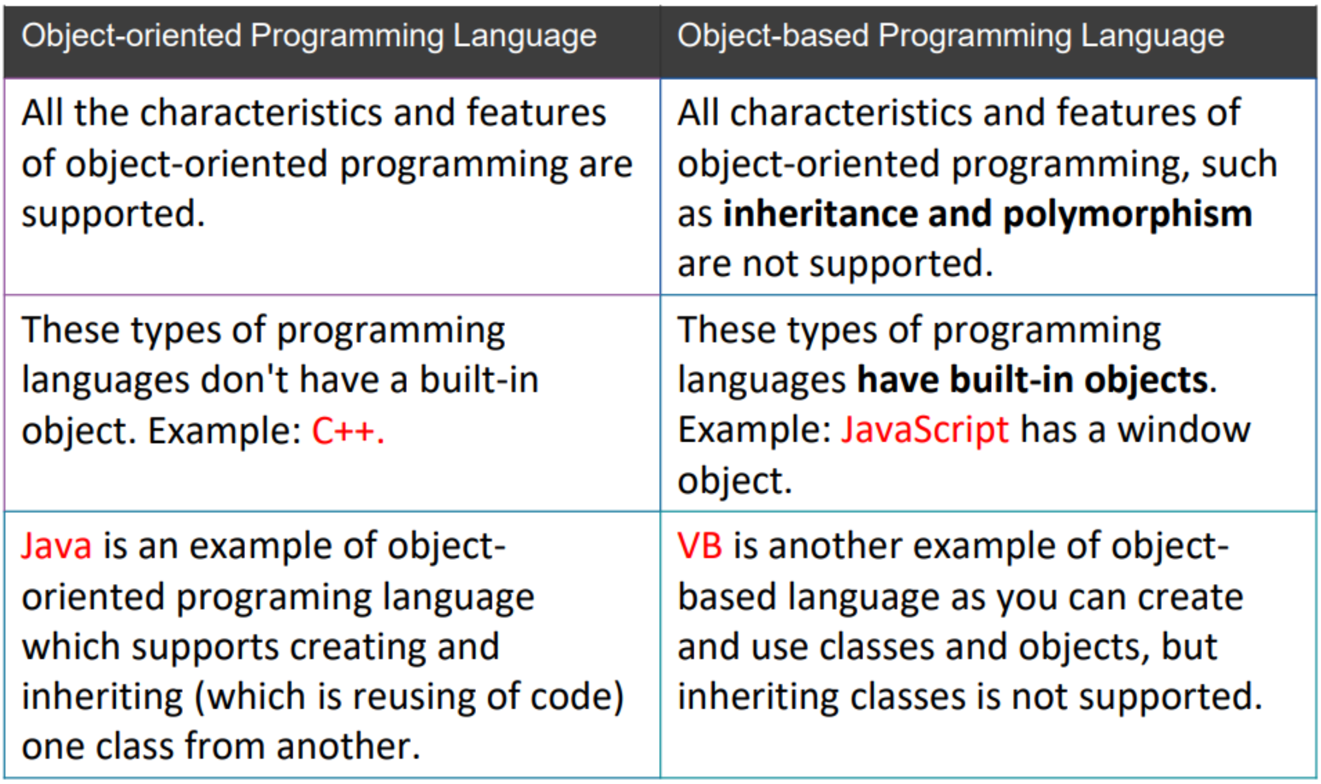
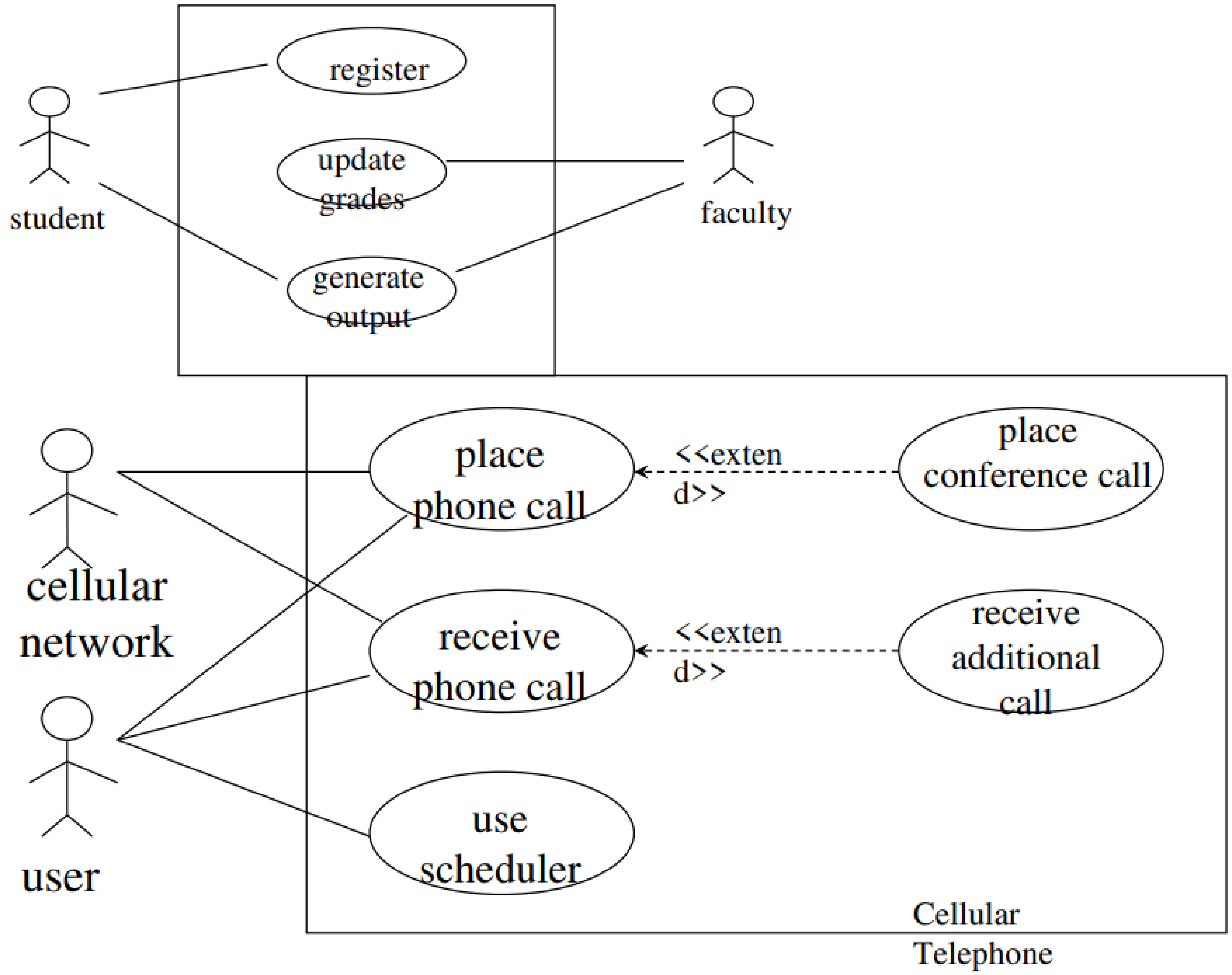
Unit 1

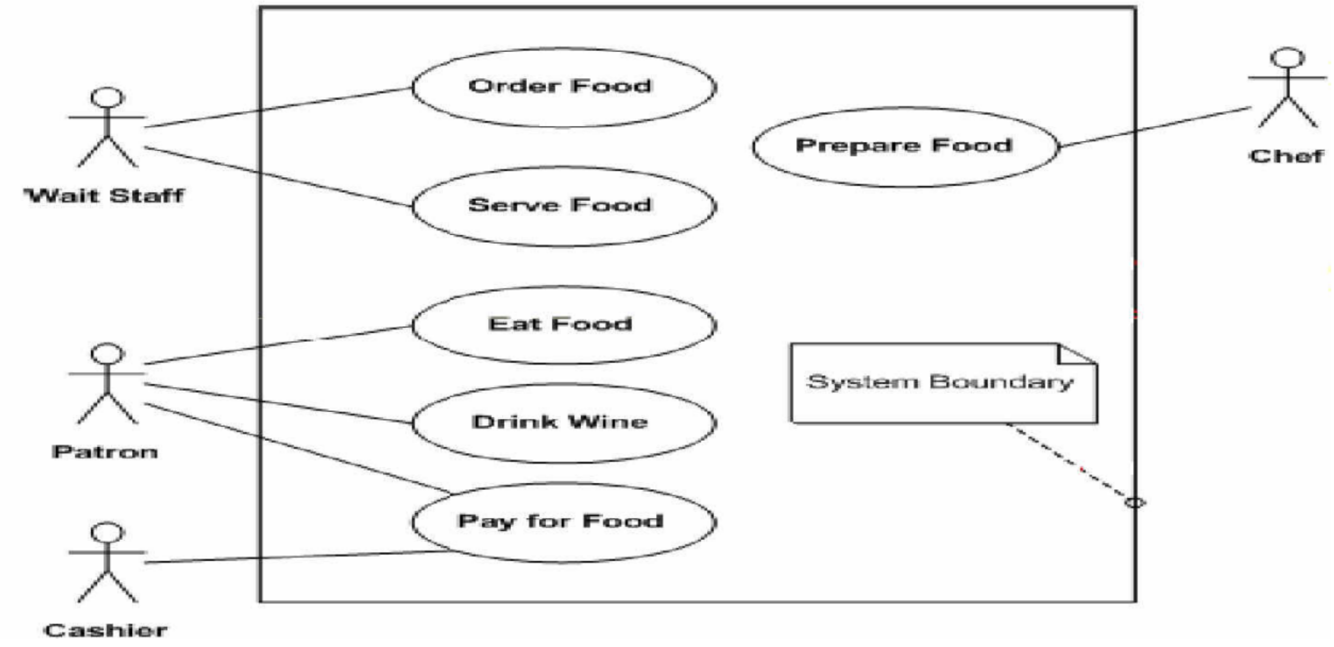


Object-oriented Programming Language	Object-based Programming Language
All the characteristics and features of object-oriented programming are supported.	All characteristics and features of object-oriented programming, such as **inheritance and polymorphism** are not supported.
These types of programming languages don't have a built-in object. Example: C++.	These types of programming languages **have built-in objects**. Example: JavaScript has a window object.
Java is an example of object-oriented programing language which supports creating and inheriting (which is reusing of code) one class from another.	VB is another example of object-based language as you can create and use classes and objects, but inheriting classes is not supported.

Use Case Diagram:



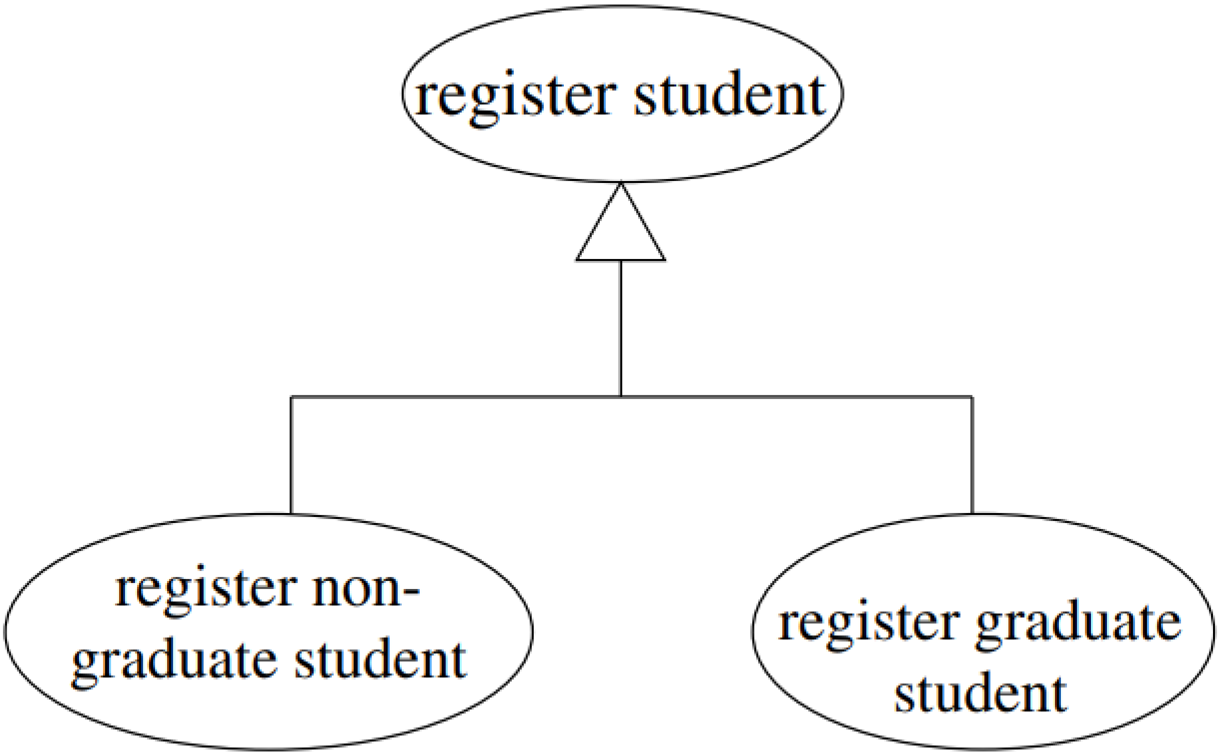
The things can also be disconnected



The various arrow things:

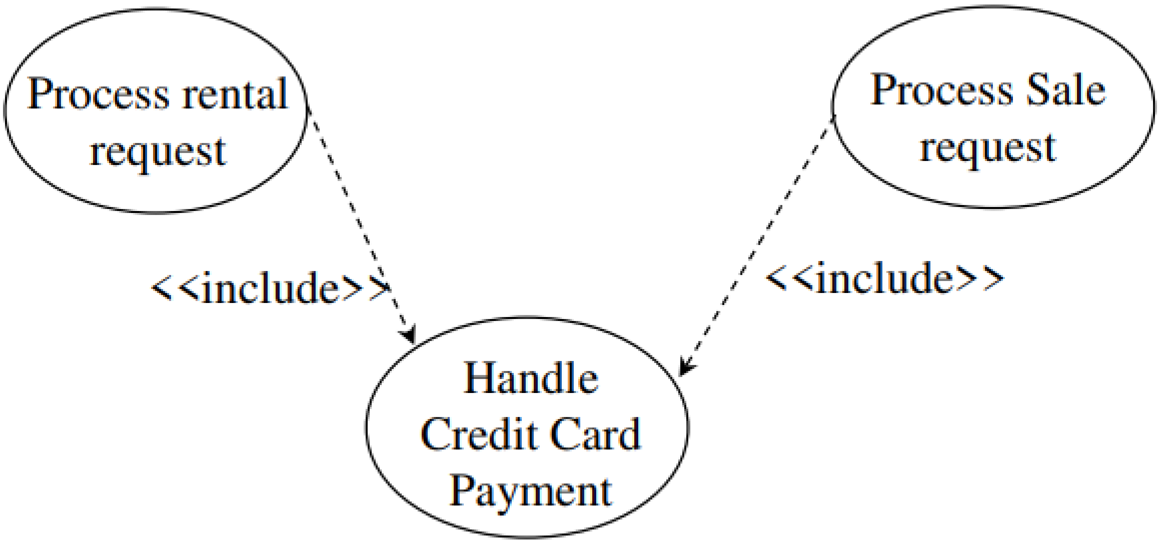
Generalization

- Hollow arrow
- Child inherits the parent's attributes and child can override the parent's stuff.



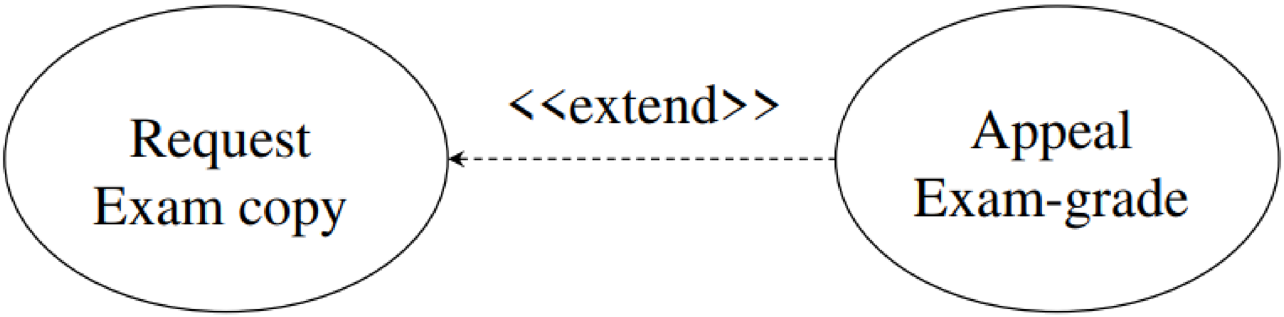
Include

- Dotted line and solid head `<<include>>`
- Base ---> include. Base uses the include thing into it. Here Base **REQUIRES** include.



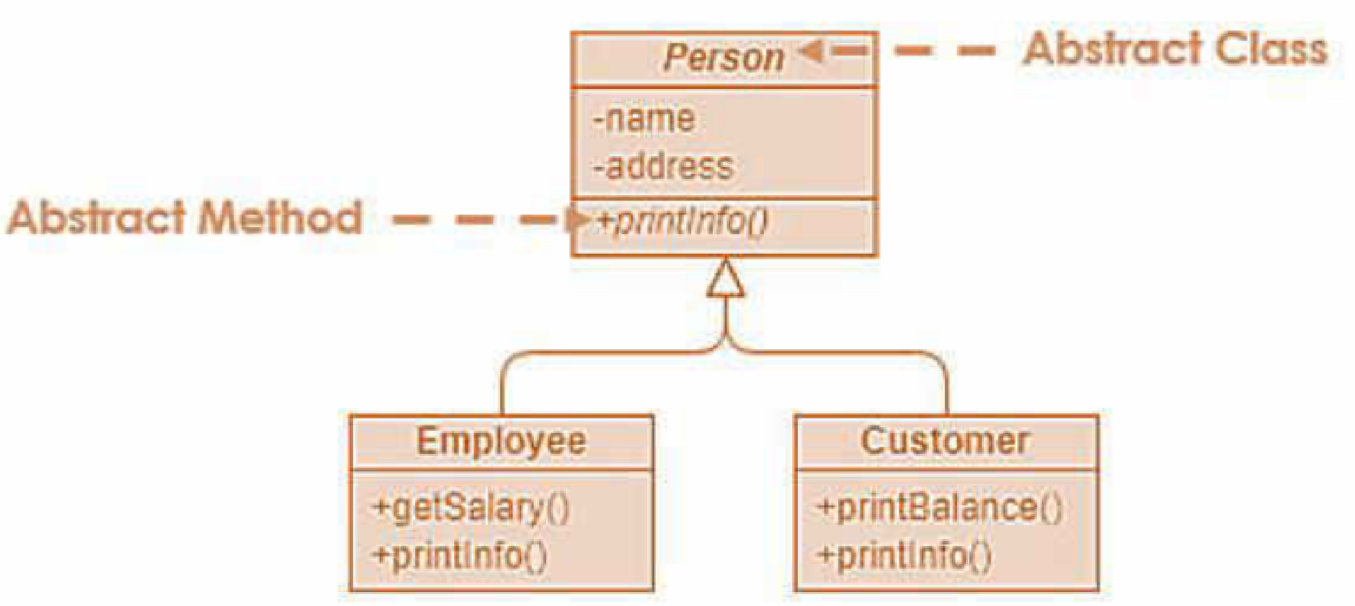
Extend

- Dotted line and solid head `<<extend>>`
- Base ---> exte. Base uses the exte thing into it. Here Base **MIGHT REQUIRE** exte.



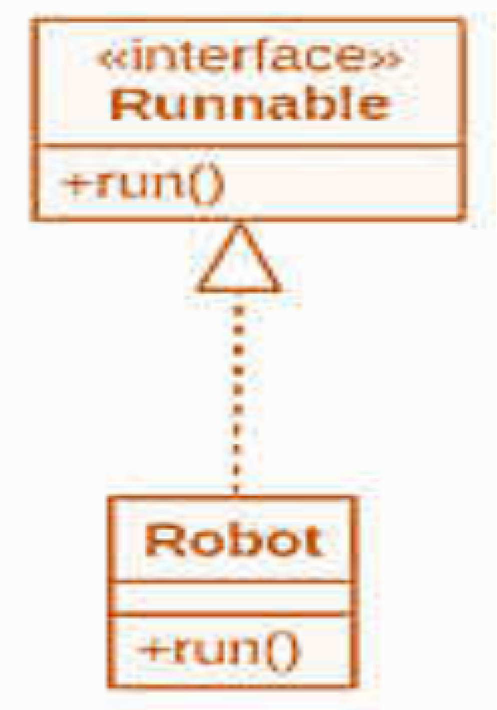
Abstraction

- Solid line hollow head
- Implementation of the Abstract class.



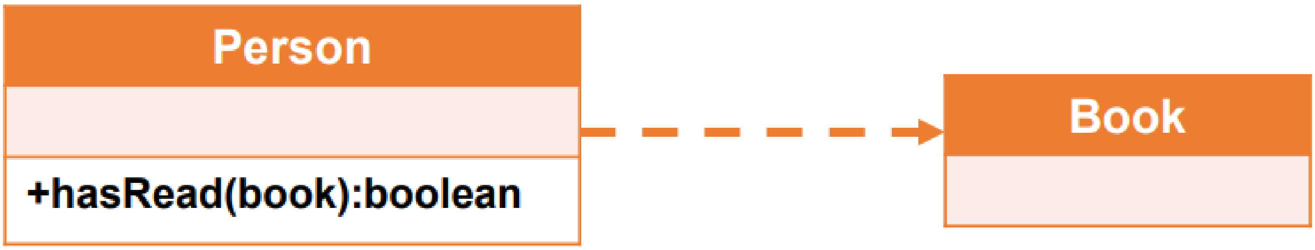
Realization

- Dotted line hollow head
- Implement of an interface.



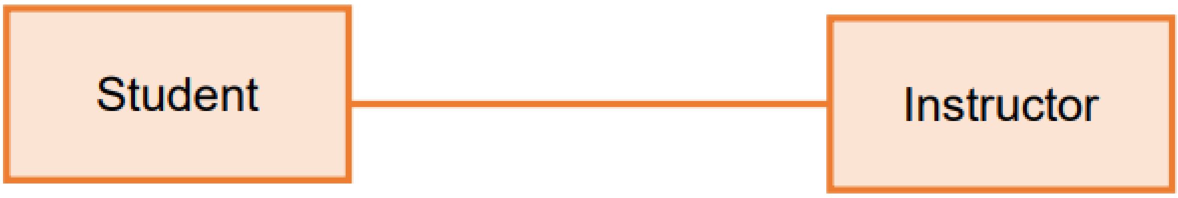
Dependency

- Dotted line and solid head
- One class uses the other inside

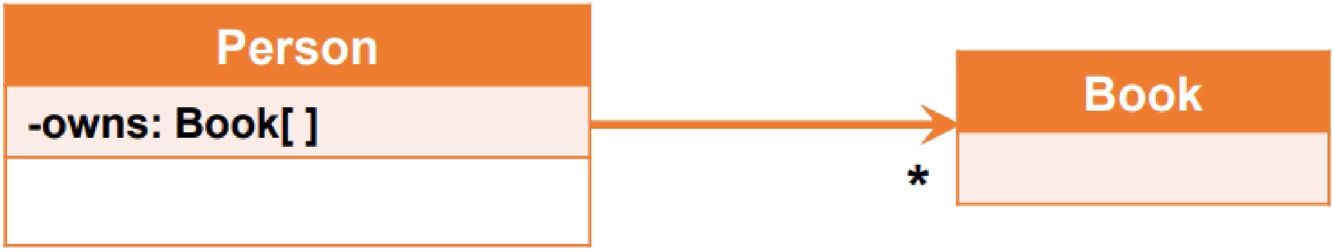


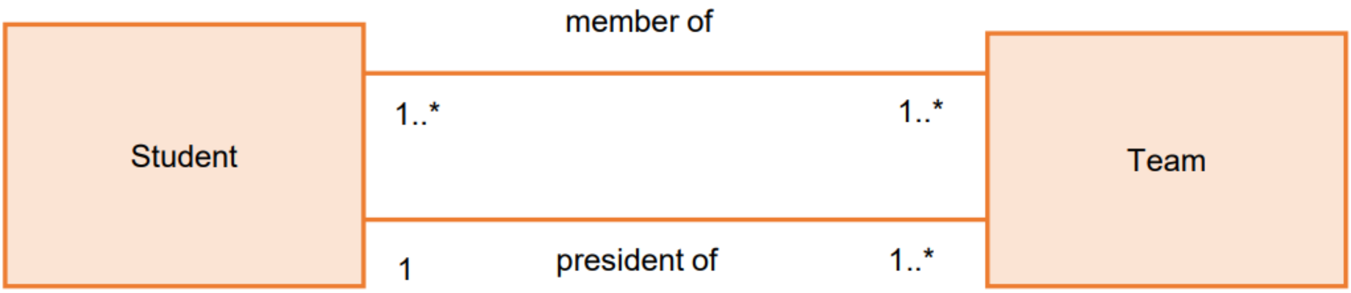
Association

- Solid line and no head*
- This is when 2 classes need to speak to each other or interact in some way.
- We can name the association. We can also have "role names".

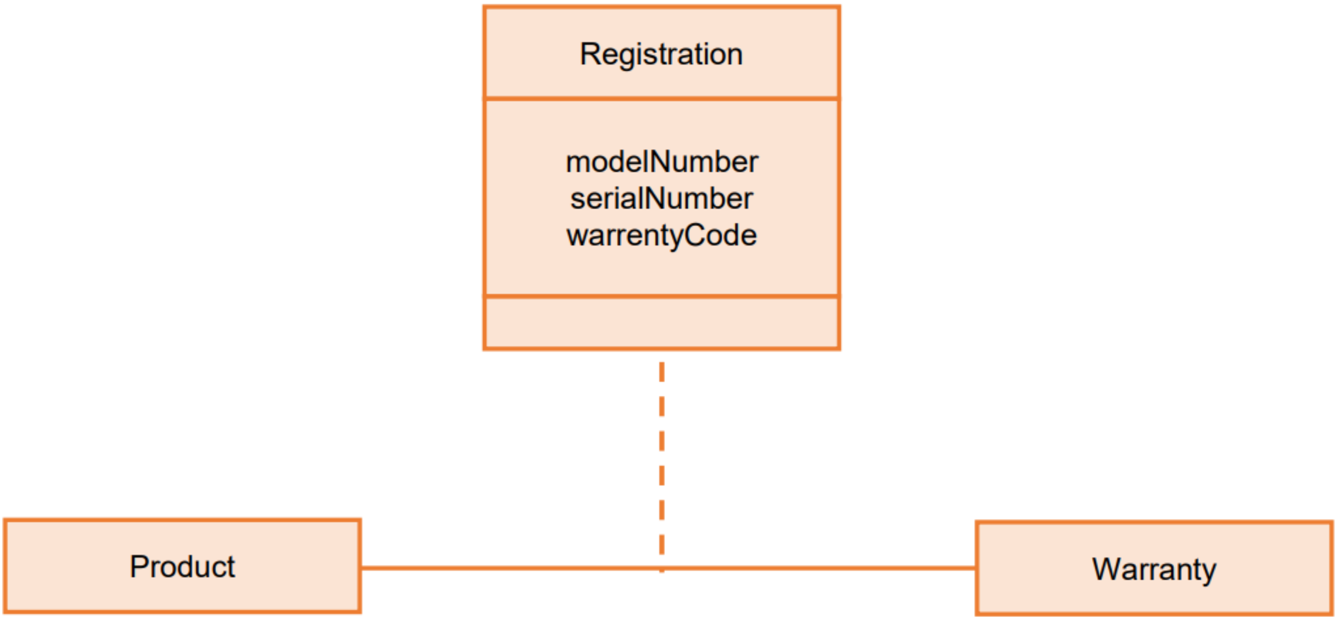


Can be unidirectional in which case solid line and regular arrow. Also multiplicity is a thing.





We can also have another class for the association.



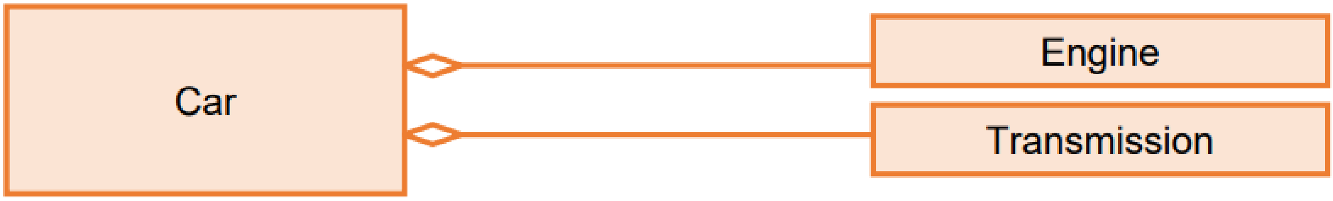
They can also self associate.

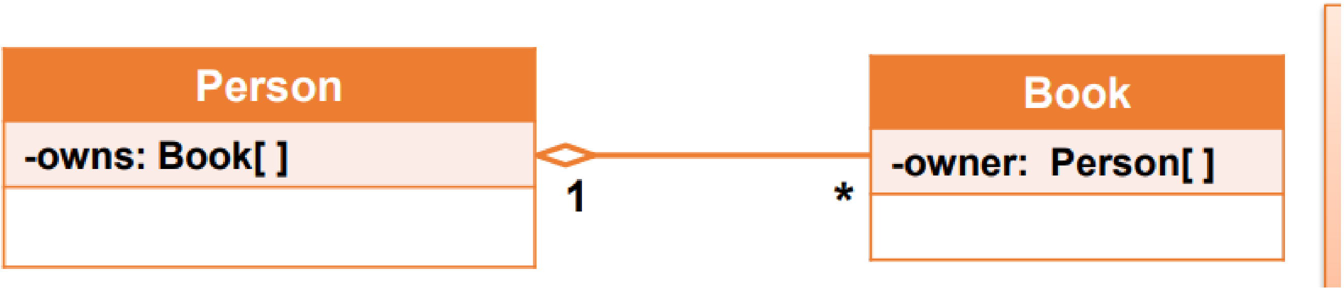
Aggregation

Solid line hollow diamond.

This is the "has-a" relationship.

One can exist without another.



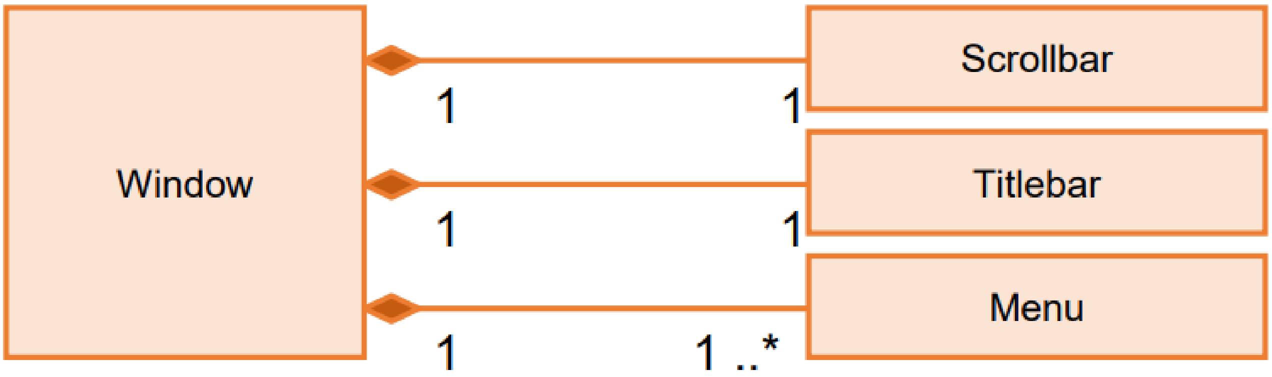


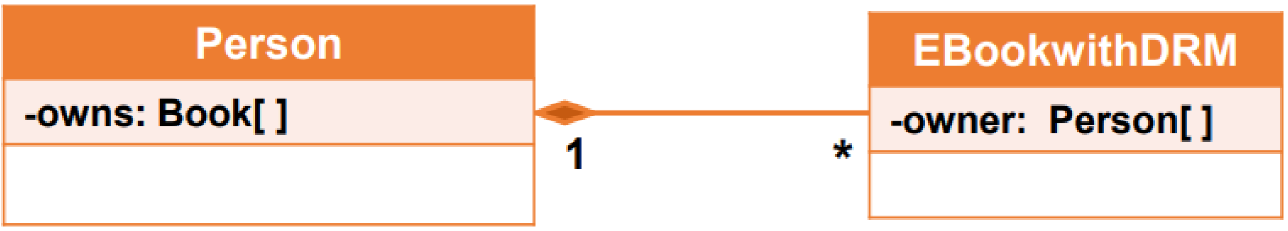
Composition

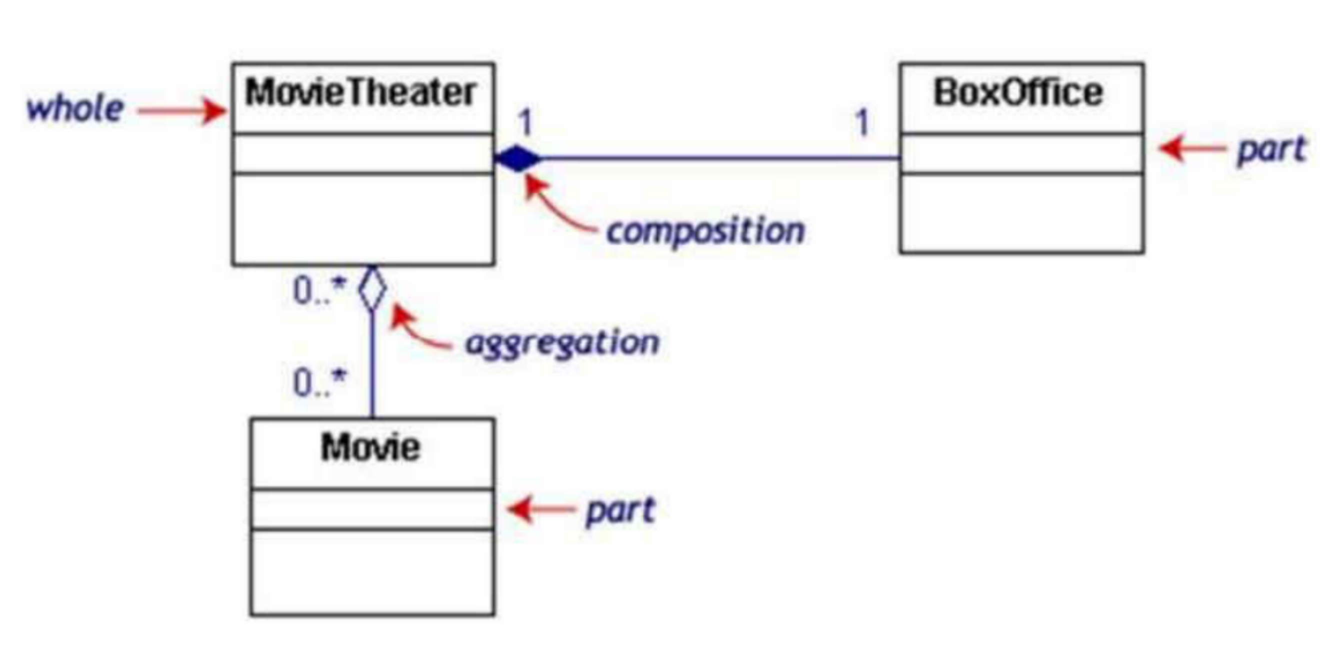
Solid line solid diamond.

This is also "has-a" relationship.

The entire exists or nothing.







If the movie theater goes away
so does the box office => composition
but movies may still exist => aggregation

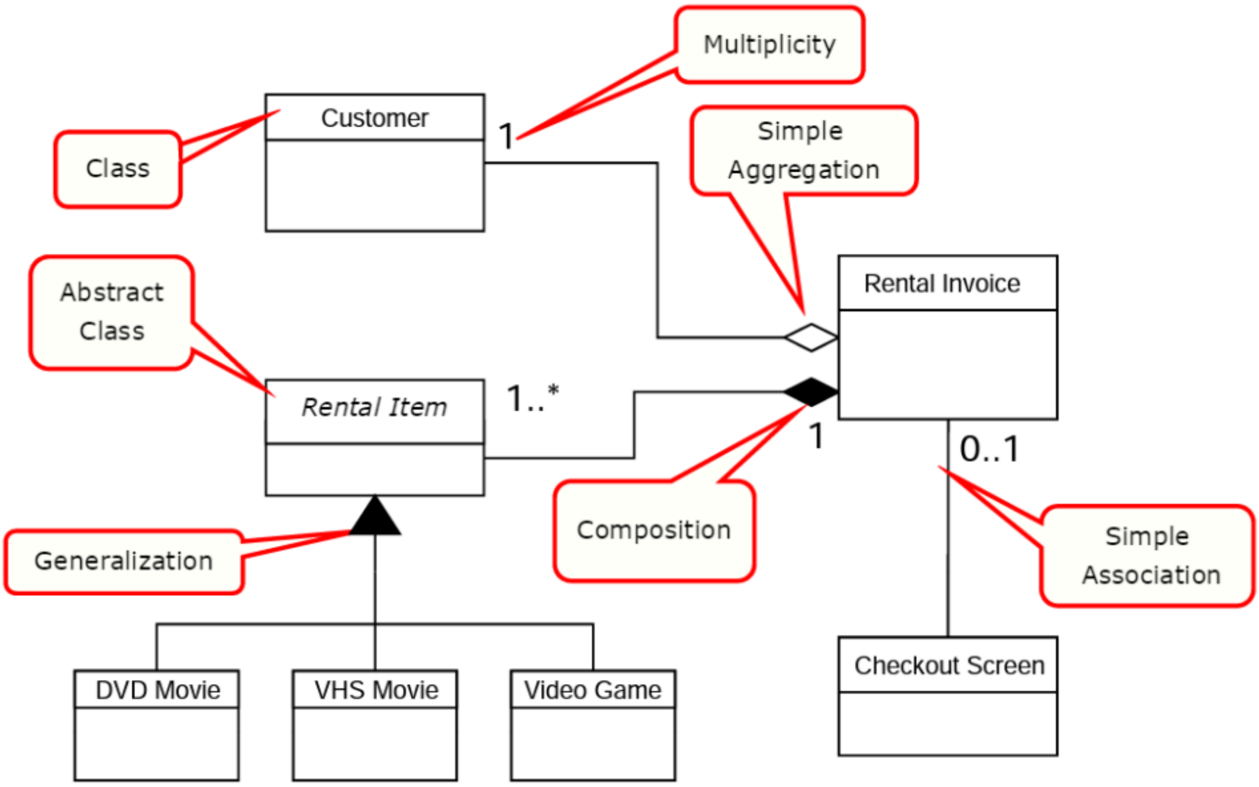
Enumeration

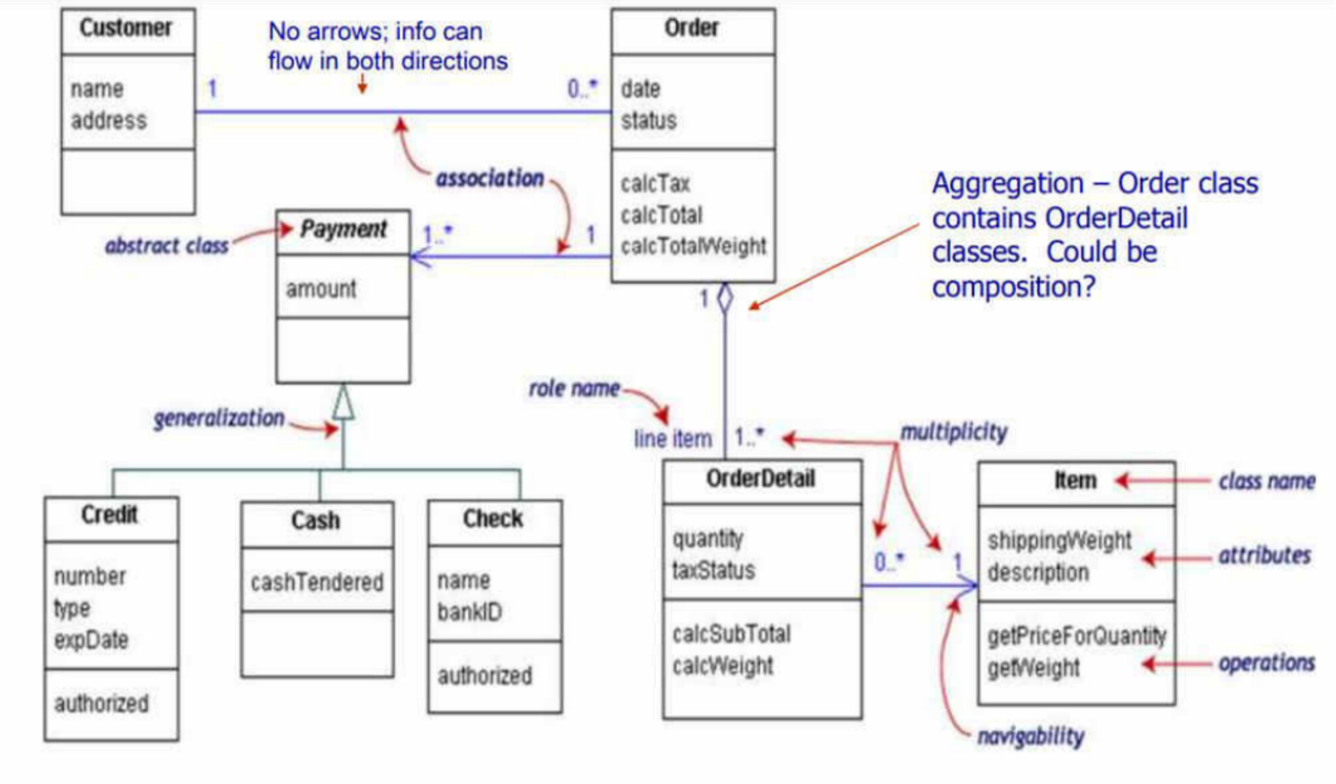
This is just a list of the same datatype.

Exception

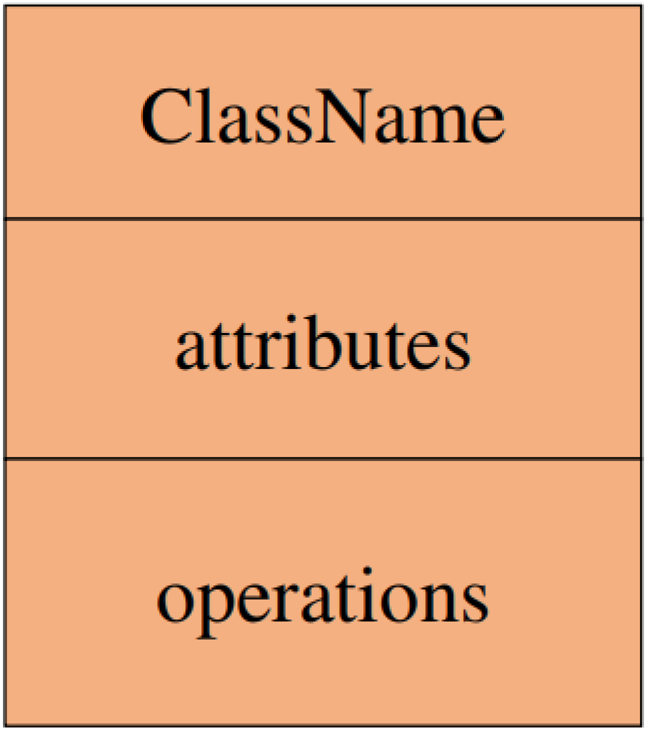
-_-

Sample Diagram:





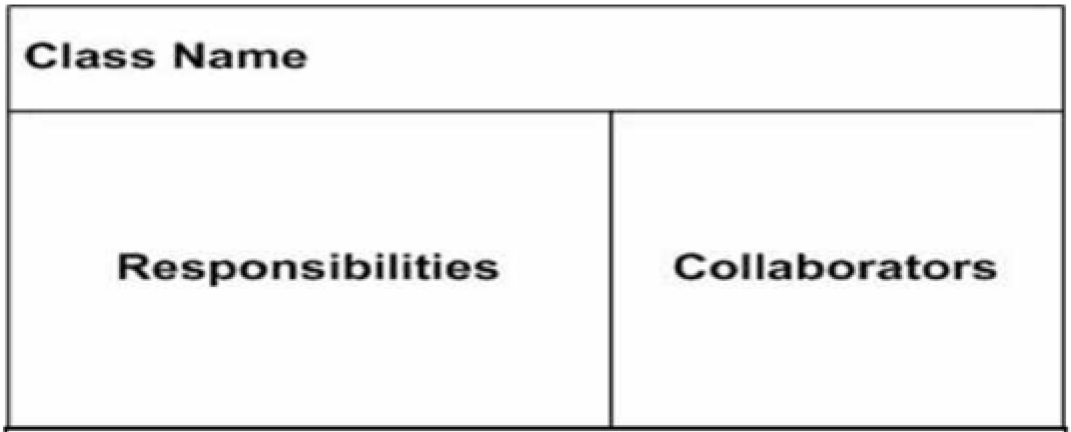
Class Diagram:



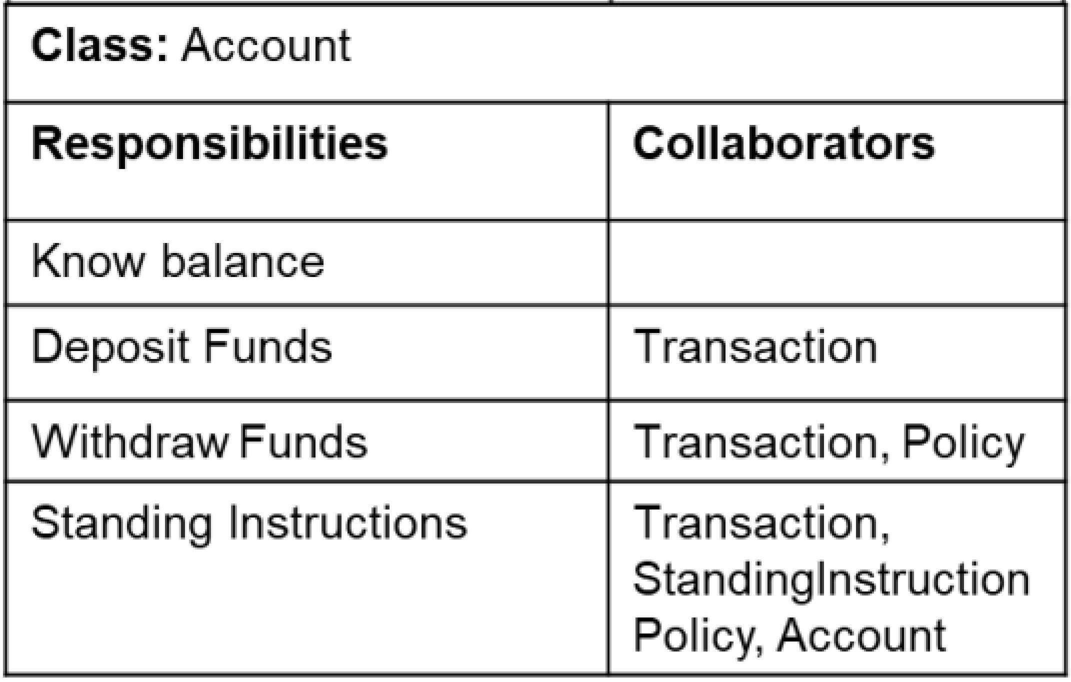
Derived attributes: `/<name>:<type> ex: /age:int`

```
+ : public
# : protected
- : private```
```

CRC (Class-Responsibility-Collaborators)



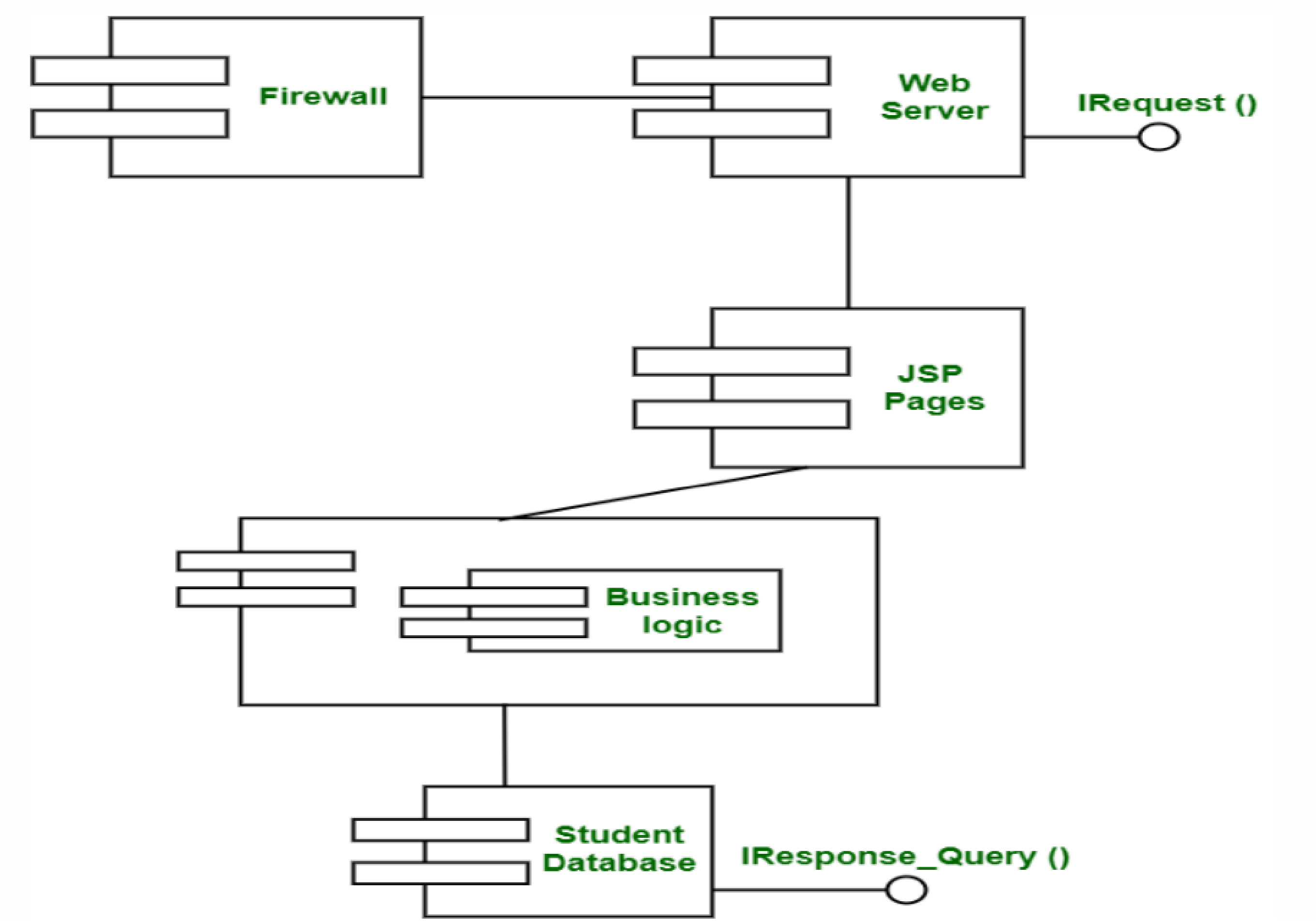
Class Name	
Responsibilities	Collaborators

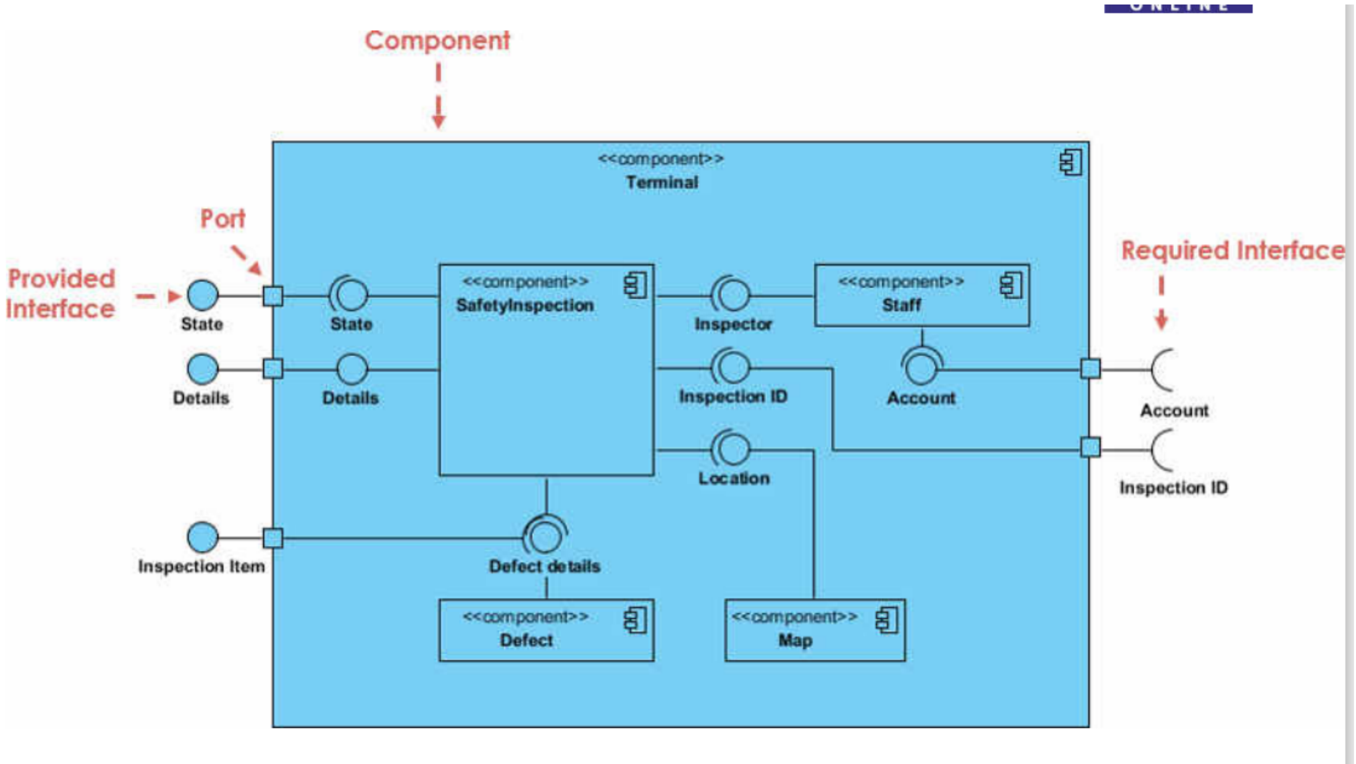


Class: Account	
Responsibilities	**Collaborators**
Know balance	
Deposit Funds	Transaction
Withdraw Funds	Transaction, Policy
Standing Instructions	Transaction, StandingInstruction Policy, Account

---

Component Diagram

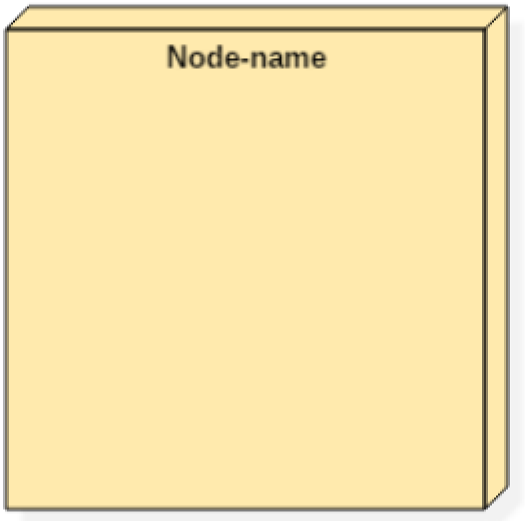


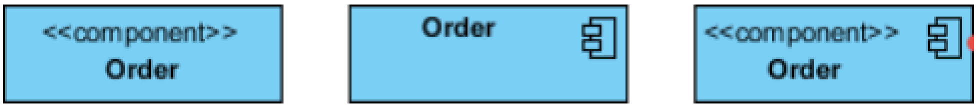


Refer for symbols

Half circle is the one that needs. Full circle is the one that gives.

A node is a thing that executes the components.





Interface in this diagram is a set of public features. Full circle.

Component has that internal and external view thing.

Usage Dependency

When one component requires another to work then it s a usage dependency.

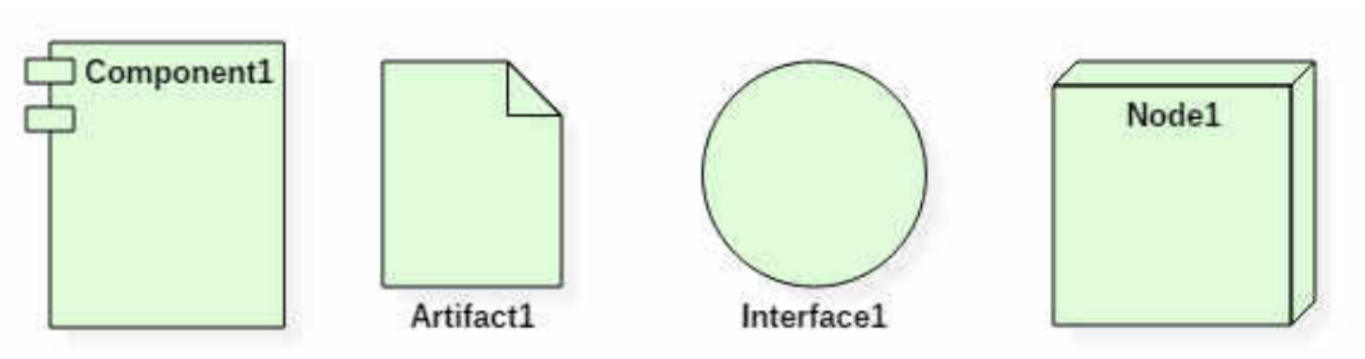
Dotted line and `<<use>>`

Port

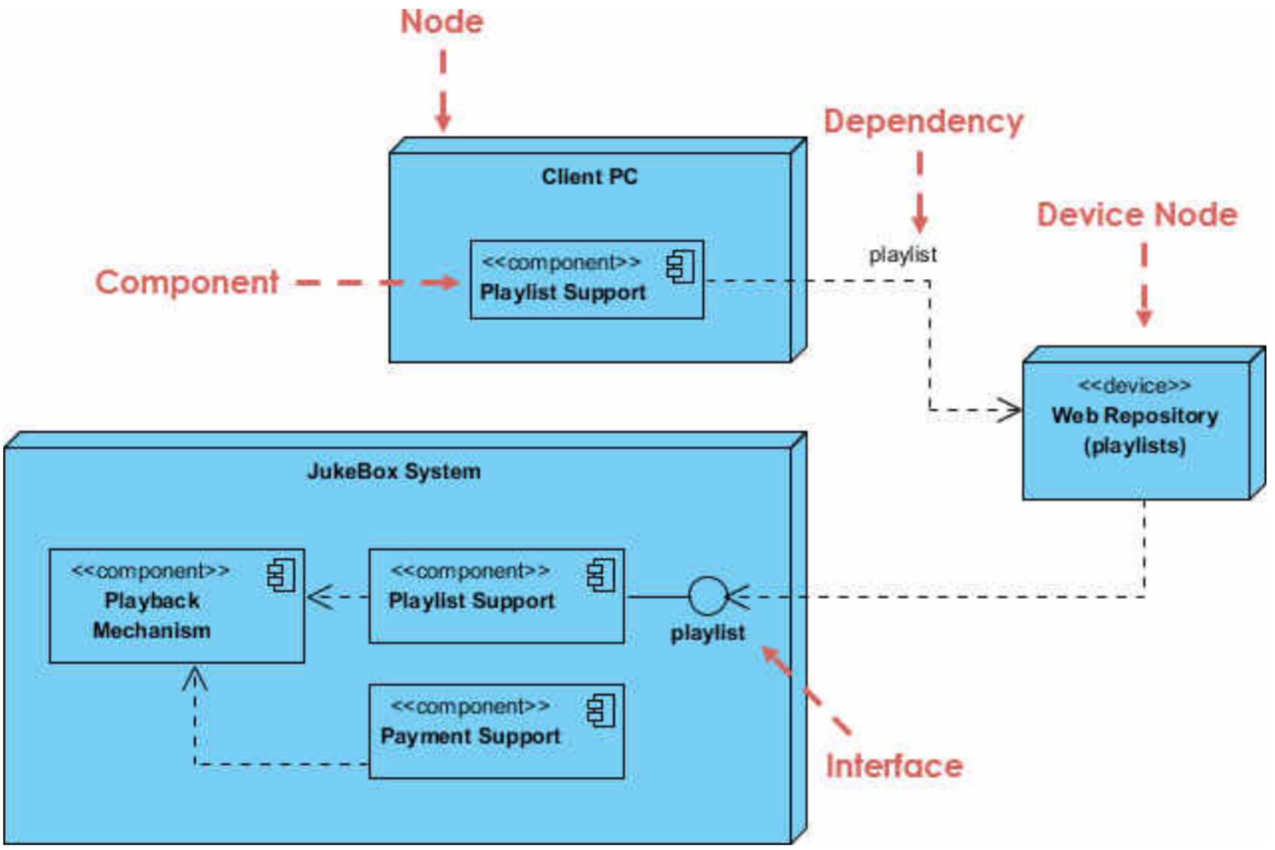
This is the access point between the component and the outside world.

Deployment Diagram

This uses component inside. So these are the symbols.

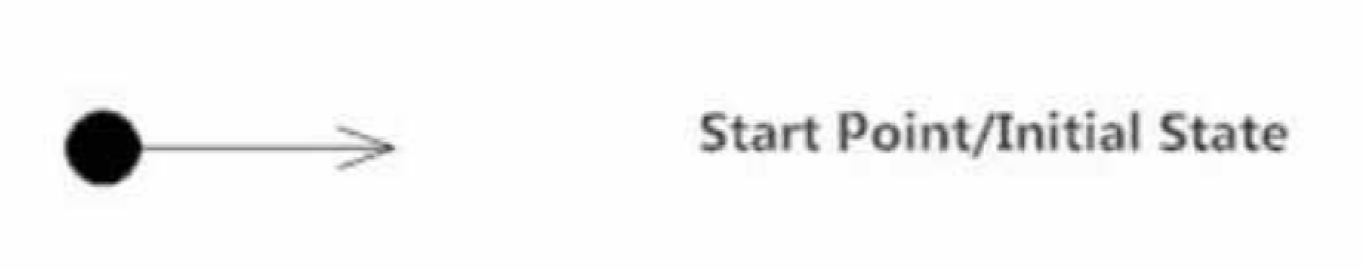


Artifacts are the things that are results from development. Configuration , exe, libraries and achieve files.

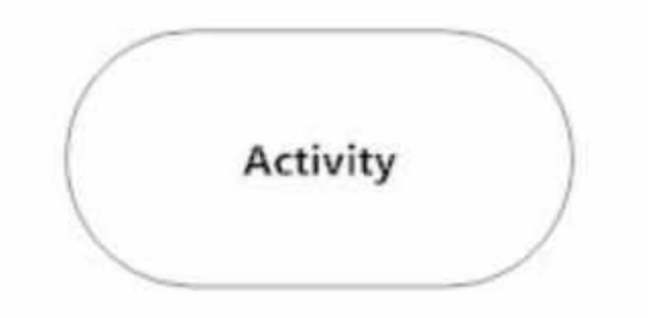


Activity Diagram

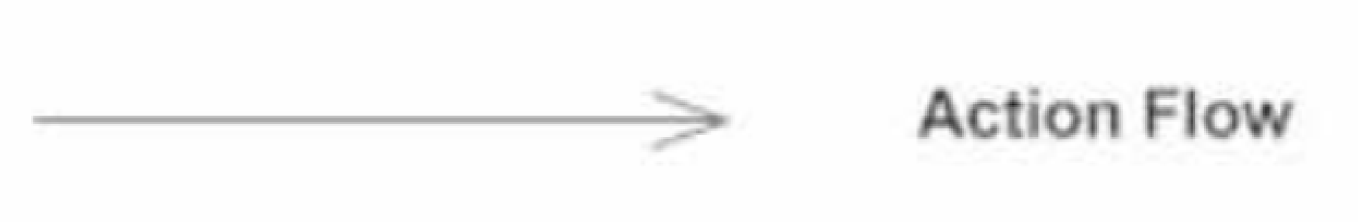
This is pretty much a flowchart type thingy



Activity / Action State non interruptible action.



Action Flow this is just an arrow.

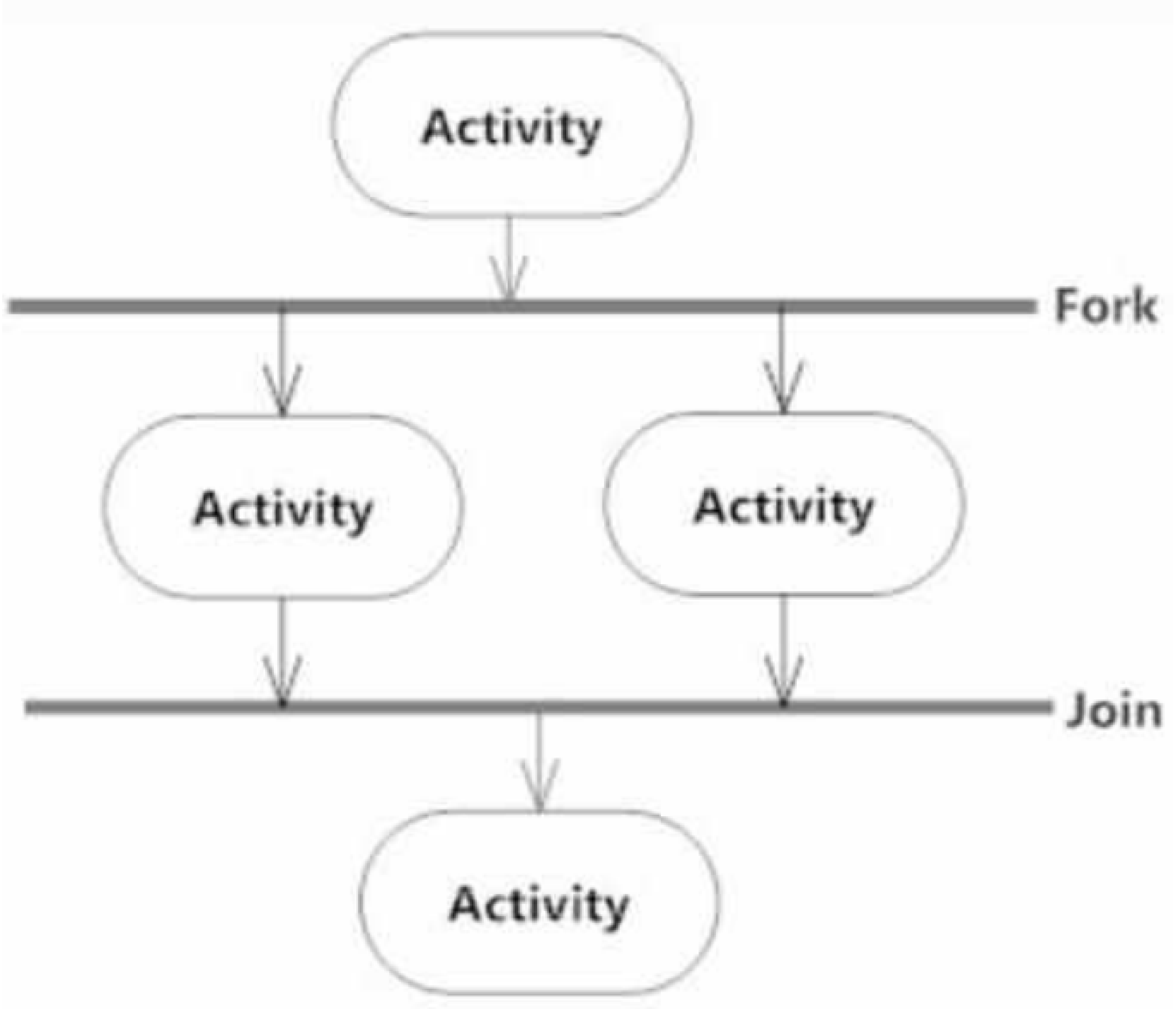


Object Flow: this is just the flow in the diagram

Decision: Symbol diamond

Guards: the condition on the arrow from a decision symbol

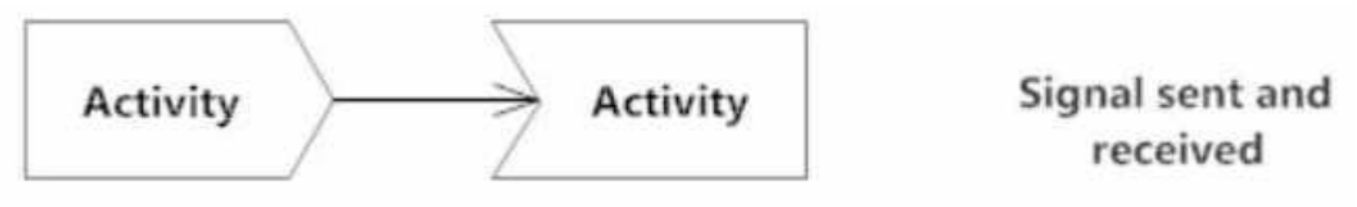
Synchronization / Fork



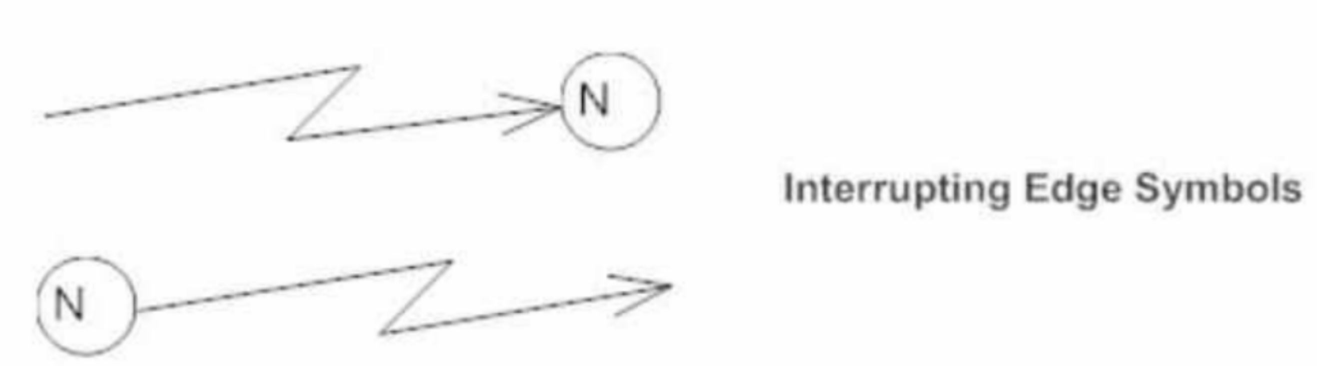
Time Event hourglass thingy

Merge -_- a fked looking fork

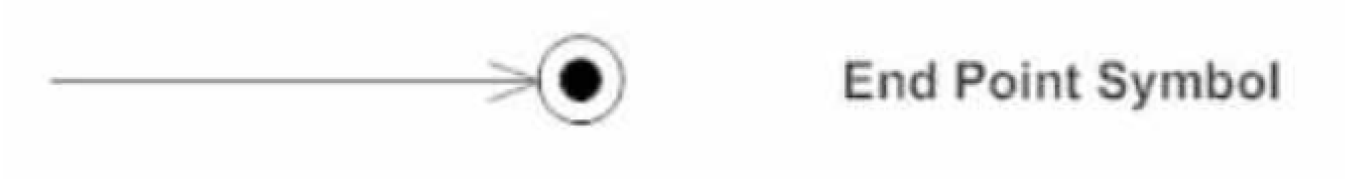
Sent and Received signals same just add signal masala



An interrupting edge : jagged line/ lightning / zig-zag

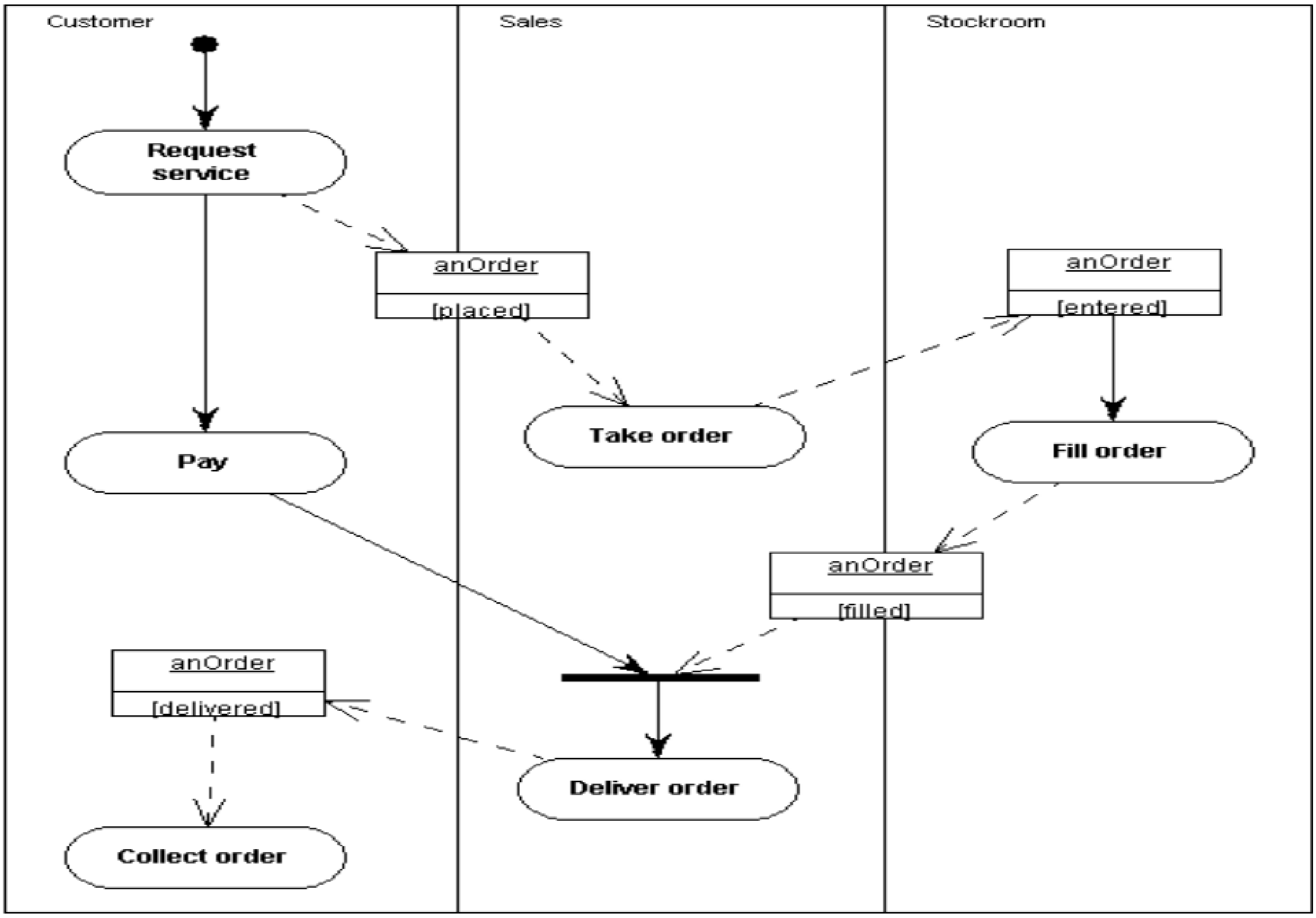


End Point -_-



Swimlanes

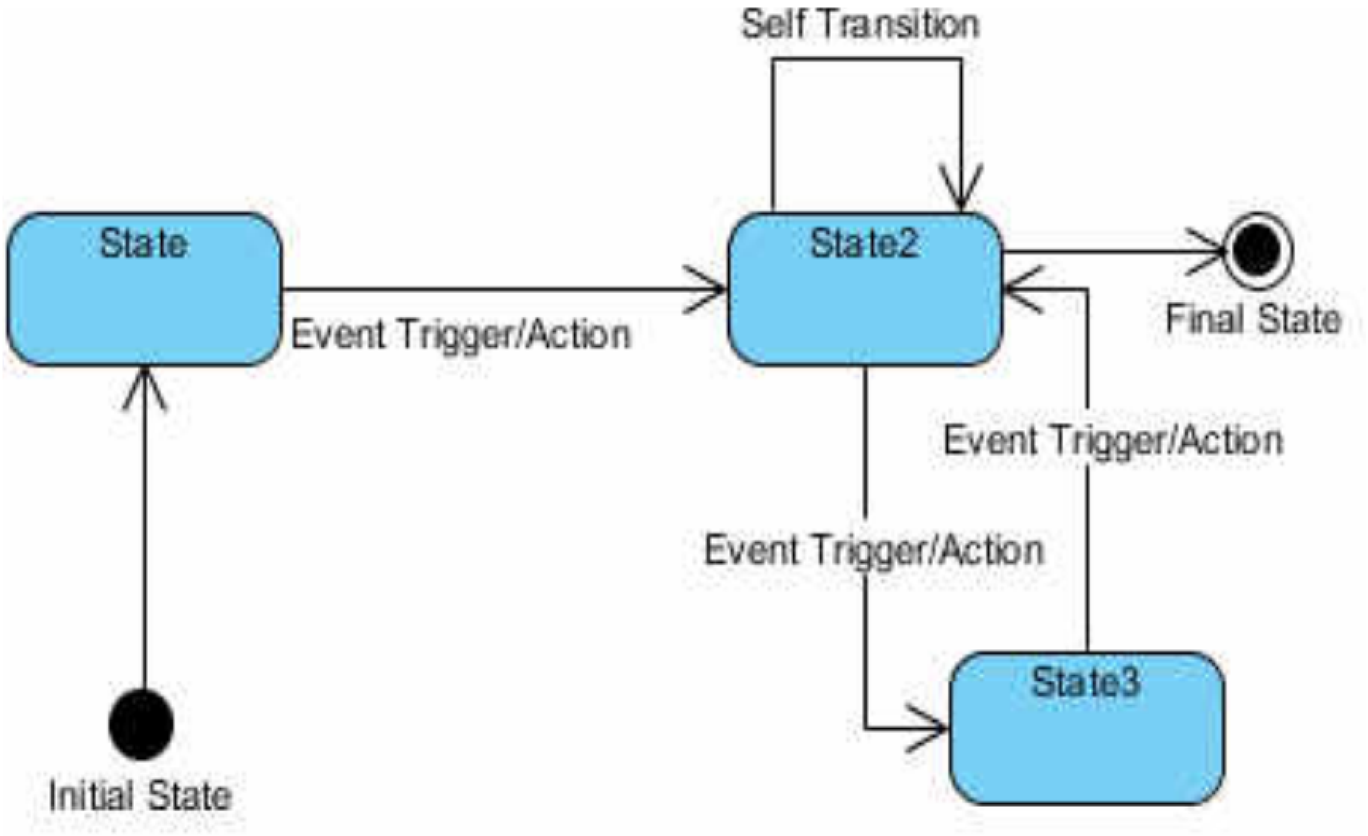
This is grouping activity diagrams into columns.



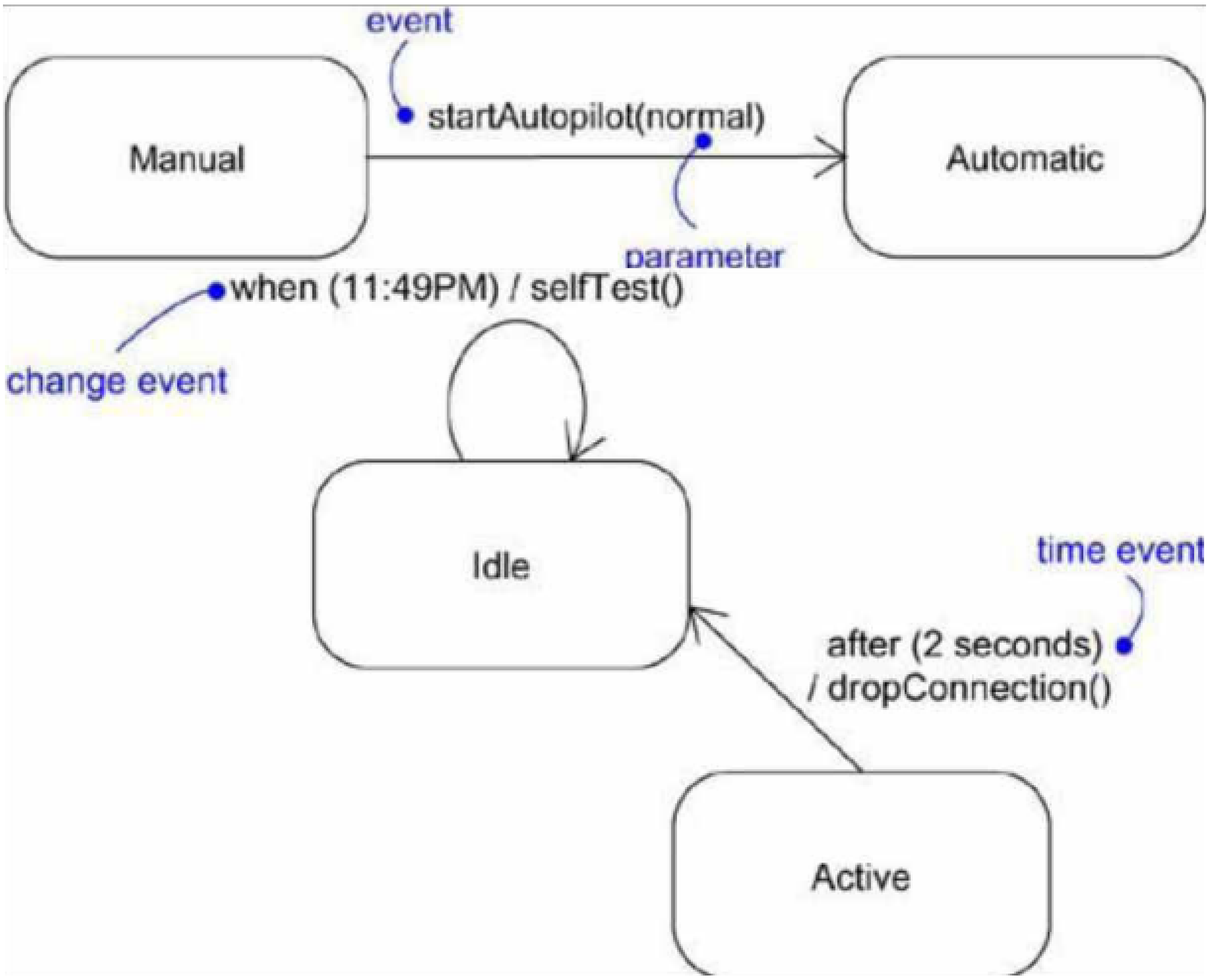
Behavior Modeling

State Diagram

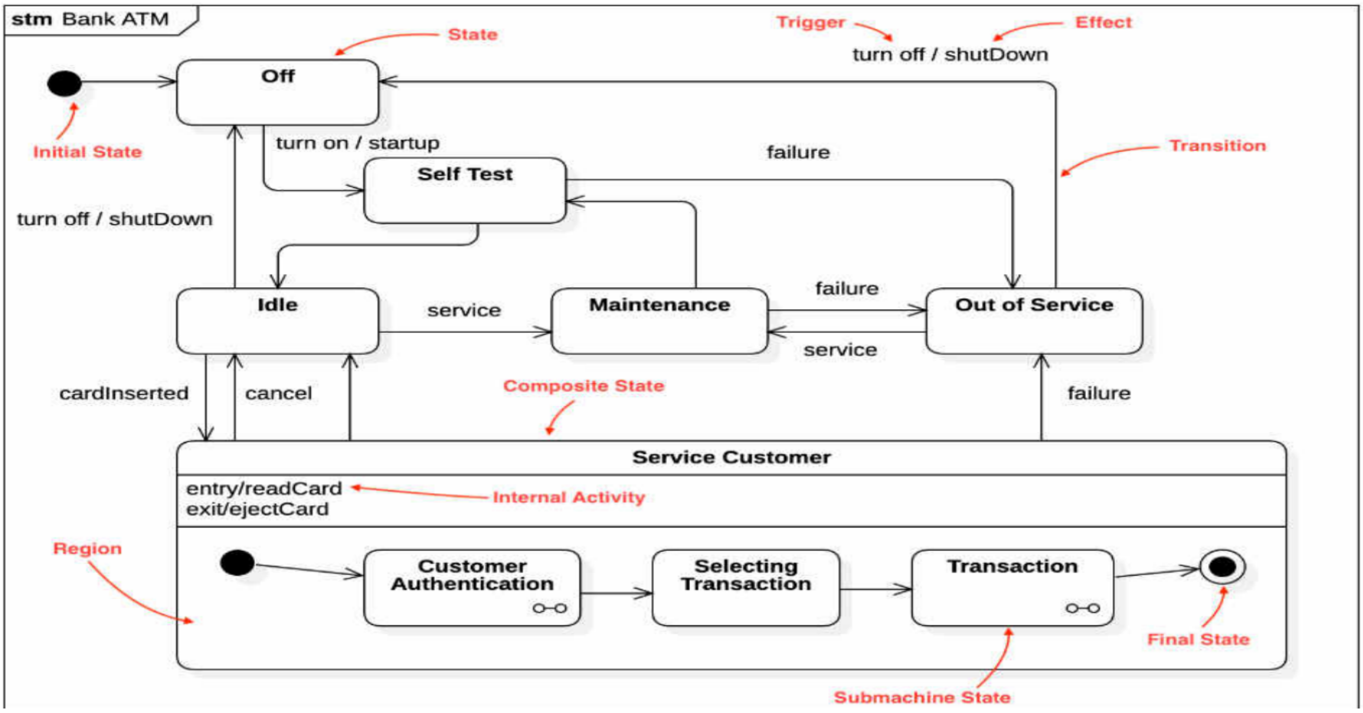
This shows the changes in the state in the program execution not the flow of commands.



Triggers: this is how things flow

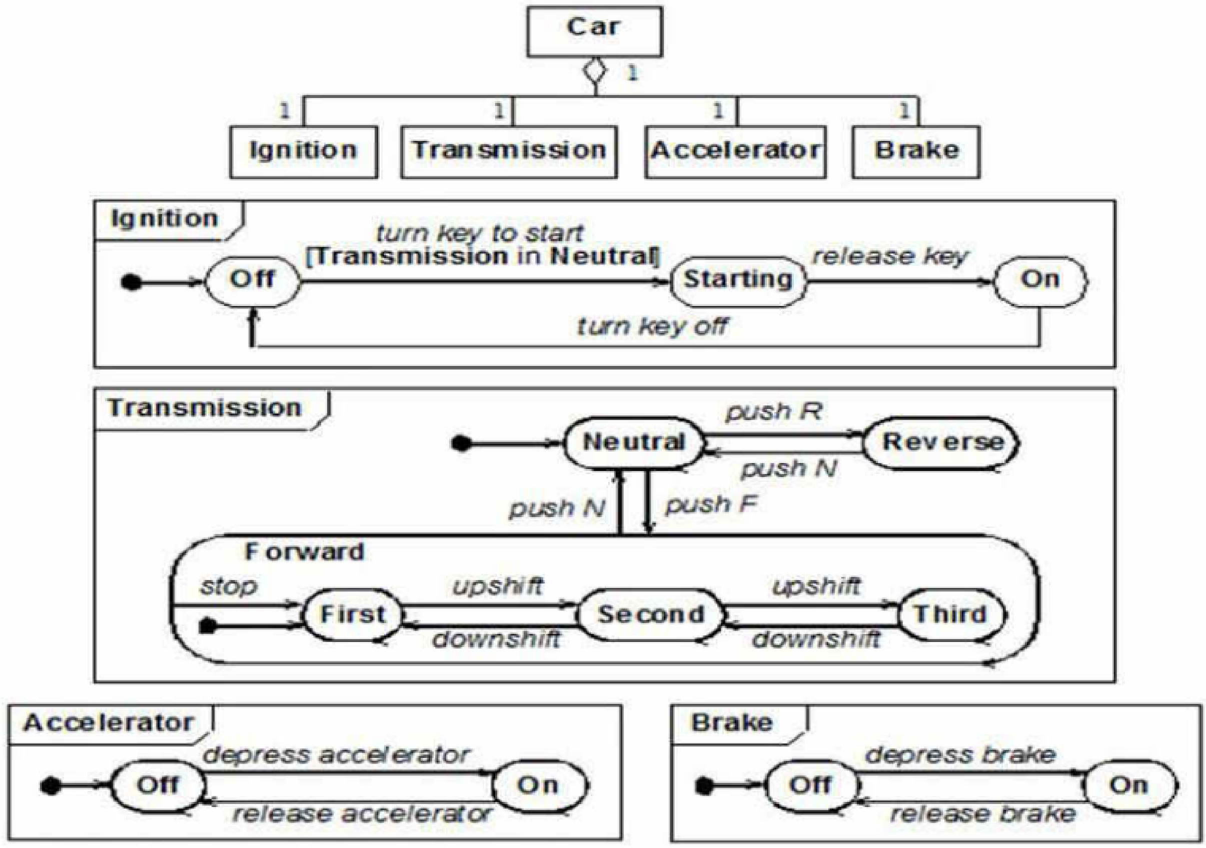


Learn from this state diagram of a ATM



Advanced state machines

This has nested states and concurrent states


Car
1
1
1
1
1
Ignition
Transmission
Accelerator
Brake
Ignition
turn key to start
[Transmission in Neutral]
Off
Starting
release key
On
turn key off
Transmission
Neutral
push R
Reverse
push N
push N
push F
Forward
stop
upshift
upshift
First
Second
Third
downshift
downshift
Accelerator
depress accelerator
Off
On
release accelerator
Brake
depress brake
Off
On
release brake