

Principles of Object Oriented

What is OOP?

- Is a programming paradigm
- Based on the concept of "objects"
- Object contain data and procedure

Why OOP?

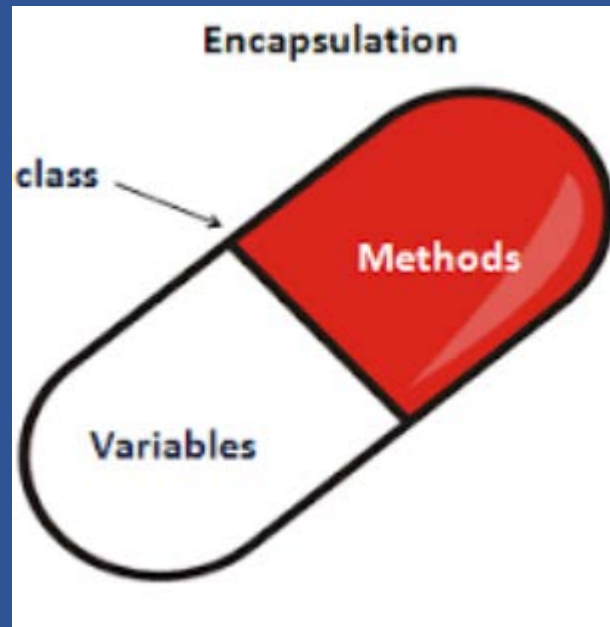
- Code reuseable
- Complexity hiding
- Easier to make large program
- Maintenance

OOP three pillars

- Encapsulation.
- Inheritance.
- Polymorphism.

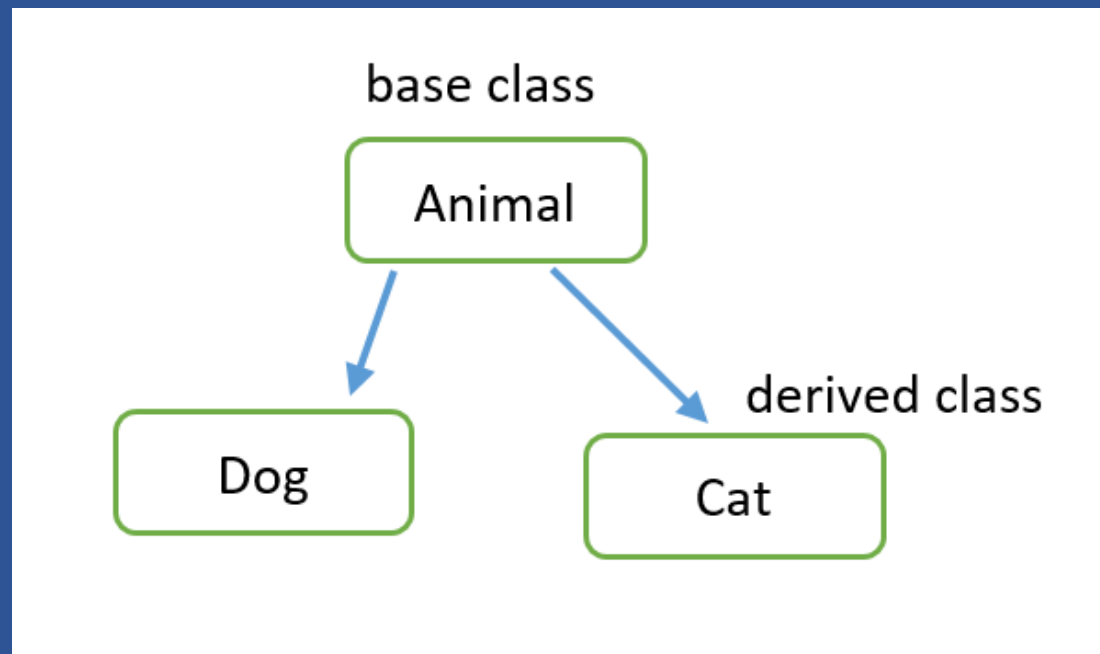
Encapsulation

- Black box
- Abstraction
- Focus on the essential features



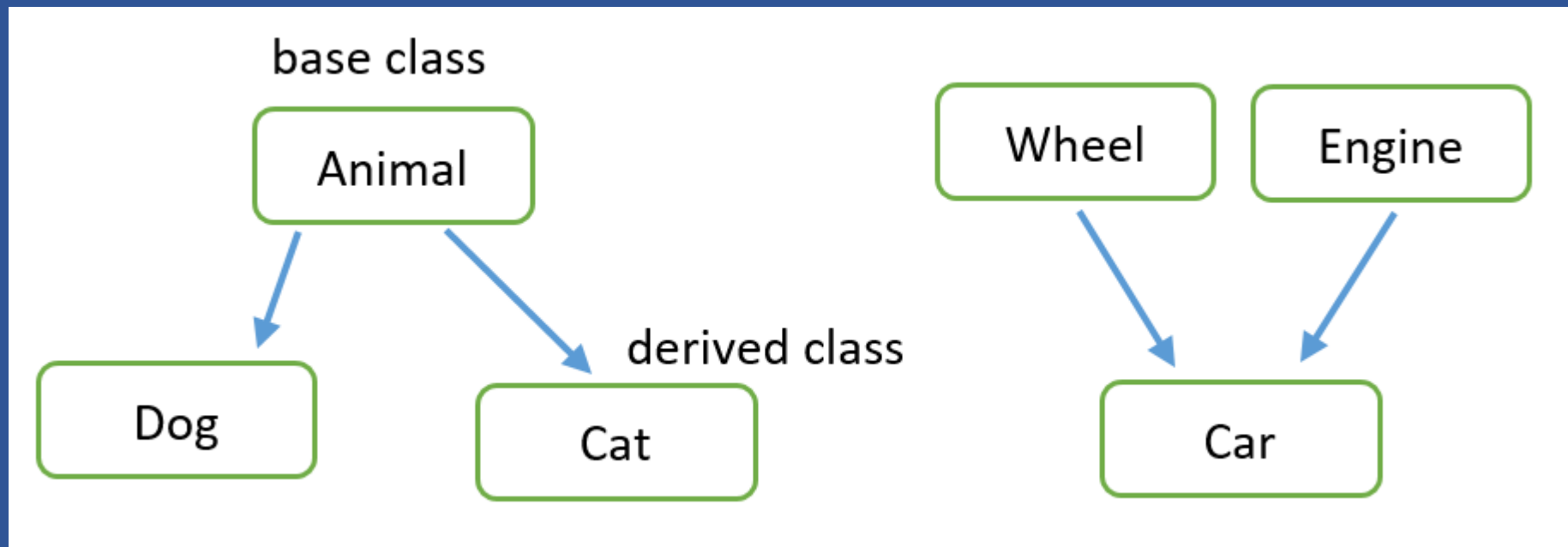
Inheritance

- Hierachy
- Ranking or ordering of abstraction



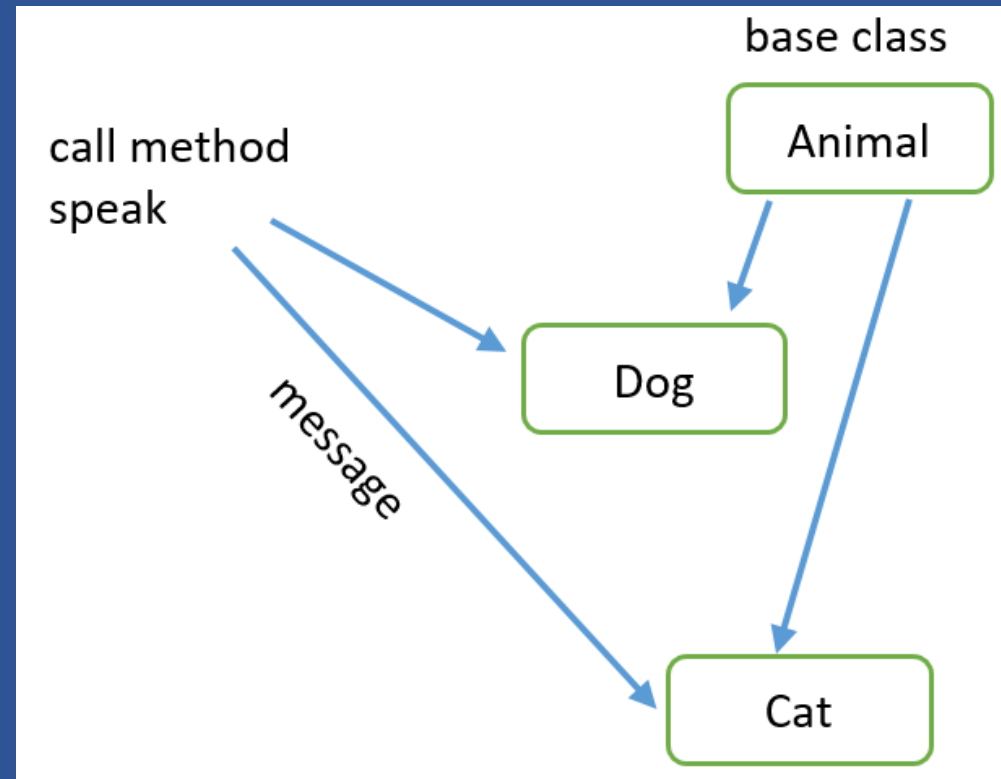
IS-A / PART-OF Relations

- “IS-A”: Animal = Cat “Is-a” Animal
- “PART-OF”: Engis is a “part of” Car



Polymorphism

The ability of different objects to respond to identical messages in its own way,.



Typing

- Class is type
- Strong typing: type check at compile time
- Weak typing: type check at run time

Concurrency

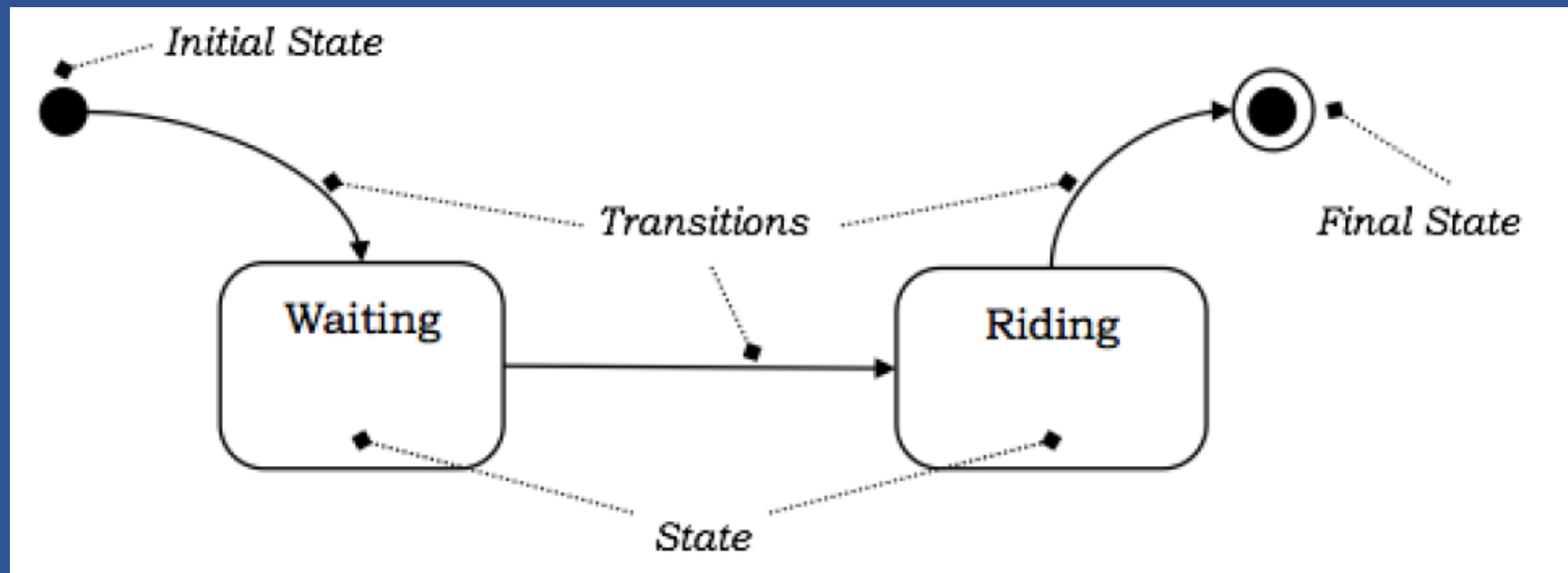
- Multi tasking
- Multi Threads
- Parallel processing
- Asynchronous programming
- GUI Thread
- Cross thread operation

Persistence

- How long an object occupies memory?
- Garbage collection
- Manage heap efficiently
- Destroy object
- Memory safety, object cannot use the content of another object

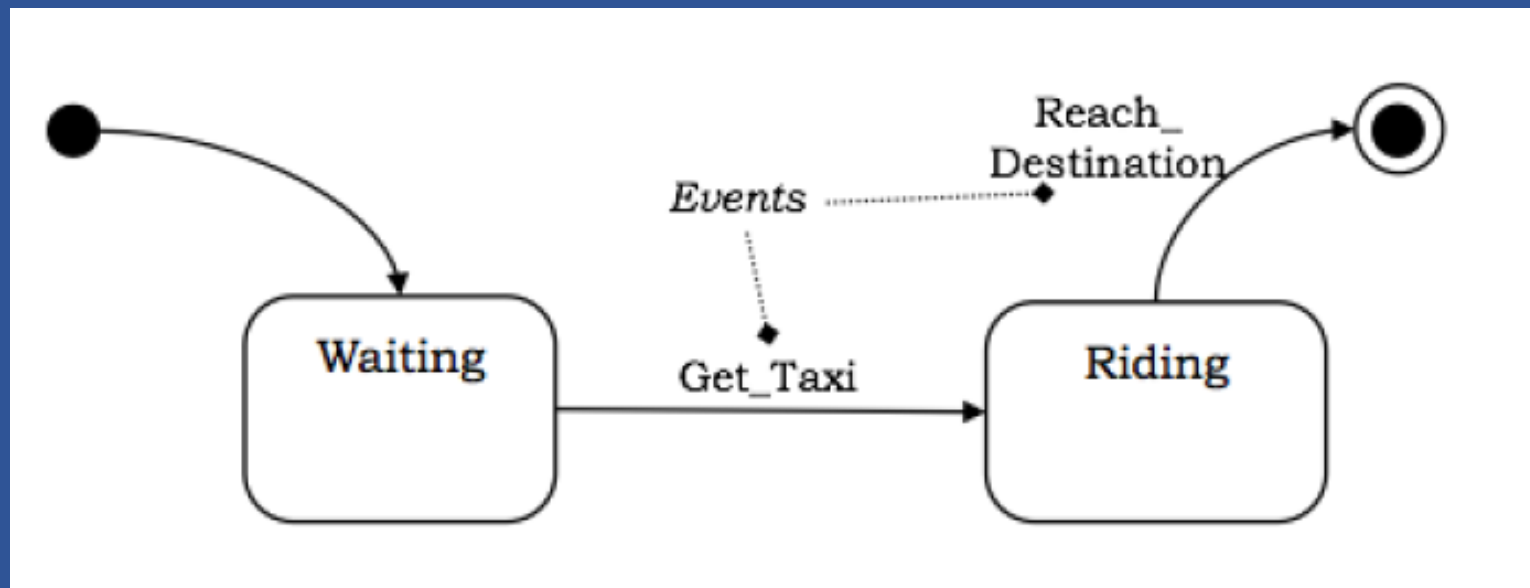
Transition

- A person is taking a taxi from place X to place Y
- Waiting
- Riding
- Reached



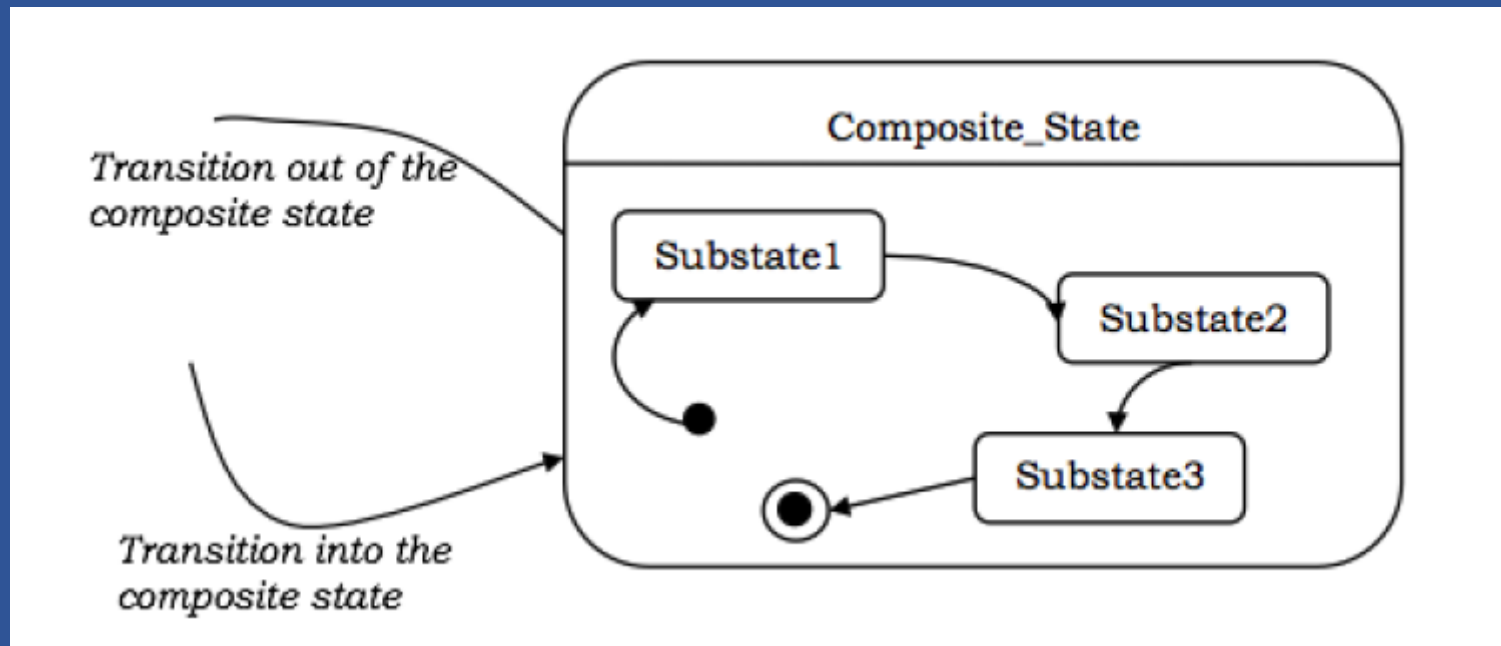
Event

- Object can send event
- Object can receive event



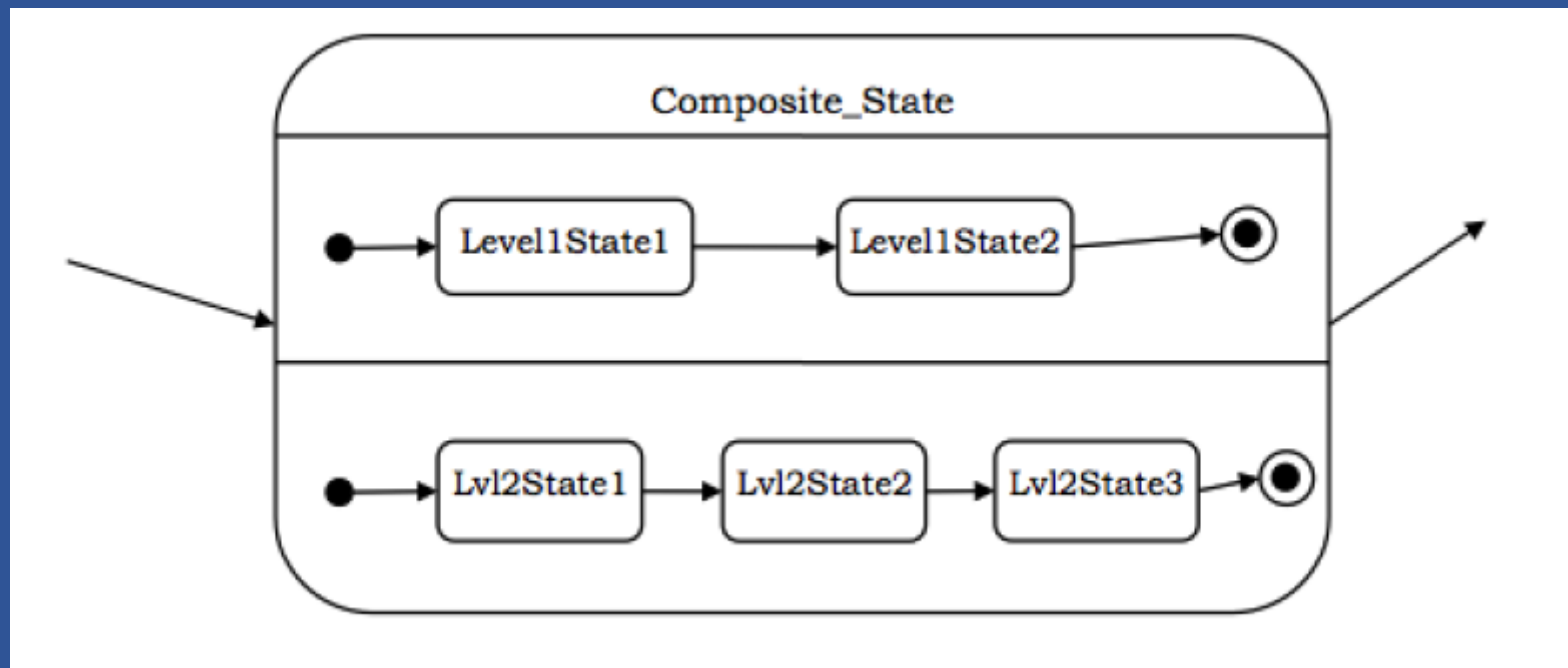
Interaction Diagrams

Describe the dynamic behavior among different objects. It comprises of a set of objects, their relationships, and the message that the objects send and receive

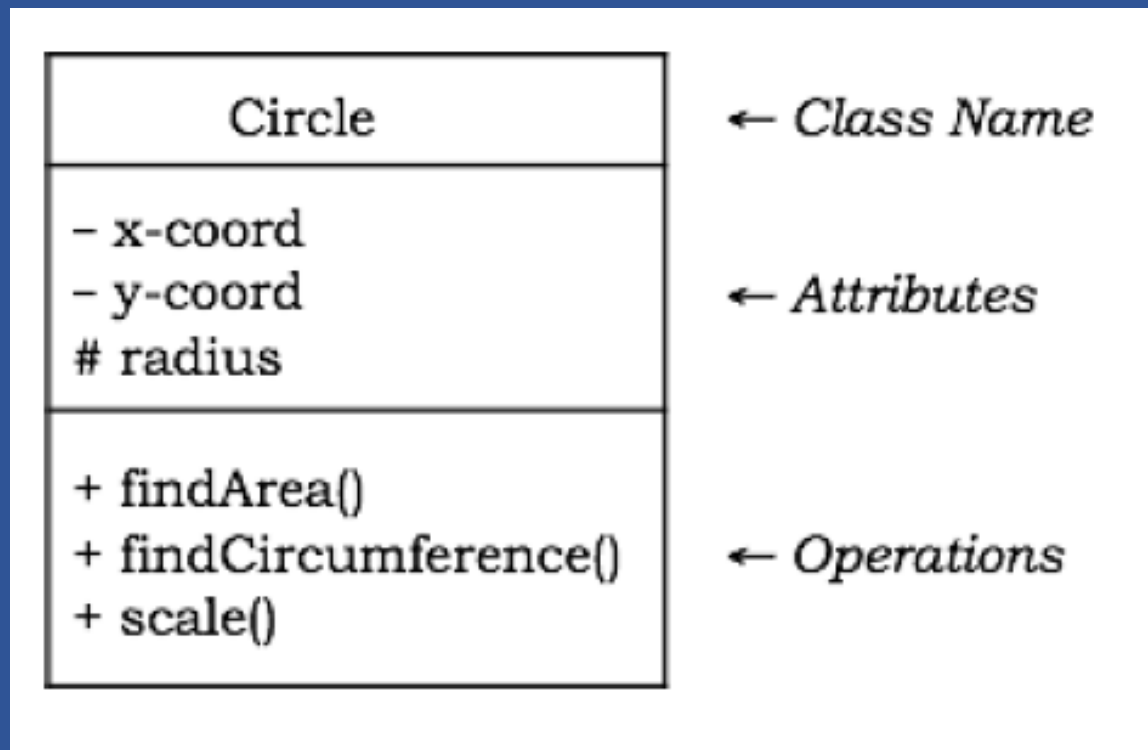


Concurrent Sub-States

- Each state has concurrently executing state
- Each has its own initial and final states

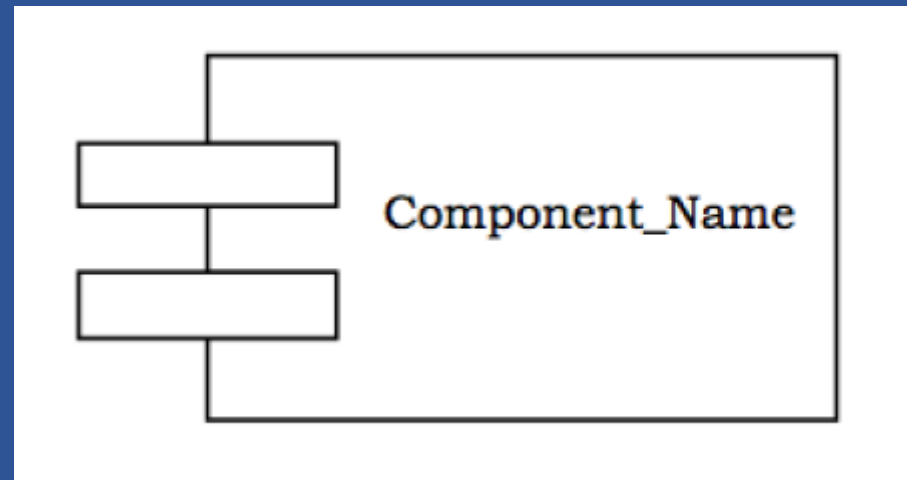


Class diagram



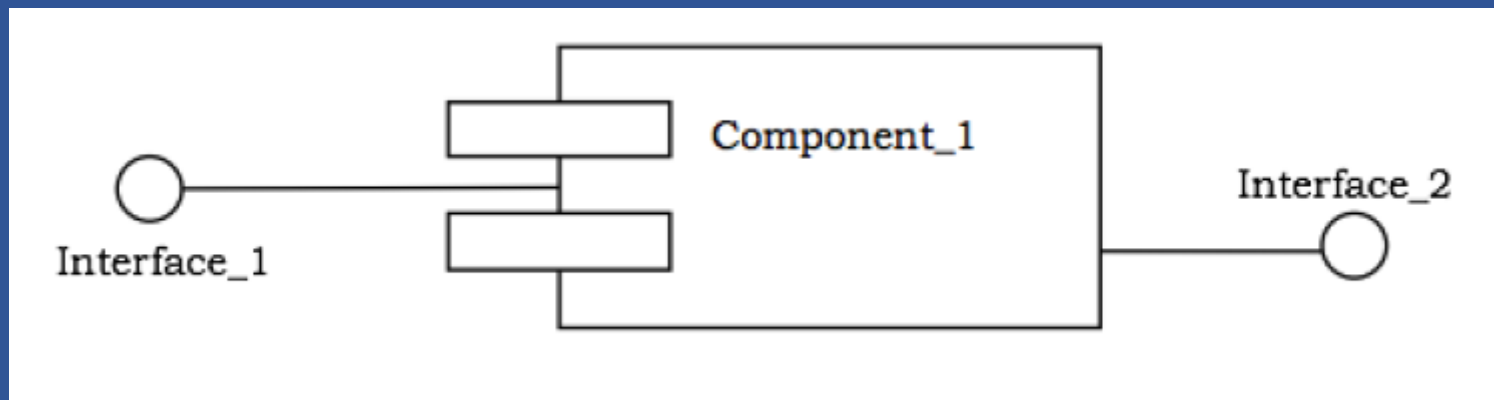
Component

- Is a physical and replaceable part of the system
- Provides the realization of a set of interfaces.
- It represents classes and interfaces

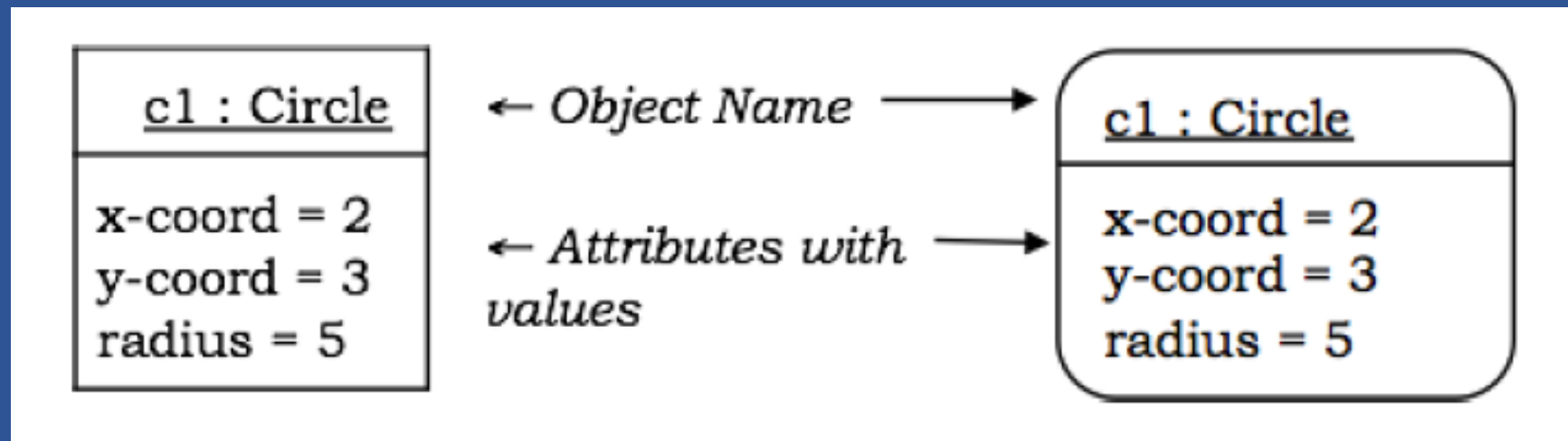


Interface







- Interface is a collection of methods of a class or component.
- It specifies the set of services that may be provided by the class or component.



Object

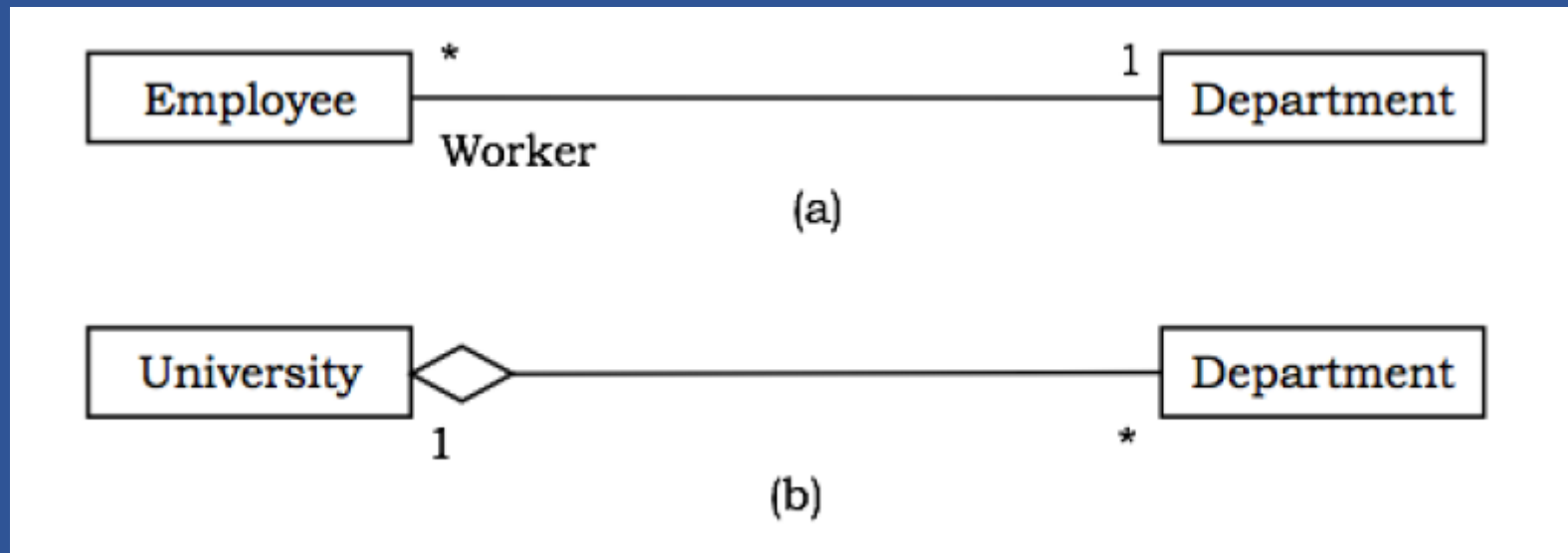


Relationship

Dependency	
Association	
Direct Association	
Inheritance	
Realization	
Aggregation	

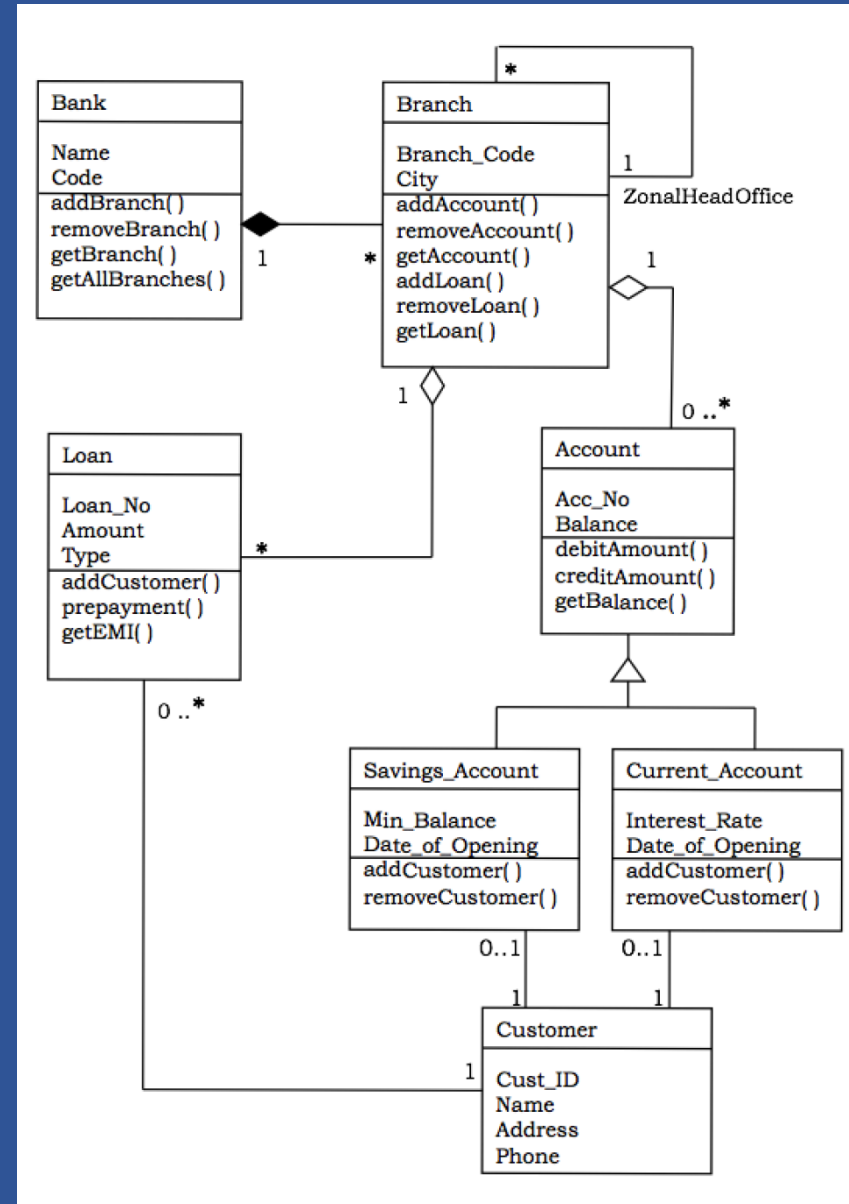
Example

- One department “has many Employee”
- A University is the “whole-of” many Departments



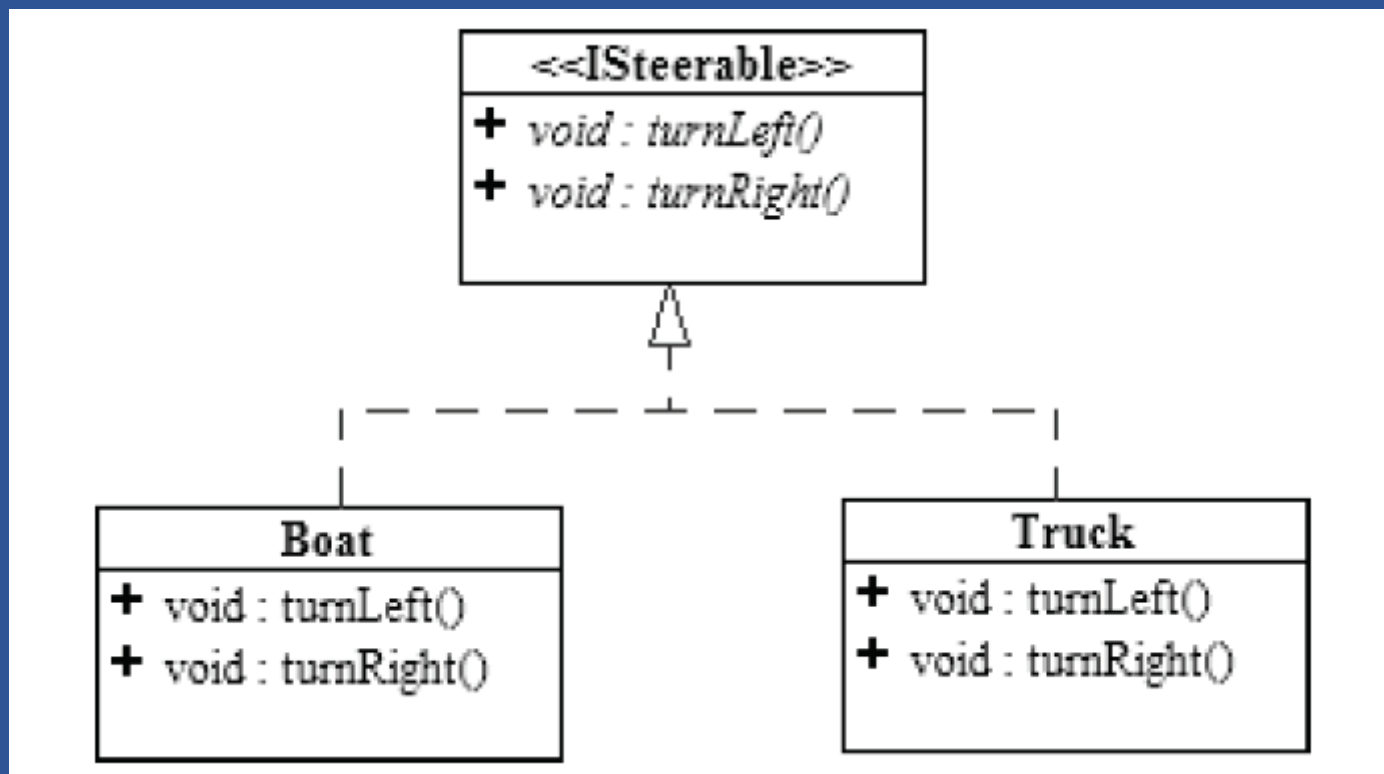
Example

- A Customer can have one Current Account :
 - association, one-to-one
- A Customer can have one Savings Account :
 - association, one-to-one
- A Branch “has-a” number of Loans :
 - aggregation, one-to-many
- A Customer can take many loans :
 - association, one-to-many



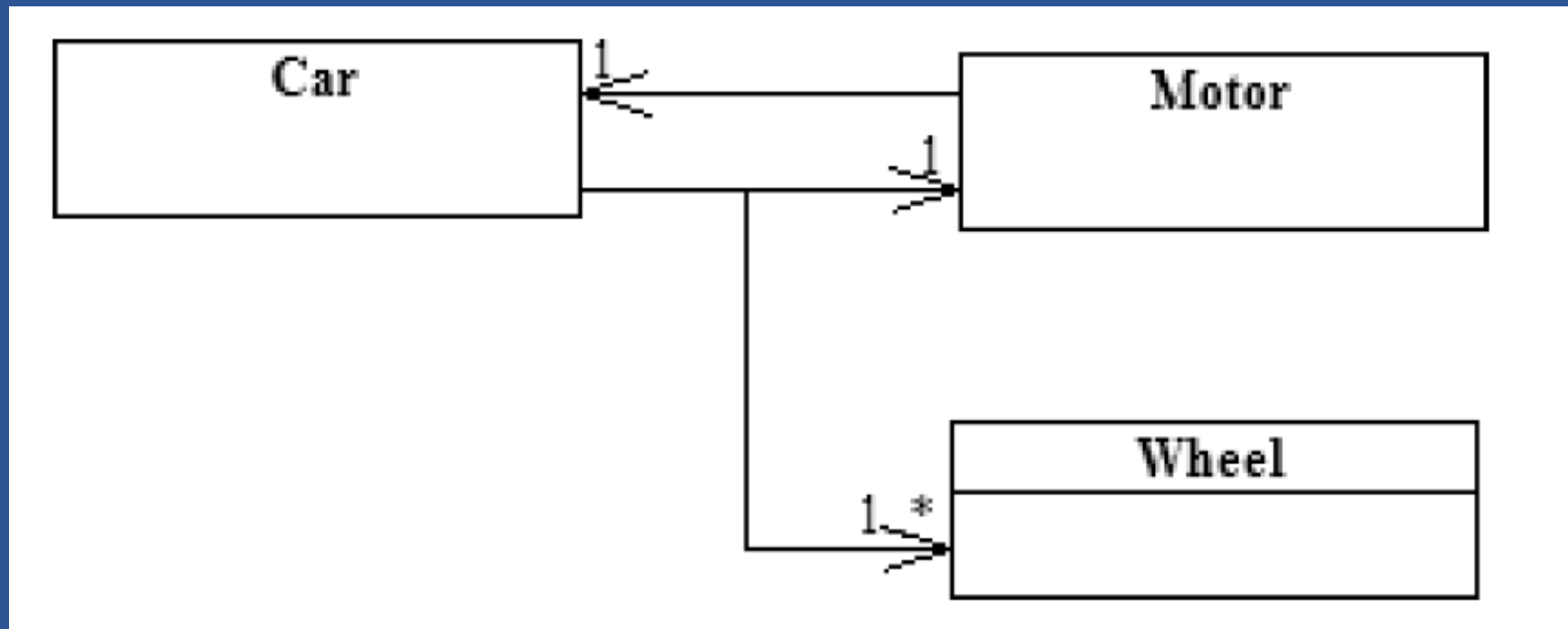
IS-A diagram

Boat and Truck “is a” ISteerable



PART-OF Diagram

Motor and Wheel are “part of” car



Polymorphism diagram

Each object response to same the message differently

