UML diagram

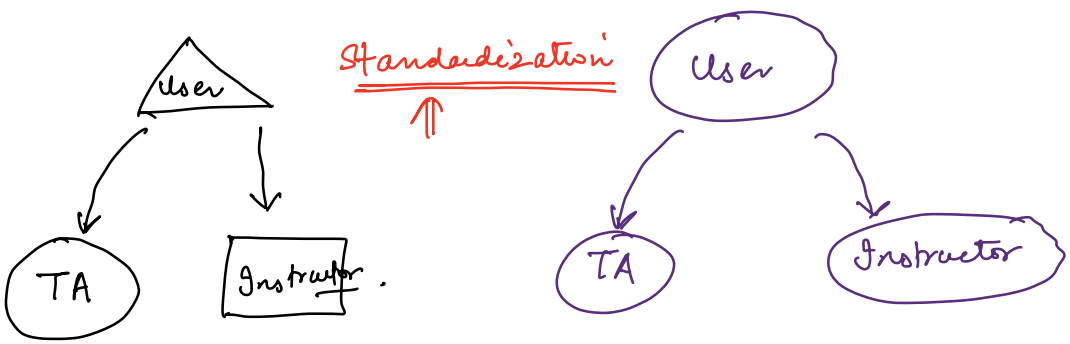
Association ← class diagram
↓ ↓ use-case diagram.
Aggregation composition

Product Manager } understanding the req,
clients } demos.
Architects } design approval
Team lead } growth
Business Stakeholders. }

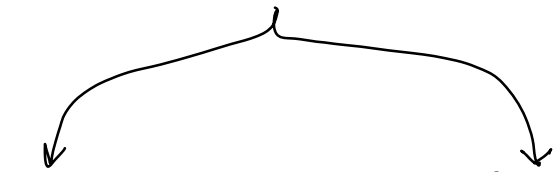
words — email
text
meeting

Misunderstanding, ambiguity

Picture → 1000 words.
↓
flowcharts / diagrams. / memes.



UML : Unified Modelling Language



Structural

codebase is structured.

Behavioral

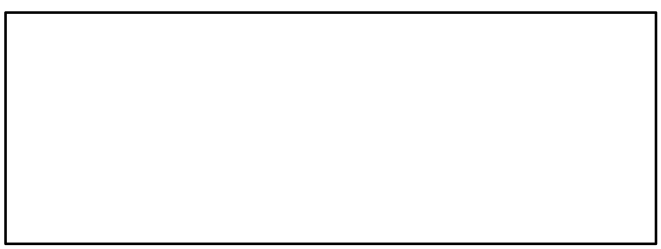
- how your system works & interacts
- what all features are supported.

class diagram	use-case diagram
Package	activity
component	sequence
	state.

use-case diagram

- diff features / functions / actions supported by system.
- who are the users.

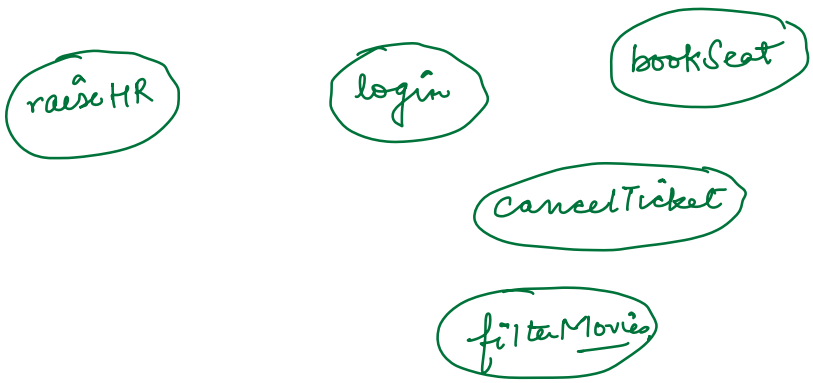
(I) system boundary.



- represents the scope of system
- It doesn't include outsourced features.

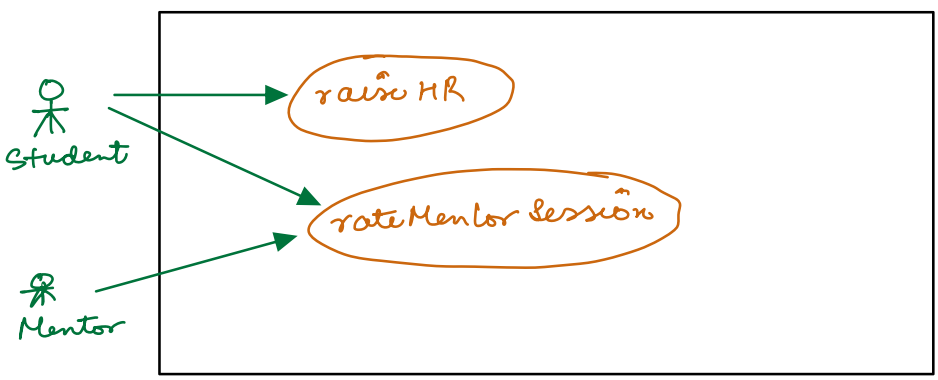
II Use-cases

- features / functions / actions
- must always be a verb
- represented by an oval.



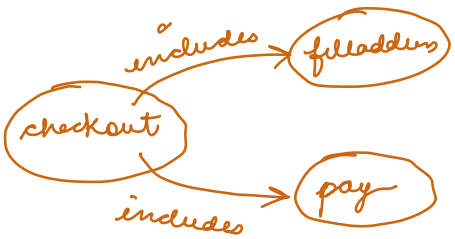
III Actors

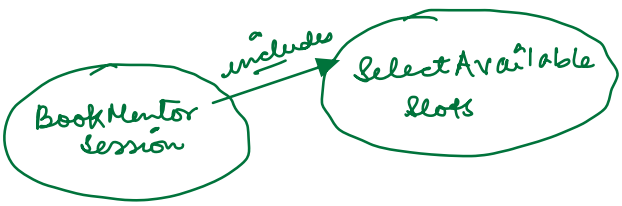
- people who use a particular use-case
- must be noun
- stick diagram:.



IV Includes

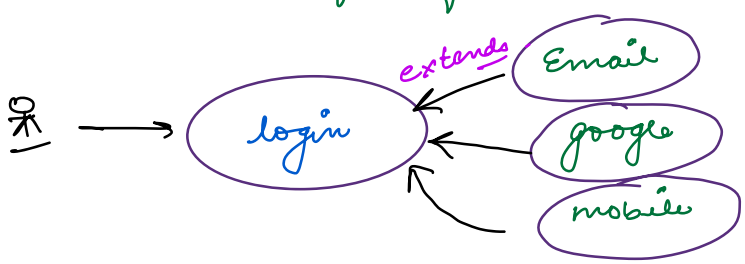
```
checkout() {
    filladdress();
    pay();
}
```

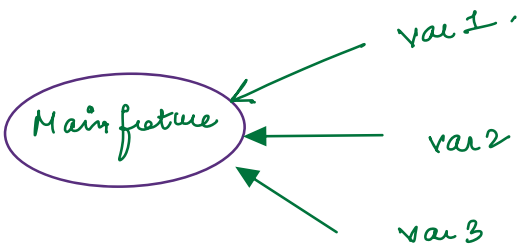




V extends

- if a feature has multiple variants.





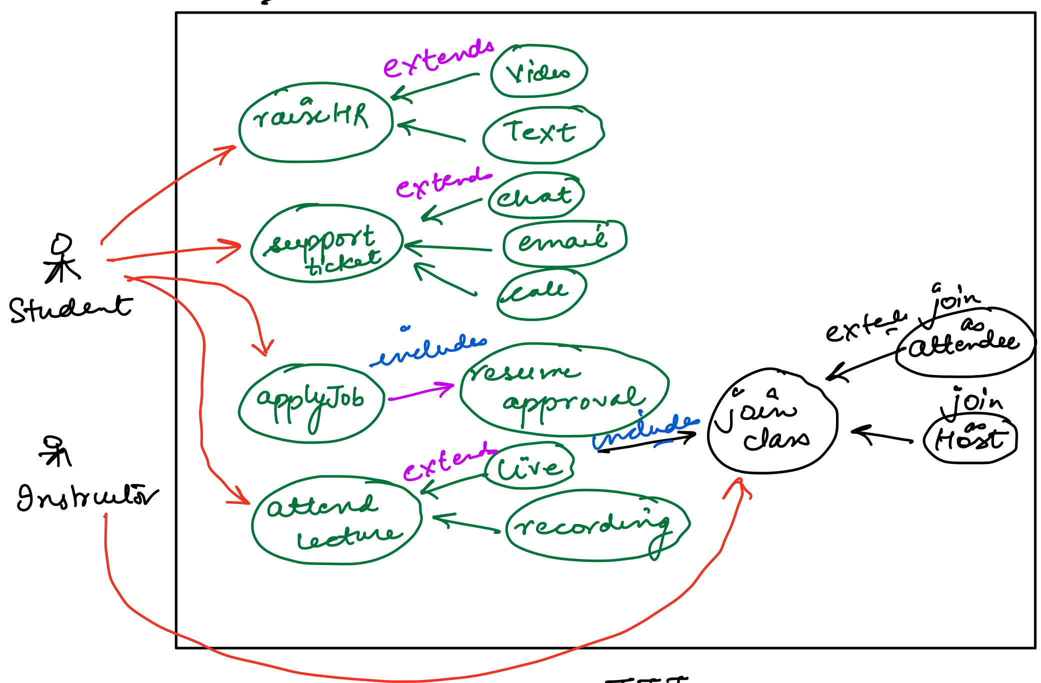
Draw a use-case diagram

I — 5 use-cases
II — 2 actors
III — 1 use case : includes
IV — 1 use-case : extends.

] Scaler.

Break: 10:06 pm

use-case.

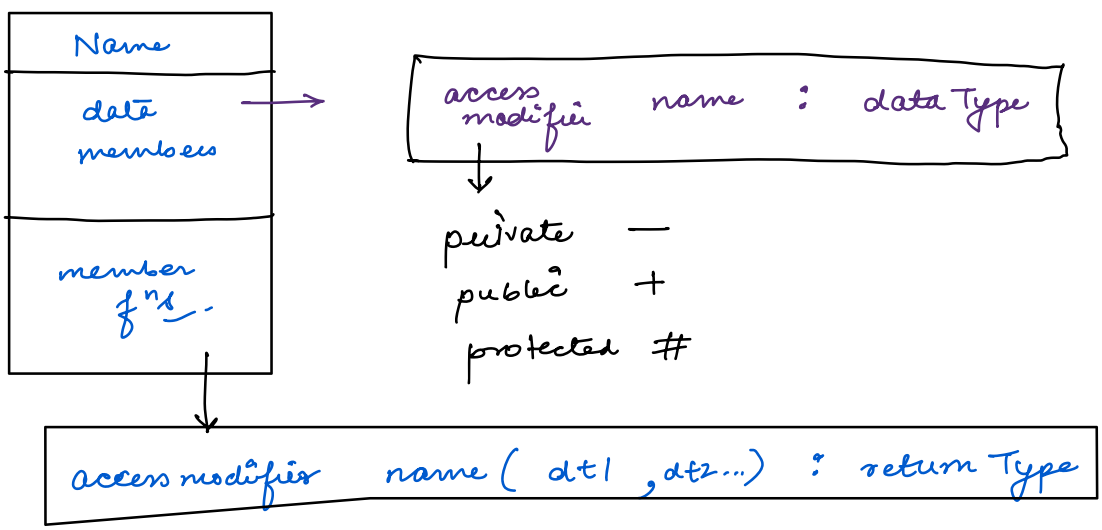


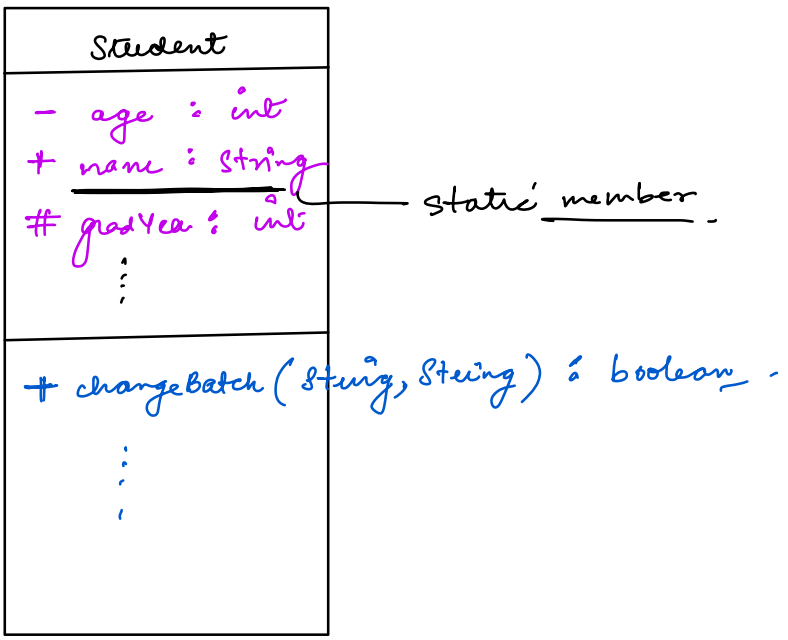
4 case-studies → TTT, parkinglot, BMS, splitwise.

Class diagram

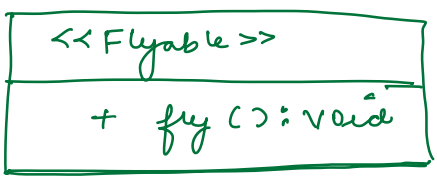
- represent diff entities
  - classes
  - interfaces
  - Abstract classes
  - enums.
- represent relationship b/w entities:.
  - implement interfaces
  - extend classes
  - having another class obj/ref as an attribute.

class.

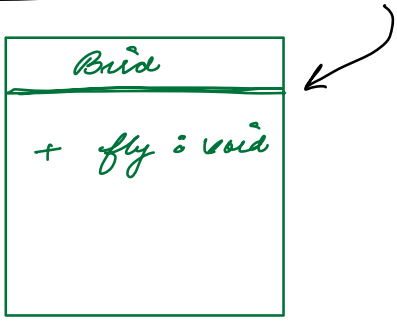




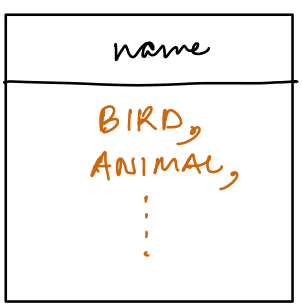
Interface



Abstract : Italic

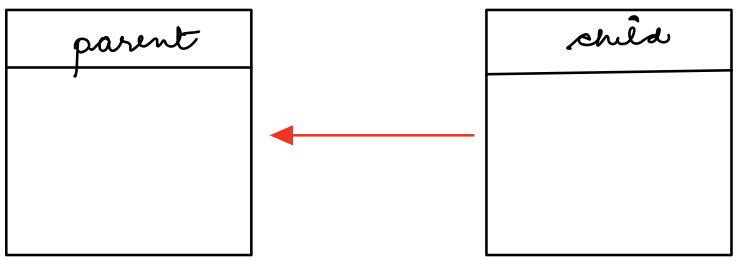


Enum.



How to represent relationships?

Interface implementation
extension of classes.



Animal

Dog

<< Flyable >>

Sparrow

Association

having another class an attribute

Aggregation

- weak assoc
- collecting.
- contained entity has an independent existence

Composition

- strong assoc
- to create
- contained & container are more dependent.

container.

classA {

classB.

} contained

```
Order {
    Customer customer;
    OrderItem oi;
}
```

aggregation

```
Group {
    User user;
}
```

```
OrderItem {
    String Item;
    int quantity;
    :
}
```

composition

```
Post {
    Comment comment;
}
```

```
Group {
    List<User> users;

    addUser(User user) {
        users.add(user);
    }
}
```

```
Post {
    List<Comment> comments;

    addNewComment(String input) {
        Comment c = new Comment();
        comms.add(c);
    }
}
```

Scala Library

```
Library {
    List<Book> books;
}
```

aggregation

composition

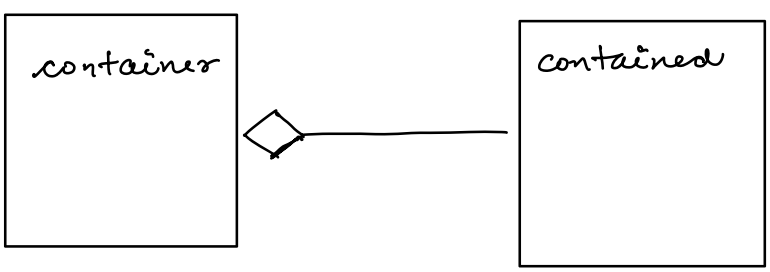
Association

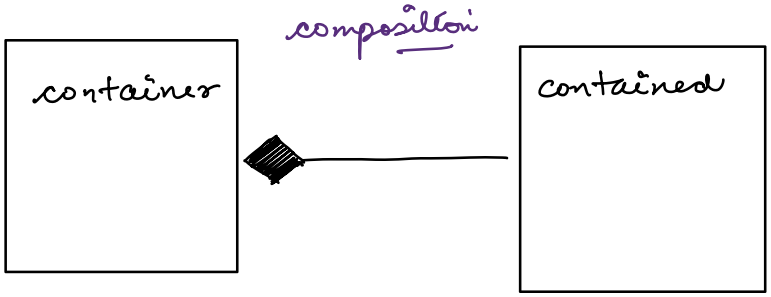
Academy ⇄ SST

SSB

DSMC

Aggregation





agg

Show show {

Movie movie;

}

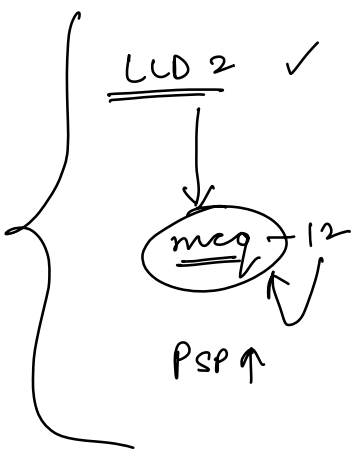
Post ← comment

Booking booking {

stay

payment payment;

}



{ wed / sunday contest ✓

mock : Backend LLD 1

3 weeks .