Introduction to Object Oriented Concepts

Object Oriented Paradigm

(Paradigm: a way of seeing and doing things)

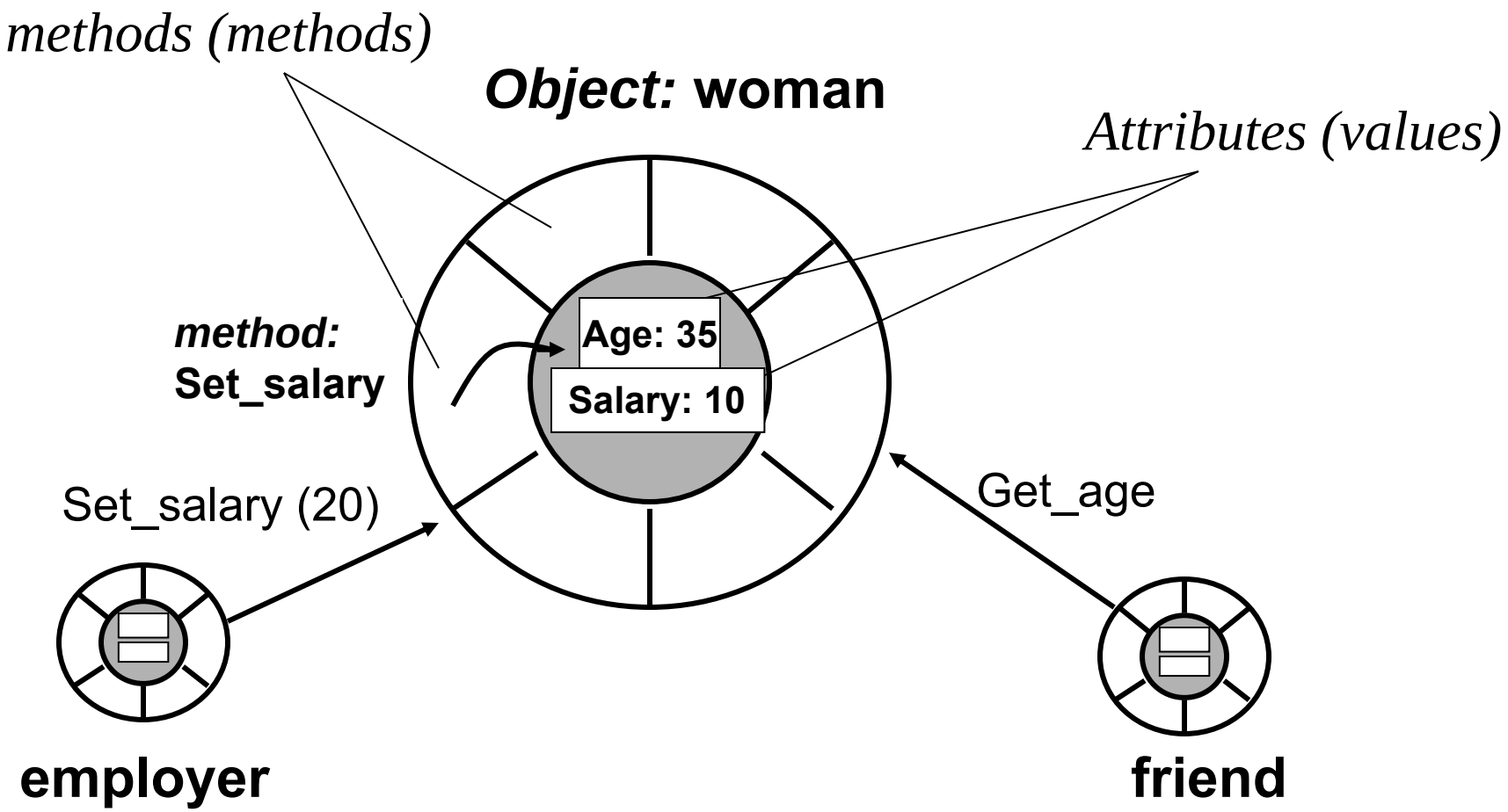
- Object - Oriented (OO) Programming:
  - Organizing software as a collection of objects with a certain state and behavior.
- Object Oriented Design:
  - Based on the identification & organization of objects.
- OO Methodology
  - Construction of models
  - The development of SW is a modeling process
- OO Modeling and Design
  - Modeling objects based on the real world
  - Using models to design independently of a programming language

.

Objects

- Object: Complex data type that has an **identity**, contains other data types called **attributes** and modules of code called **operations** or **methods**
- Attributes and associated values are **hidden** inside the object.
- Any object that wants to obtain or change a value associated with other object, must do so by sending a **message** to one of the objects (invoking a method)

Objects


methods (methods)
Object: woman
Attributes (values)
method:
Set_salary
Age: 35
Salary: 10
Set_salary (20)
Get_age
employer
friend

Encapsulation

- Each objects methods manage it's own attributes.
- This is also known as ***hiding.***
- An object A can learn about the values of attributes of another object B, only by invoking the corresponding method (message) associated to the object B.

- Example:
  - Class: Lady
  - Attributes: Age, salary
  - Methods: get_age, set_salary

Procedural vs. Object-Oriented

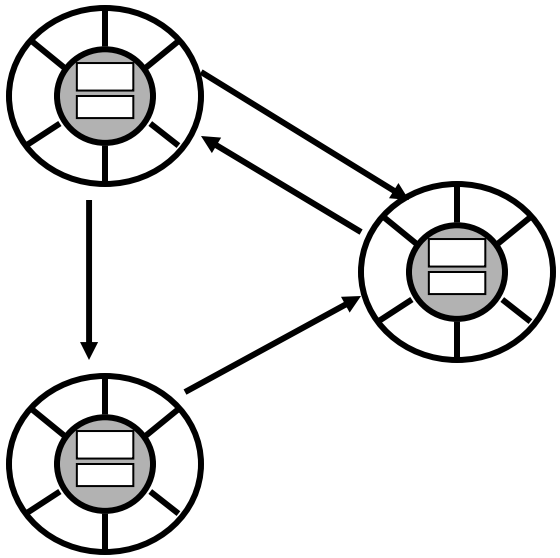
Procedural application

DATA

Line of code
Line of code
Line of code

Data is stored independent of application

OO-application

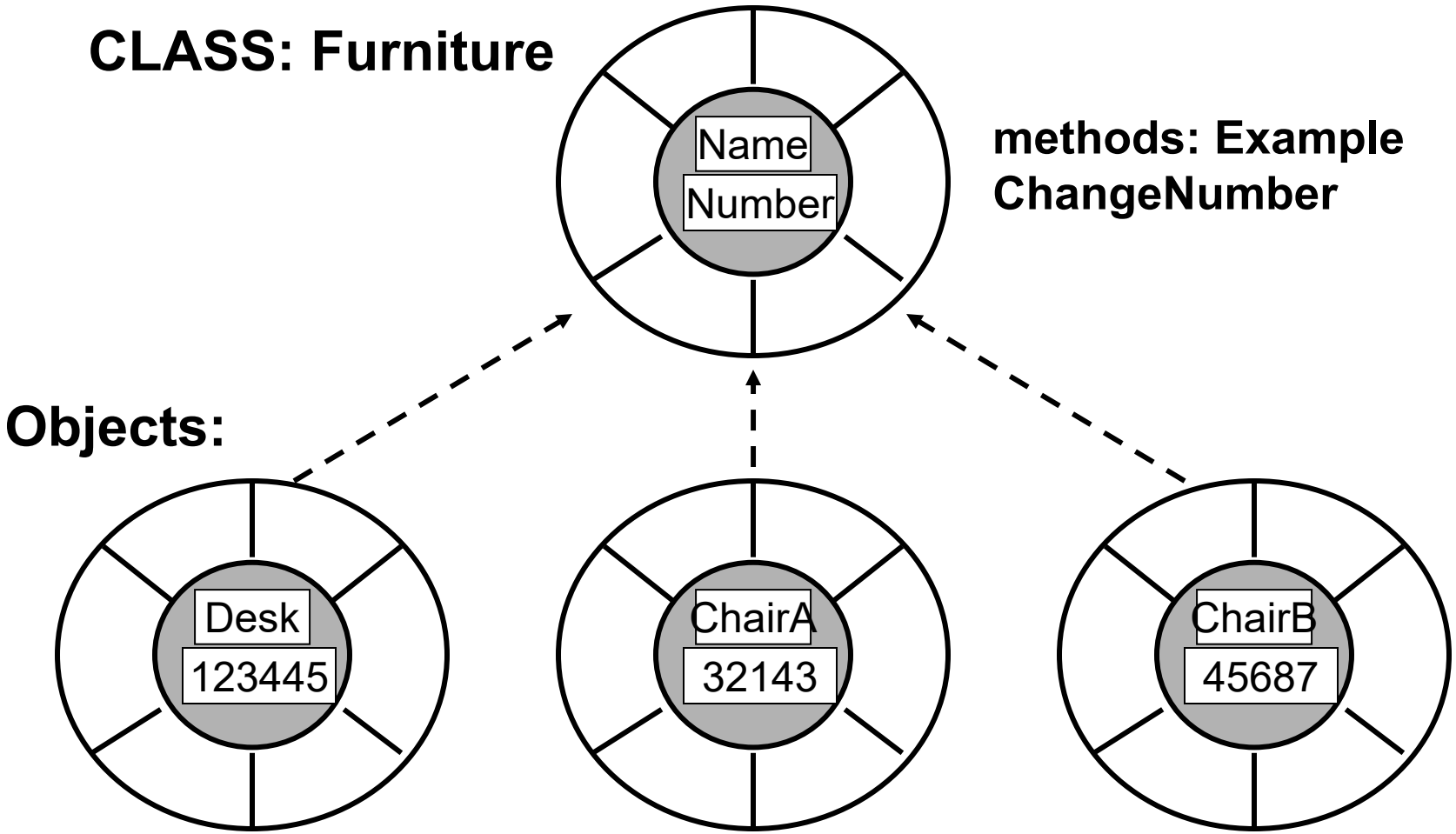


Each object is independent of the others

Classes

- Classes are **templates** that have methods and attribute names and type information, but no actual values!
- Objects are generated by these classes and they actually contain values.
- We design an application at the class level.
- When the system is running objects are created by classes as they are needed to contain state information.
- When objects are no longer needed by the application, they are eliminated.

Class & Objects

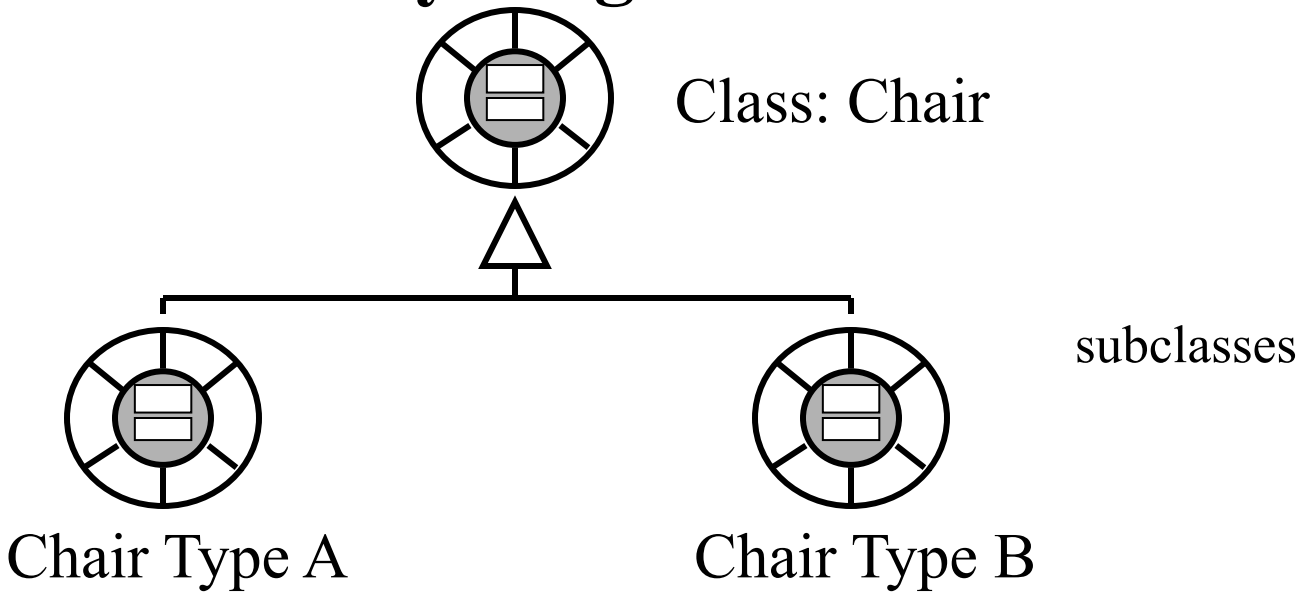


Message Passing & Associations

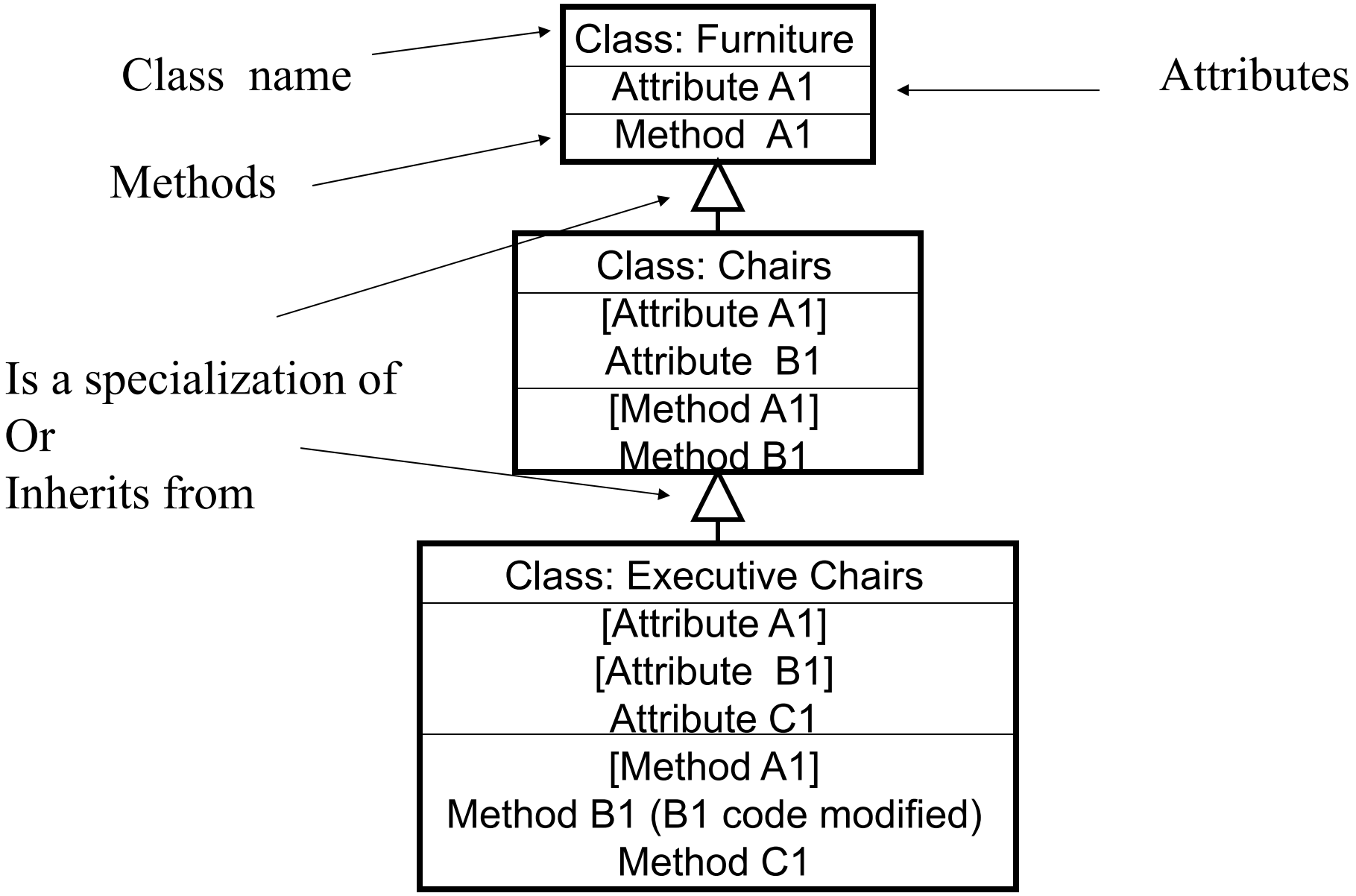
- Methods are associated with classes but classes don't send messages to each other.
- Objects send messages.
- A **static diagram (class diagram)** shows classes and the logical associations between classes, it doesn´t show the movement of messages.
- An **association** between two classes means that the objects of the two classes can send messages to each other.
- **Aggregation**: when an object contains other objects (a part-whole relationship)

Class Hierarchies & Inheritance

- Classes can be arranged in hierarchies so that more classes inherit attributes and methods from more abstract classes
- **Class hierarchy diagrams**



Class Inheritance & Specialization



Public, Private & Protected

- Attributes can be public or private:
  - **Private**: it can only be accessed by its own methods
  - **Public**: it can be modified by methods associated with any class (violates encapsulation)
- Methods can be public, private or protected:
  - **Public**: its name is exposed to other objects.
  - **Private**: it can't be accessed by other objects, only internally
  - **Protected**: (special case) only subclasses that descend directly from a class that contains it, know and can use this method.

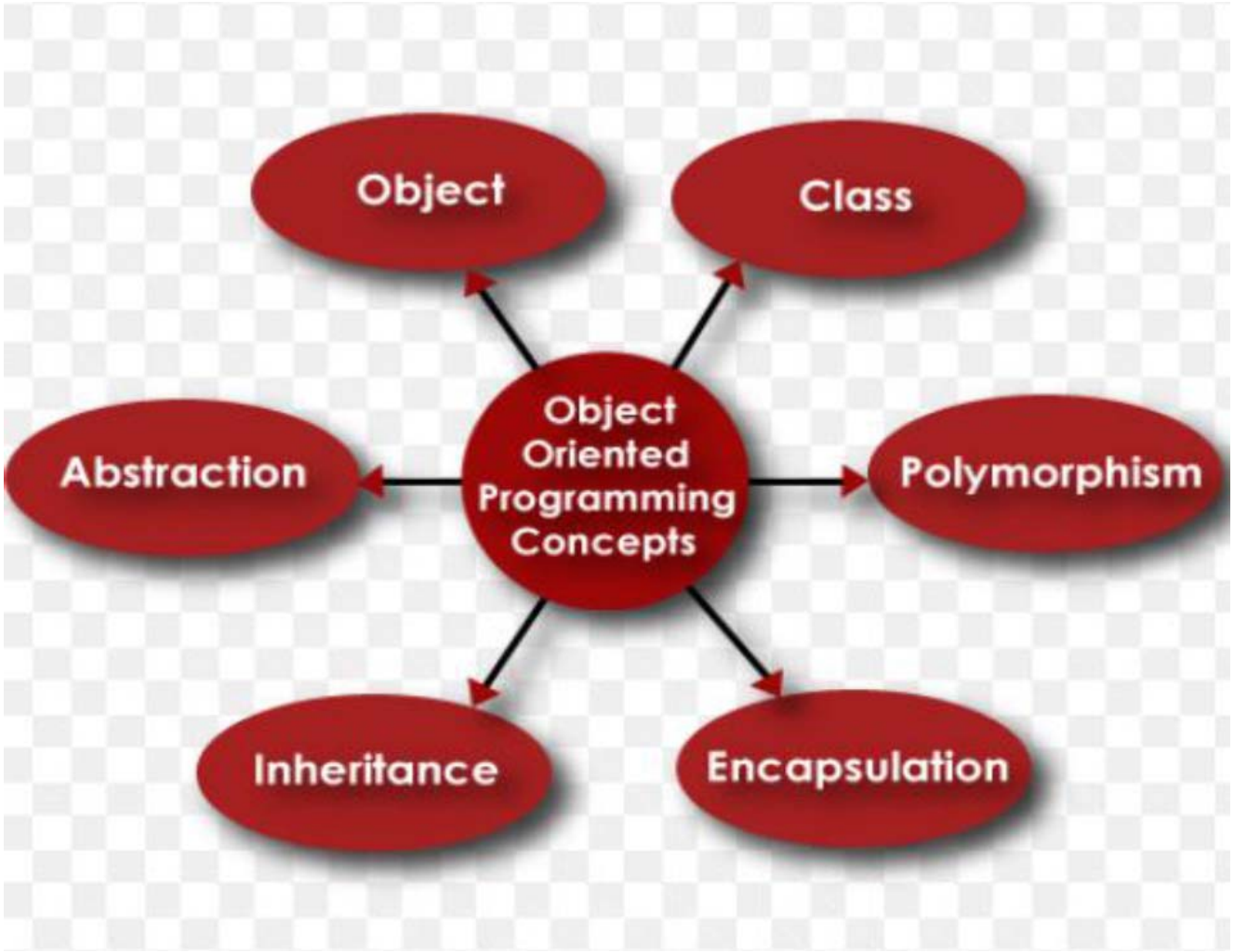
Method signature

- It is the method's name and the parameters that must be passed with the message in order for the method to function.
- The parameters are important because they assure that the method will function properly.
- Additionally they allow a compiler or interpreter to discriminate between two different methods with the same name.

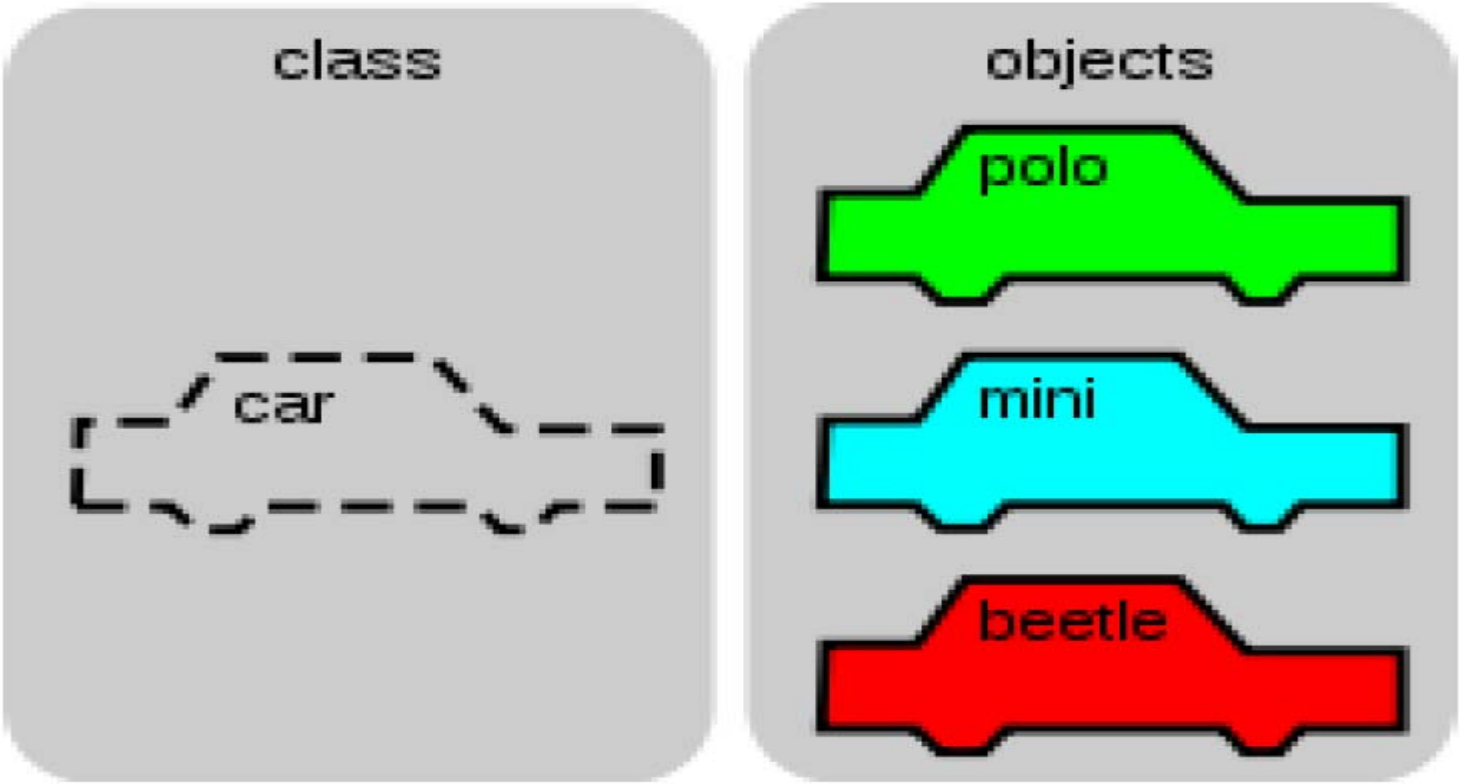
Polymorphism

- Means that the same method will behave differently when it is applied to the objects of different classes
- It also means that different methods associated with different classes can interpret the same message in different ways.
- Example: an object can send a message PRINT to several objects, and each one will use its own PRINT method to execute the message.

REVIEW


Object
Class
Object Oriented Programming Concepts
Abstraction
Polymorphism
Inheritance
Encapsulation

Object Basics



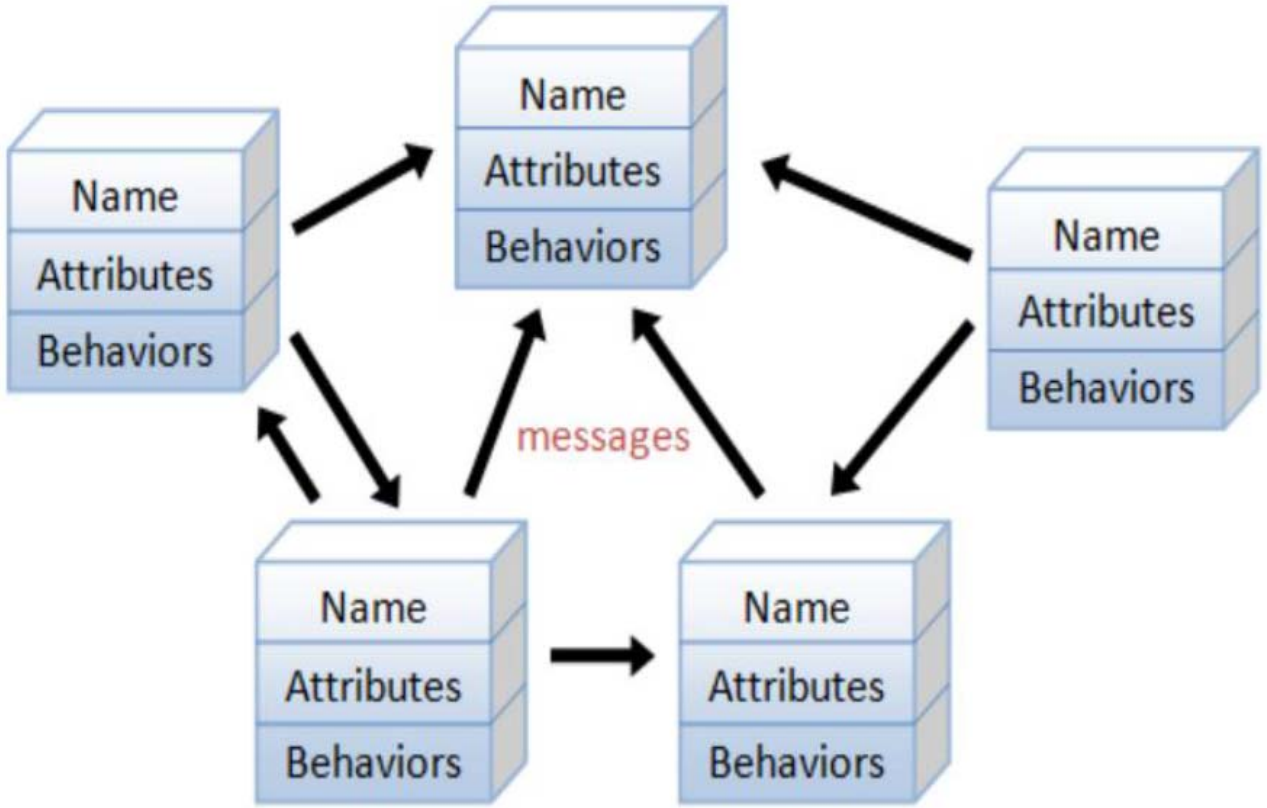
Object Basics

Properties
Make
Model
Color
Year
Price

Events
On_Start
On_Parked
On_Brake

Methods
Start
Drive
Park

Object Basics



Object oriented program consists of many well-encapsulated Objects and interacting with each other by sending messages

An Object

- In object-oriented programming(OOP), objects are the units of code that are eventually derived from the process.
- Each object is an instance of a particular class.
- They have:
  - Properties or attributes
    - Describe the state of an object
  - Methods or procedures
    - Define its behavior

An Object-Oriented Philosophy

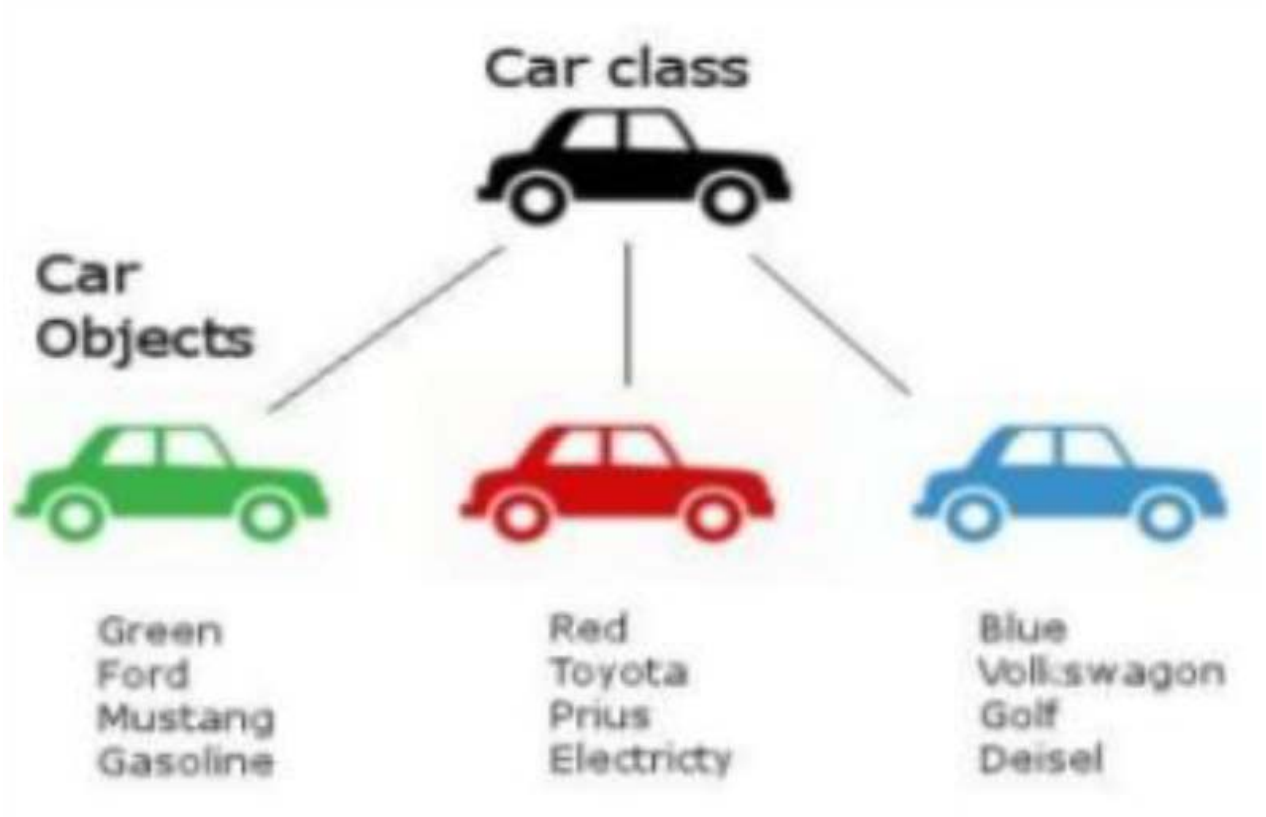
- Traditional development methodology
  - Algorithm-centric methodology
    - Think algorithm, then build data structures
  - Data-centric methodology
    - Think how to structure data, then build algorithm
- Object-oriented programming
  - Allows the basic concepts of the language to be extended to include ideas and terms closer to those of its applications.
  - The algorithm and data are packaged together as an object, which has a set of attributes or properties.

An Object-Oriented Philosophy

- The term Object means a combination of data and logic that represents some real world entity.
- In an object-oriented system, everything is an object.

Classes

- Objects are grouped in Classes



Classes

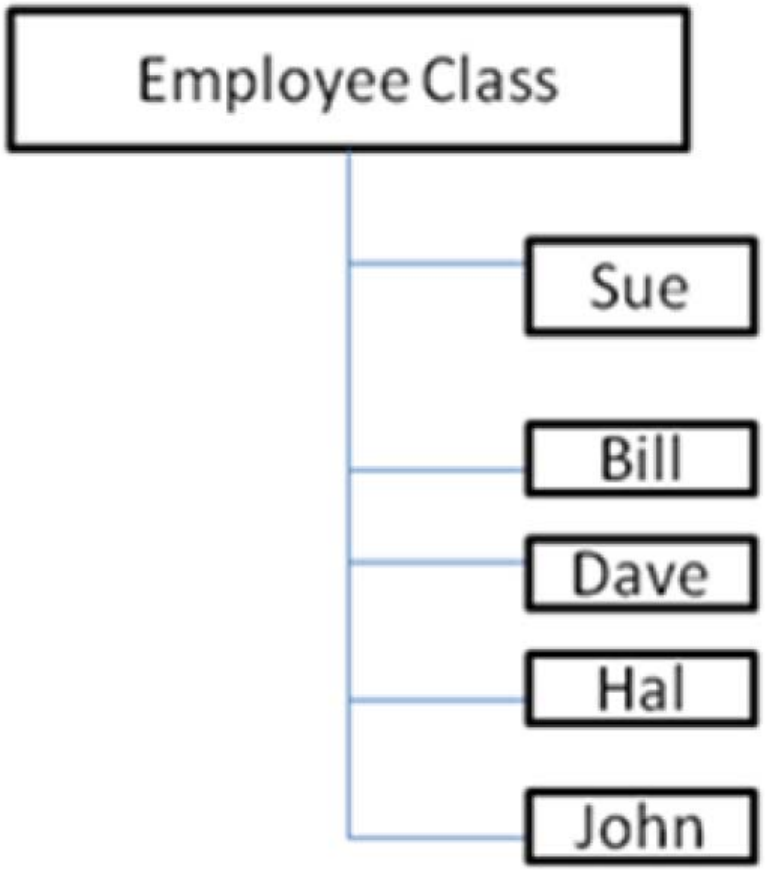
- Classes are used to distinguish one type of object from another.
- A class is a set of objects that share a common structure and behavior.
- A single object is simply an instance of a class.

Classes

- A class is a specification of structure (instance variables), behavior(methods) and inheritance for objects.
- Classes are an important mechanism for classifying objects.
- A method or behavior of an object is defined by its class.
- Each object is an instance of a class.

Classes

- Example: Objects of the class Employee



Attributes

- Attributes: Object state and Properties.
- Properties represent the state of an object.
- Example: the attributes of a car object.

Methods

- In the object model, object behavior is described in methods or procedures.
- A method implements the behavior of an object.
- A method is a function or procedure that is defined for a class and typically can access the internal state of an object of that class to perform some operation.

Methods

- Behavior denotes the collection of methods that abstractly describes what an object is capable of doing.
- The object is that on which the method operates.
- Methods encapsulate the behavior of the object, provide interfaces to the object, and hide any of the internal structures and states maintained by the object.

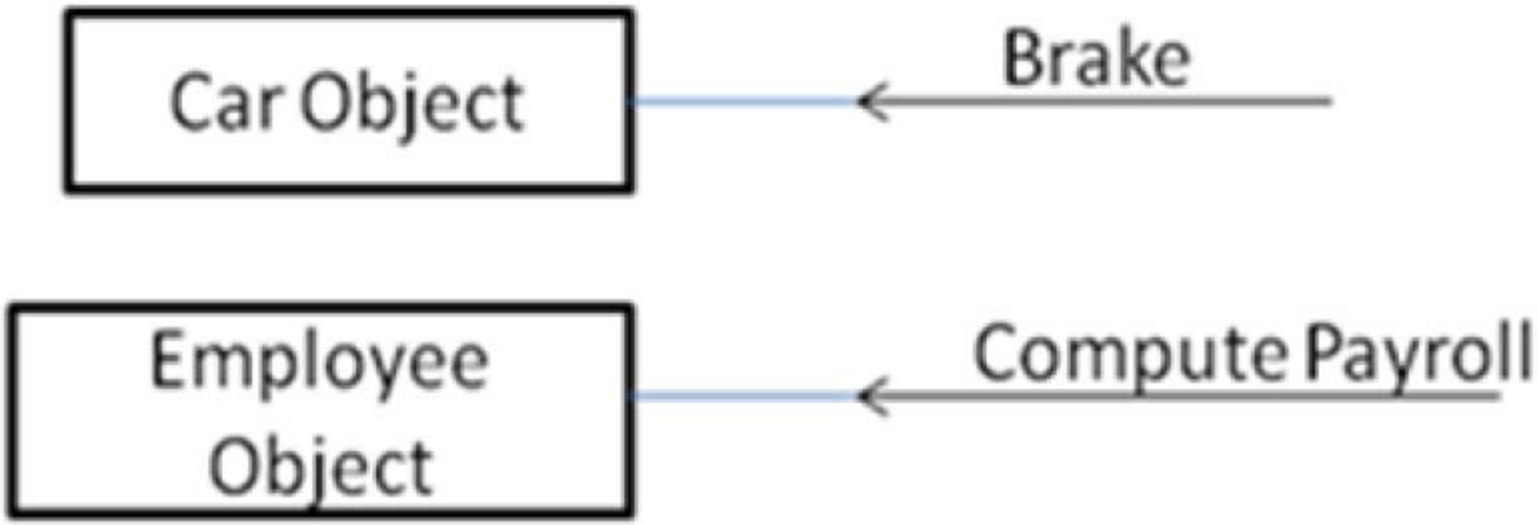
Methods and Messages

- Objects Respond to Messages.
- An object's capabilities are determined by the methods defined for it.
- Objects perform operations in response to messages.
- Example: stop method -> car object

Methods and Messages

- Messages especially are function calls.
- Different objects can respond to the same message in different ways (polymorphism).
- Message is the instruction (function call) and method is the implementation.
- An object understands a message when it can match the message to a method that has the same name as the message.

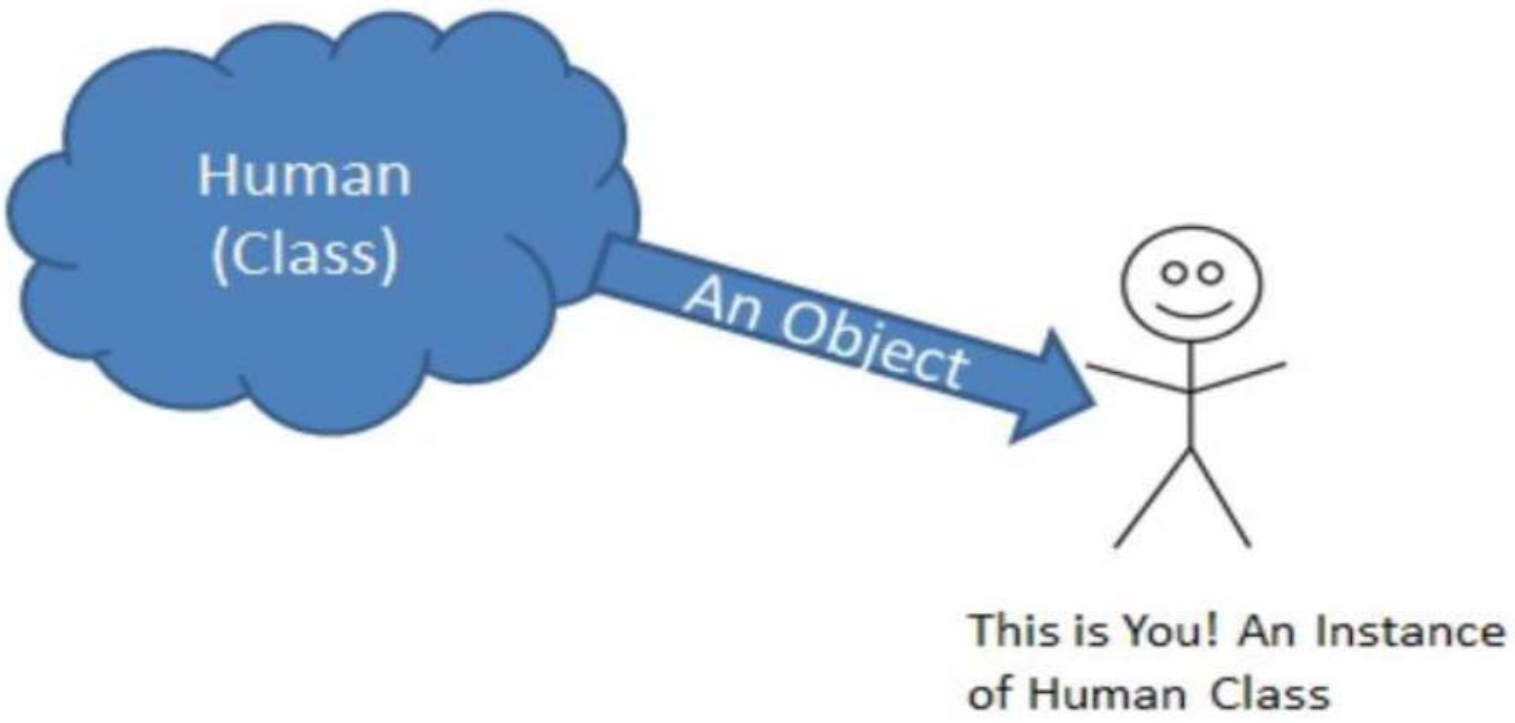
Methods and Messages



Methods and Messages

- The object first searches the methods defined by its class.
- If not found, it searches the super class of its class.
- An error occurs if none of the super classes contains the method.

Class and Object

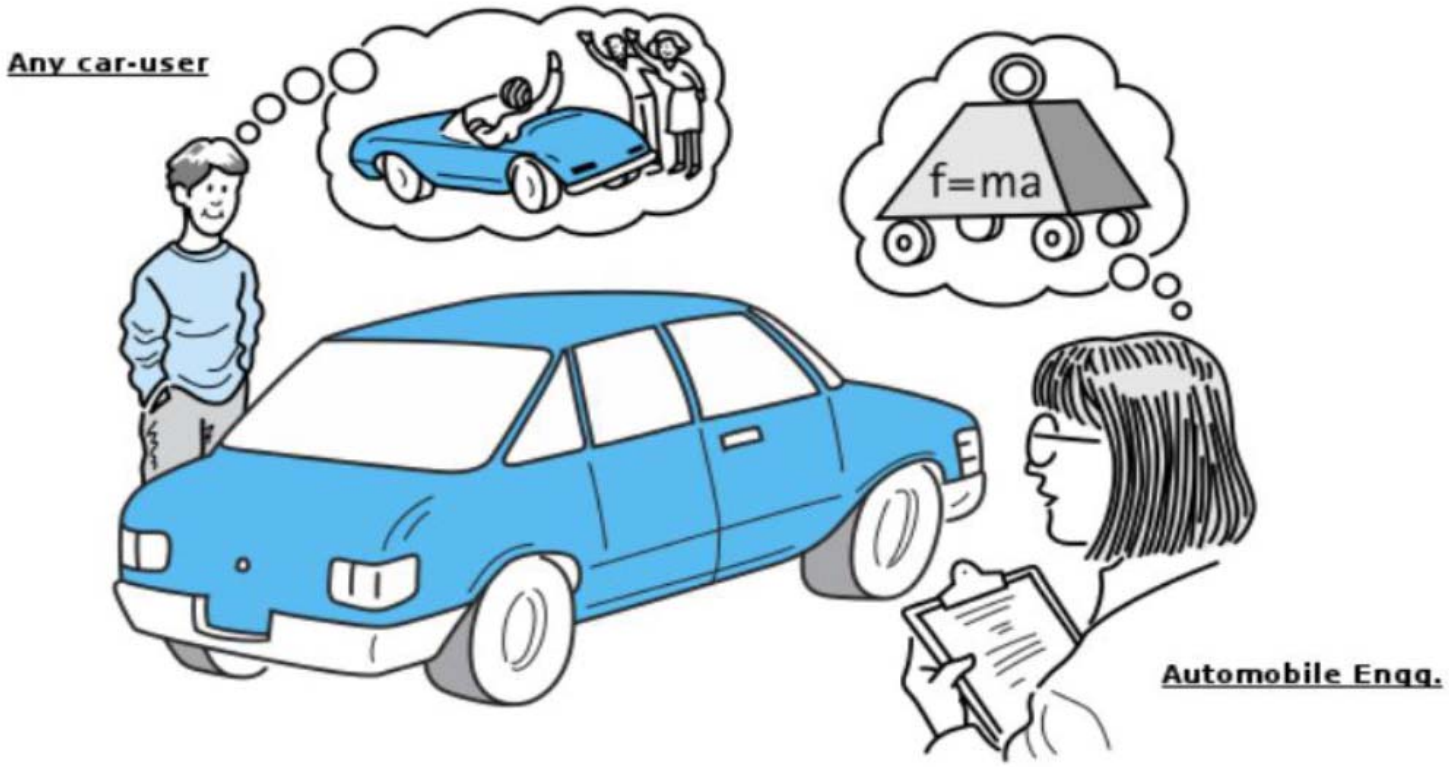


Human Class and Human Object

Abstraction

- Abstraction means displaying only essential information and hiding the details.
- Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.
- Consider a real life example of a man driving a car.

Abstraction



Abstraction includes the essential details relative to the perspective of the viewer

Encapsulation and Information Hiding

- Principle of concealing the internal data and procedures of an object and providing an interface to each object in such a way as to reveal as little as possible about its inner workings.

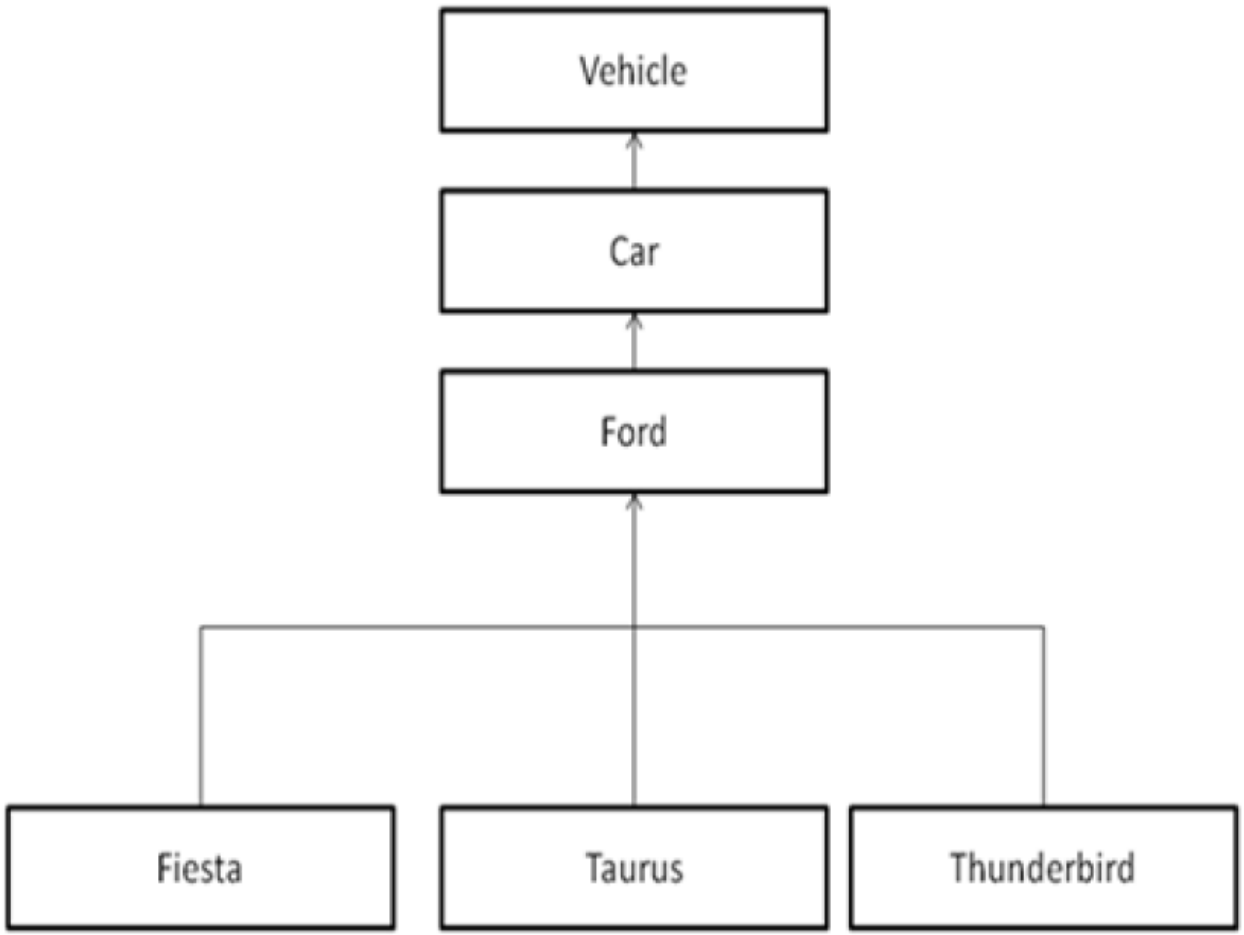
Encapsulation and Information Hiding

- Public members may be accessed from anywhere.
- Private members are accessible only from within a class.
- Protected members can be accessed only from subclasses.

Inheritance

- An object-oriented system organizes classes into subclass-superclass hierarchy.
- At the top of the class hierarchy are the most general classes and at the bottom are the most specific.
- A subclass inherits all of the properties and methods defined in its superclass.
- A class may simultaneously be the subclass to some class and a superclass to another class or classes.

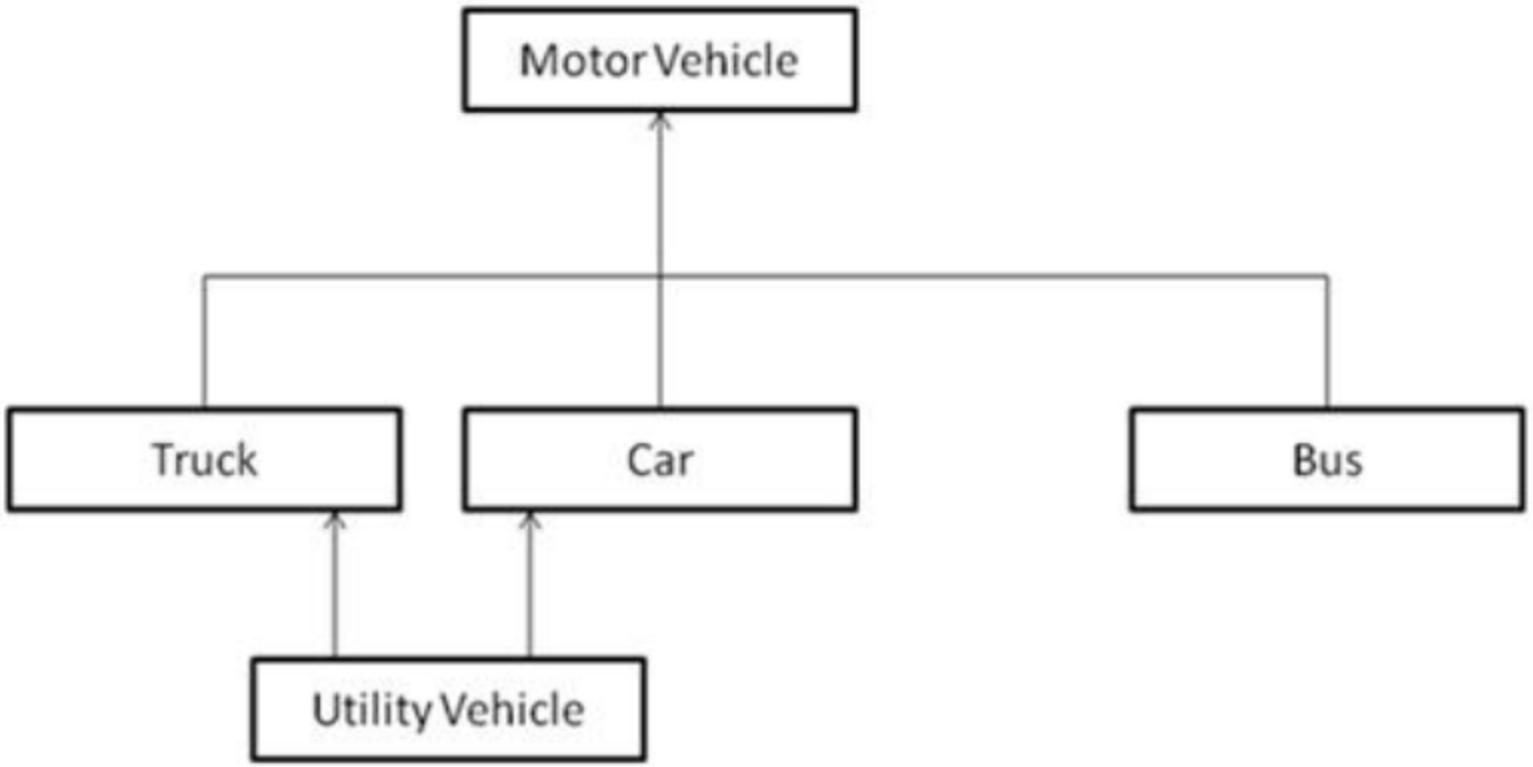
Inheritance



Multiple Inheritance

- Some object-oriented systems permit a class to inherit its state and behaviors from more than one superclass.
- This kind of inheritance is referred to as multiple inheritance.

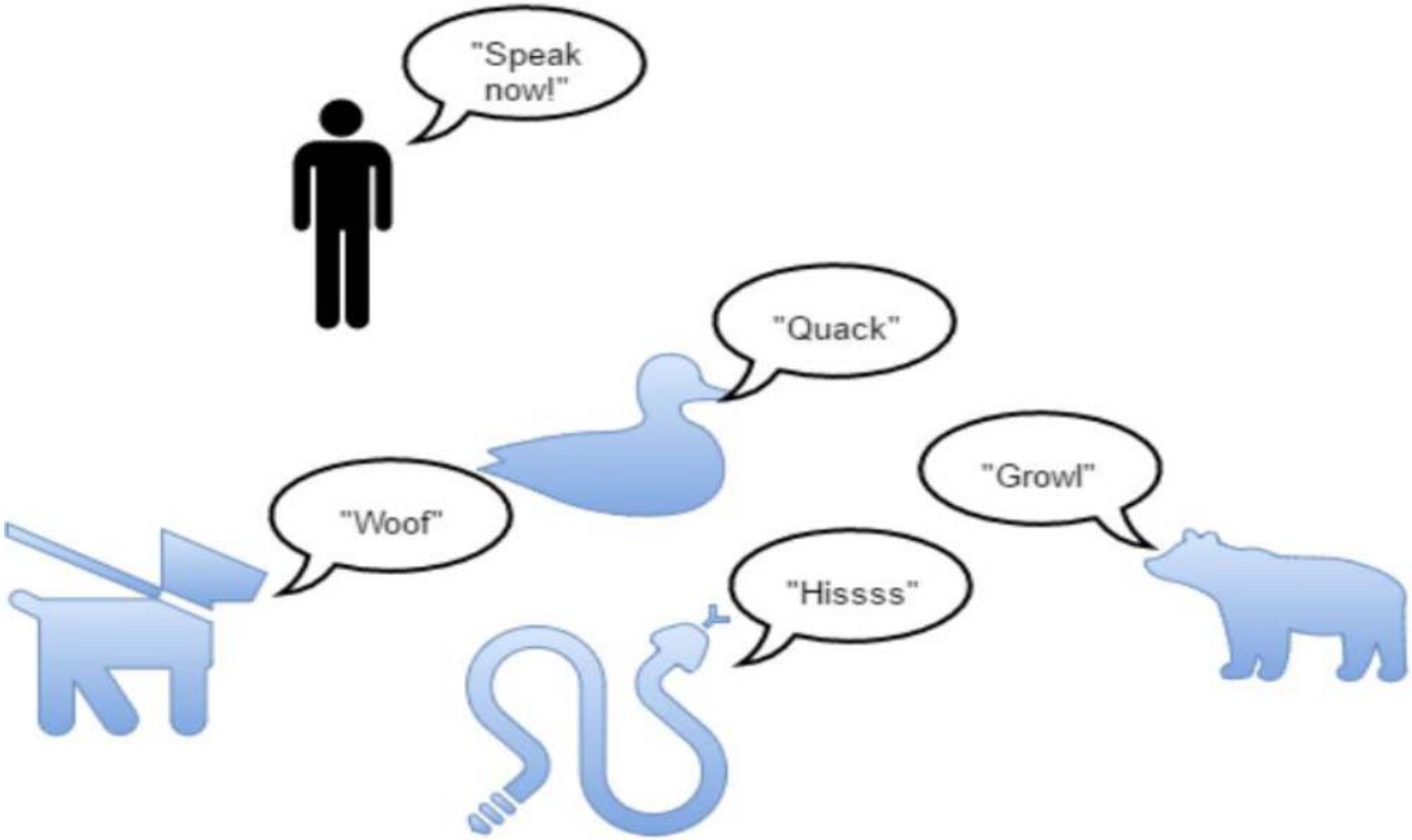
Multiple Inheritance



Polymorphism

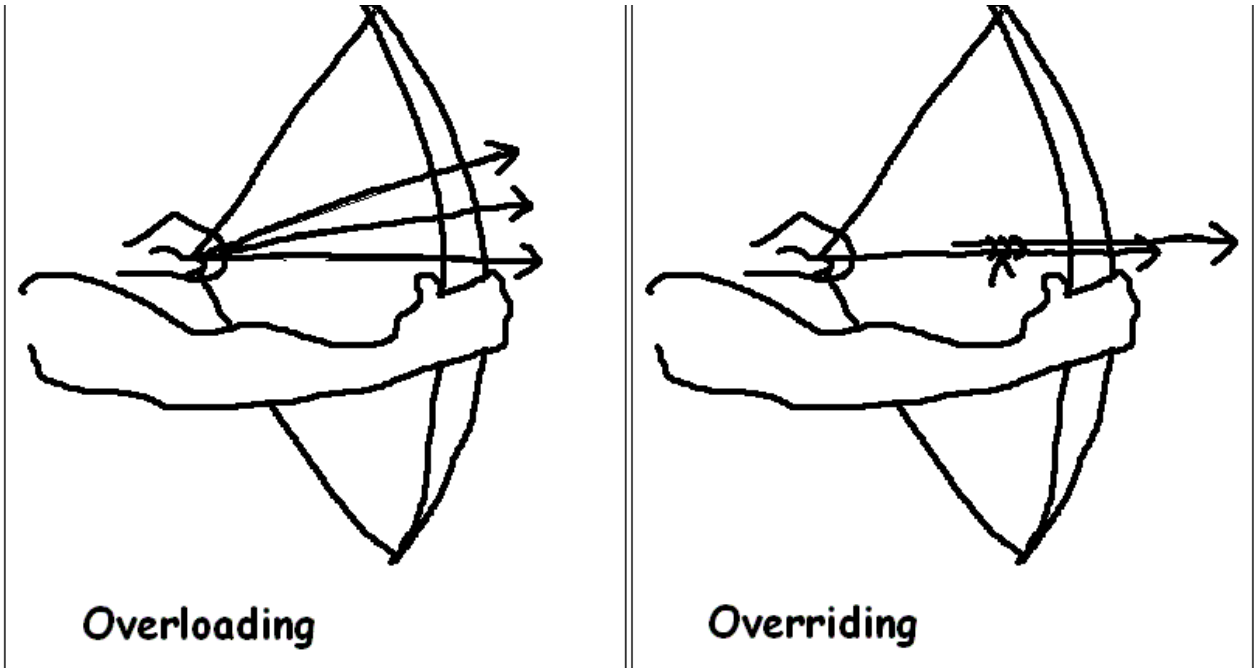
- It means objects that can take on or assume many different forms.
- The same operation may behave differently on different classes.
- Allows us to write generic, reusable code more easily.

Polymorphism



Polymorphism: Overloading and Overriding

- Different methods with same name !!!

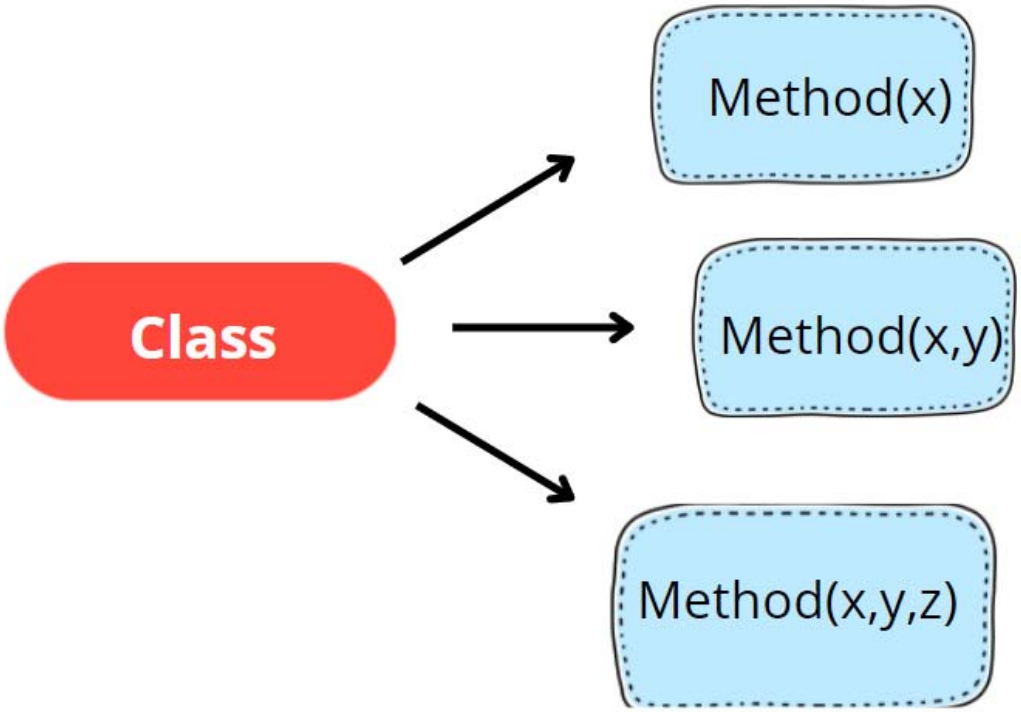


Polymorphism: Overloading

- Method overloading is a form of polymorphism in OOP.
- Polymorphism allows objects or methods to act in different ways, according to the means in which they are used.
- One such manner in which the methods behave according to their argument types and number of arguments is method overloading.

Polymorphism: Overloading

- Overloading



Polymorphism: Overloading

- Method overloading can be achieved by the following:
  - By changing the number of parameters in a method
  - By changing the order of parameters in a method
  - By using different data types for parameters

Polymorphism: Overloading

- A very common example, to find the area of any polygon:

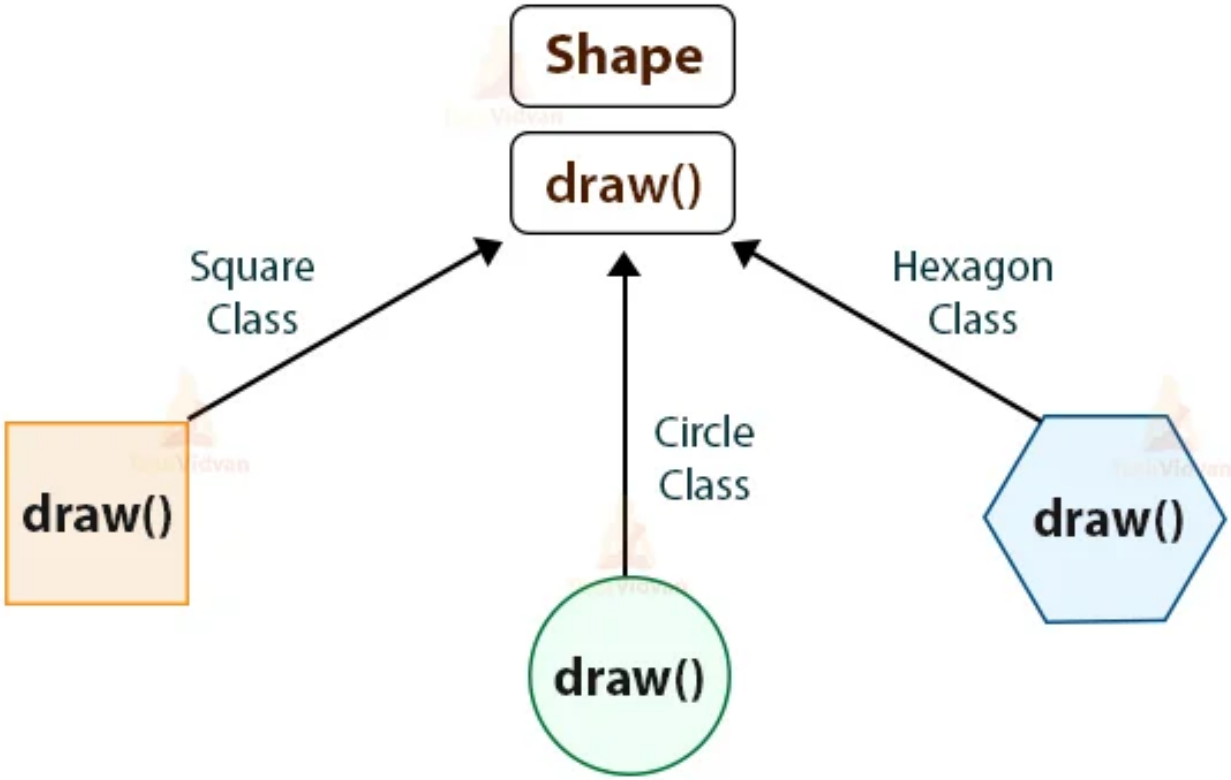
```
1  public class Area {
2      public double area(double s) {
3          double area = s * s;
4          return area;
5      }
6
7      public double area(double l, double b) {
8          double area = l * b;
9          return area;
10     }
11 }
```

Polymorphism: Overriding

- Overriding is an object-oriented programming feature that enables a child class to provide different implementation for a method that is already defined and/or implemented in its parent class or one of its parent classes.
- The overridden method in the child class should have the same name, signature, and parameters as the one in its parent class.

Polymorphism: Overriding

- Overriding



Polymorphism: Overriding

- Overriding enables handling different data types through a uniform interface.
- Hence, a generic method could be defined in the parent class, while each child class provides its specific implementation for this method.