1__26-06-2022

Sunday, 26 June 2022 8:03 PM

oops overview

- class
- objects
- Encapsulation
- Data abstraction
- Polymorphism
- Inheritance

Apart from these concepts, we will be using some other concepts in object oriented design –

- coupling
- cohesion
- Aggregation
- Composition
- Association

oops – object oriented programming system.

↳ oops provides us concepts which we

will be using to design programs using classes and object.

<u>Object</u>

↓

Any entity in the real world having its properties and state.

It is also known as instance of the class.

Eg- Dog
- Breed
- name
- birth year

} properties

- Barking
- Eating

} Behaviour

<u>Class</u>

↳ Template or blueprint to create object.

A Class

public class className

{

Methods + properties

}

Car
- model
- color
- year
- start()
- stop()

Coupling

C1 C2

when one class has information of the another class then we call it as strong coupling.

Access modifiers —
- public
- private
- protected
- Internal

$\downarrow$

This provides visibility levels for class | methods | properties

Interfaces provides weaker coupling.

$$\begin{tikzcd} C1 \arrow[d] \\ C2 \end{tikzcd}$$

They will have concrete implementation

$\downarrow$

Stronger coupling

$$\begin{tikzcd} I1 \arrow[d] \\ C1 \end{tikzcd}$$

No concrete implementation

$\downarrow$

weaker coupling.

[illegible]

Cohesion:

weakly cohesive will split the task into separate parts.

Java . util. io → strongly cohesive class

Java. util. lang → weaker cohesive class.

Association

↳ Relationship between two classes — how two classes are associated with each other.

$$\begin{tikzcd} O1 \arrow[r, leftrightarrow, "Association"'] & O2 \end{tikzcd}$$

- → one to one
- → one to many
- → Many to one

↳ Many to many

Eg- PM of a country

↳ one to one

one PM having many ministers under him

↳ one to many

Many MPs having one PM

↳ Many to one

Many ministers can have many departments

↳ Many to many.

Two flows for association

↳ unidirection

↳ Bidirection

Aggregation (weak association)

↳ It is a way to achieve association.

$$\begin{tikzcd}
\boxed{C1} \arrow[r] \arrow[d] & \boxed{C2} \arrow[d] \\
O1 & O2
\end{tikzcd}$$

Aggregation means when an object contains another one as part of itself.

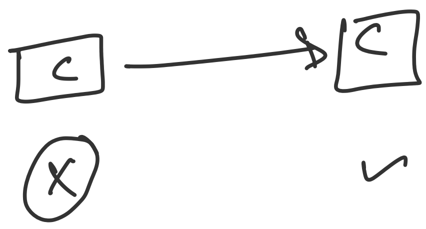
- has-a relationship
- is-a relationship

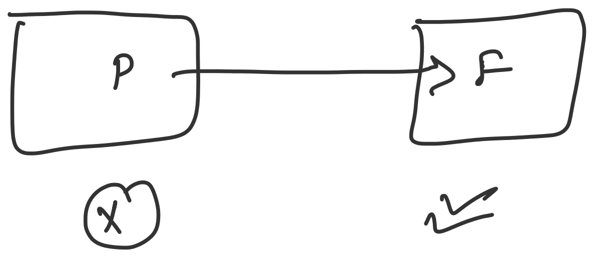
Composition

↳ It is also a way to implement association.

↳ strong association

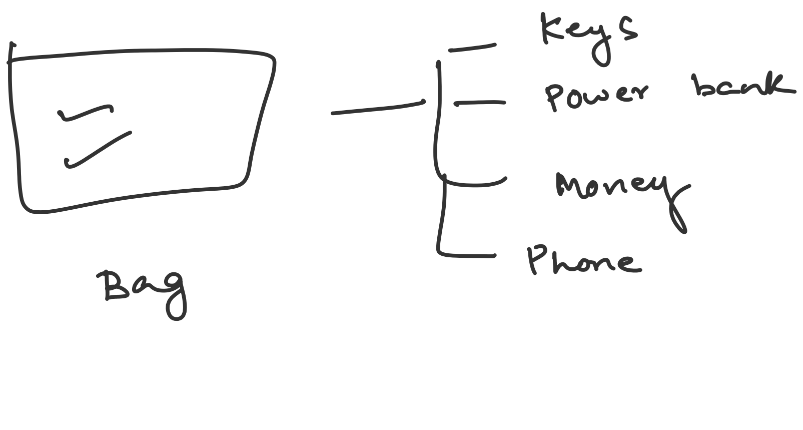
↳ Here we have strong relationship between containing object and dependent object.

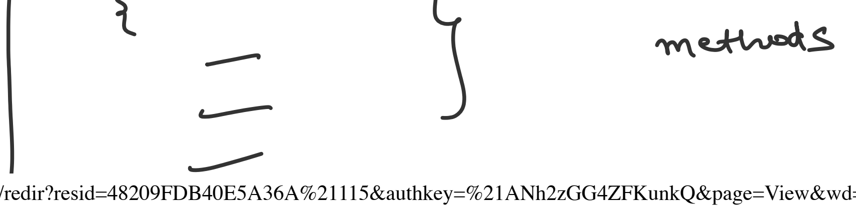




Encapsulation

↳ wrapping up of data into single unit.





L }

Inheritance

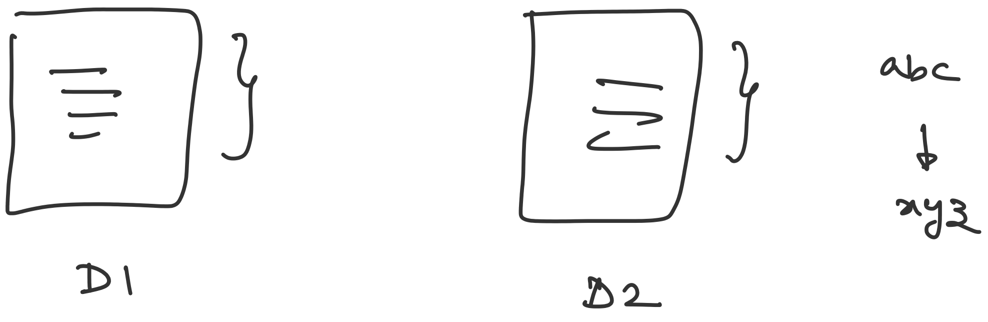
Parent class / super class | base class

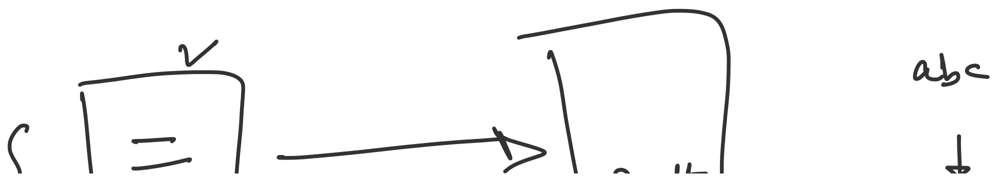
↓

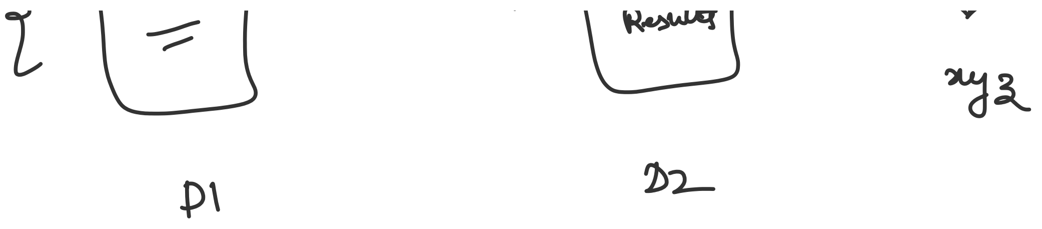
Child class | sub class | Derieved class

This is a mechanism where one object acquires properties and behaviour of another object.

This is used for reusability.







when we inherit parent class then we reuse the methods and fields of the parent class.

Moreover you can add new fields and methods to your class.

we use "extends" keyword to perform inheritance.

```
class A
{
   ==
}

class B extends A
```

```
extends
{
    =
}
```

Subclass inherits all the members from the superclass.

members
- fields
- methods
- nested classes

constructors
- Default
- Parameterized ctr
- copy ctr
- Private ctr

↓

are not members

constructors are not inherited by the sub-classes, but it can be invoked.

```
Super class
{
    ctor( )
    {  =
    }

    =           Print()

}


subclass
{
    ctor ()
       ↳  super. ()
                   Print()
  }
```

For methods with similar names —

super . method( );

For variable with similar names —

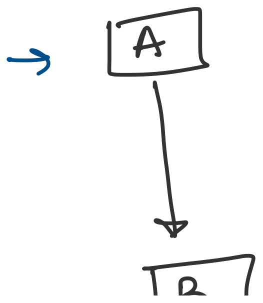
super . variable ;

Invoking super class constructor –

super( );
↑
Parameters

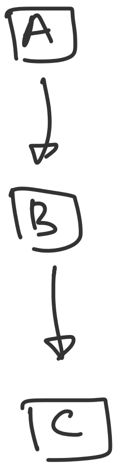
Types of inheritance
- → Single inheritance
- → Multi level inheritance
- → Hierarchical inheritance
- → Multiple inheritance
- → Hybrid inheritance

Single inheritance



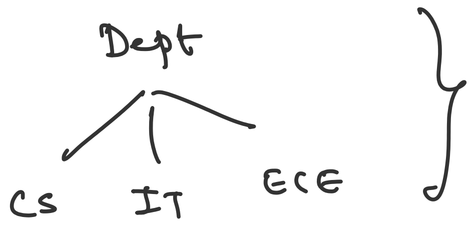
} single level inheritance

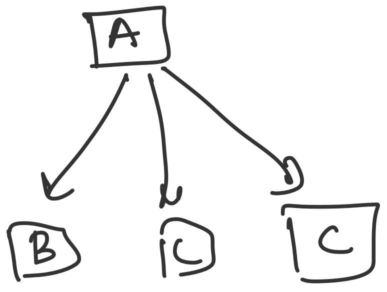
Multi level inheritance



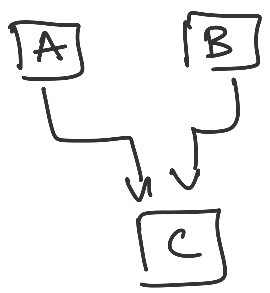
$\}$ More than one levels are there

Hierarchical inheritance



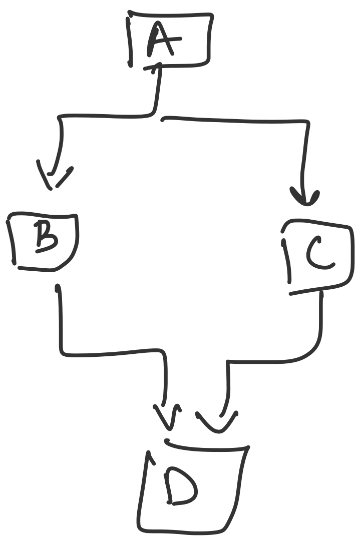


Multiple inheritance



Java does not support multiple inheritance.

Hybrid



Java does not support hybrid inheritanc