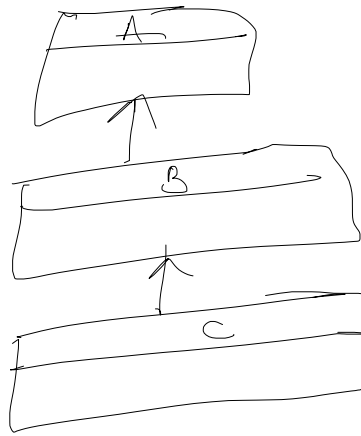
1



Single

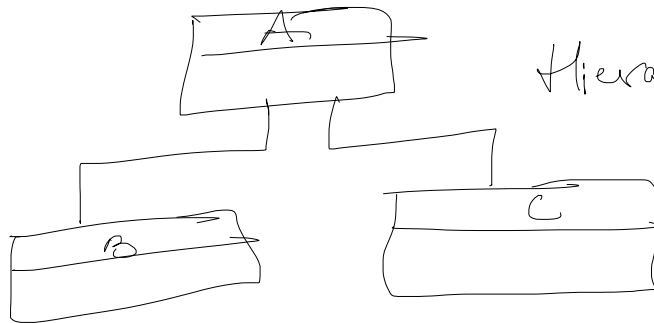
2

Multi-level



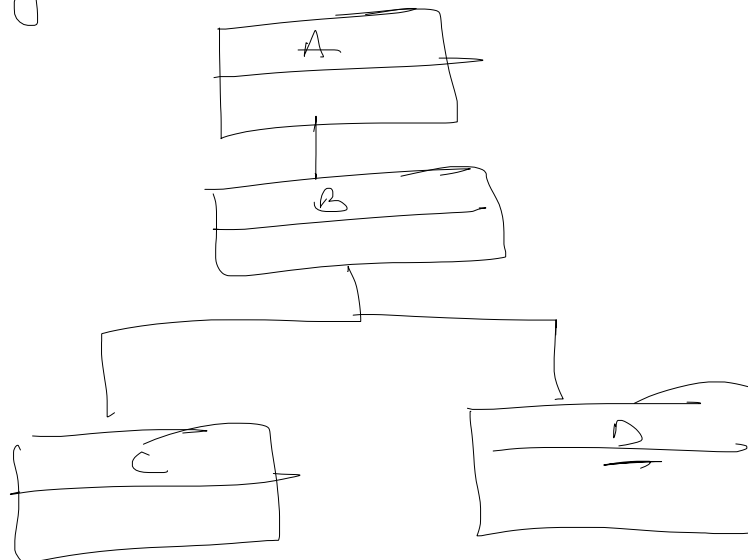
3

Hierarchical



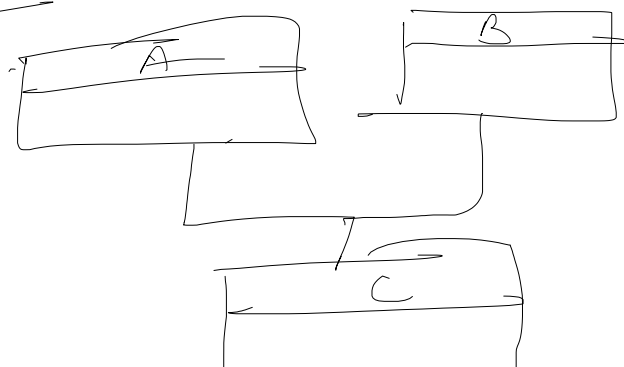
4

Hybrid



5

Multiple





Not through classes.
Only through interface

class A

```
{ void Test()
  { S.O.P("Hi I'm A");
  }
}
```

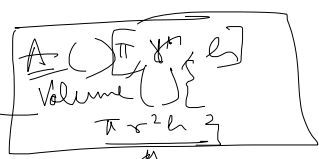
class B

```
{ void Test()
  { S.O.P("Hi I'm B");
  }
}
```

class C extends A extends B

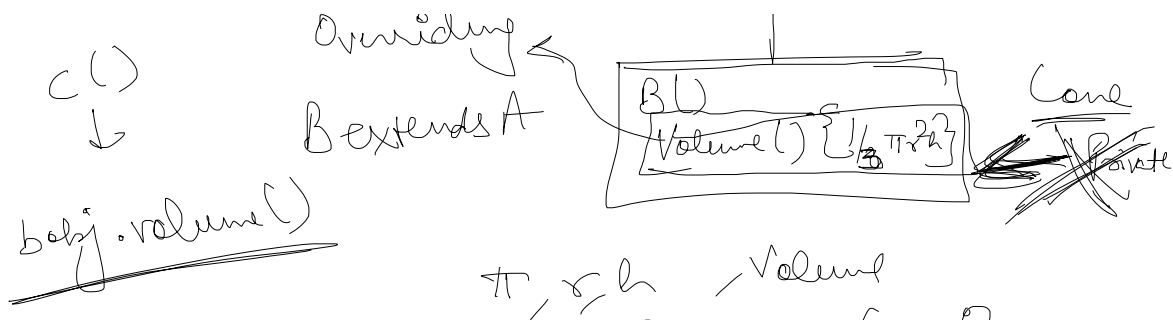
```
{ S.O.P(Test());
}
```

4 Overriding & Overloading / Polymorphism / Binding

Overridden ←  Cylinder
Public

Overriding

 Cone



Overriding \rightarrow $\text{Equals}()$ [String]
 Content of Object.

\rightarrow Signature of Overriding func
 should be same as that of
 Overridden function.

\rightarrow Access Modifier of overriding
 method must have higher/equal
 level of accessibility

Base \rightarrow public int fun (int i, float f, String s)

Yes \leftarrow public int fun (int i, float f, String s)

No \leftarrow protected int fun (int i, float f, String s)

No \leftarrow public float fun (int i, float f, String s)

No \leftarrow public int fun (int i, float f)

No \leftarrow public int fun (float f, int i, String s)

No \leftarrow public int fun (int f, float s, String s)

Method Overloading

within the same class

Within the Same Class

Salary Calculator

```
{  
    Calculate(int Salary)  
    {  
        return Salary * 12;  
    }  
    Calculate(int Salary, int months)  
    {  
        return Salary * months;  
    }  
}
```

Method

Overloading

- No. of Parameters
- Types of Parameters
- Return types not considered.

```
class Test  
{  
    void fun(int, int)  
    void fun(int, char)  
    int fun(int, int)  
}
```

→ Error

Confusion

```
class Main  
{  
    PSVM(S A[]) {  
        fun(5, 5);  
        fun(5, 10);  
        fun(5, 5); *  
    }  
}
```

Run Time

```
class A { void Test() {} }  
class B extends A { void Test() {} }
```

Run Time

```
class B extends A { void Test () {} }
class C { main() {
    B.Test();
} }
```

Method Overloading

println()

println(int)
(float)
(float, int)
etc... etc.

Poly Morphism
Multiple ways to exist

Keywords

Super → Refers to the Parent/Base class of class in which it is used.

A
int i = 5
fun() { S.O.P("A") }

B
int i = 7
S.O.P(i);
S.O.P(super.i);
fun(S.O.P(i) in B)

fun() → B
super.fun() → A

1. Variables (Instance) ✓
2. Methods ✓
3. Constructor ✓

Overriding

super();

super(parameters)

Derived

Constructor Overloading

```
class A {
    int n;
    A() { n = 5; }
}
```

class B

Parent first

Exception

$$\{ B(\cdot) \}_{\mathbb{Z}} \circ \underline{A(\cdot)}' \rightarrow$$

B.obj = new D();