OOP

Prattya Datta

Task organization

Task organization

Task	Priority	Description	Difficulty	Time
Read the description of the project	5	This task involves complete understanding of requirements for the project	1	30 mins
Create git repo	5	Creating of git repository for project execution and delivery	1	2 mins
Create the directory structure	4	Creating the necessary files, folders and subfolders for this project.	1	2 mins
OOP theoretical	4	Understand completely the overview of OOP	2	1-2 hrs
OOP practical	4	Use the understanding gained in the previous section to apply in practical purposes	1	30mins-2 hrs
Presentation	3	Create a presentation for the project	1	1 hr
Documentation	1	Write the documentation for this project	1	3-4 hr

Use cases and Conclusions

OOP use cases

- You have multiple programmers who don’t need to understand each component.
- There is a lot of code that could be shared and reused.
- The project is anticipated to change often and be added to over time.
- Different sections can benefit from different resources like datasource or hardware.

Advantages and Disadvantages

Advantages include:

- Modular: Provides a clear modular structure for programs which makes it good for defining abstract data-types where implementation details are hidden and the unit has a clearly defined interface.
- Scaleable: Adding developers to a project often easier because the don't have to understand the entire code base, just the section they're working on. Adding Hardware resources can be more cost effective because you can have different resources for each modular piece.
- Maintainable: Makes it easy to maintain and modify existing code as new objects can be created with small differences to existing ones.
- Extensible: Provides a good framework for extending a project through libraries where these components can be easily adapted and modified by the programmer. This is particularly useful for developing graphical user interfaces, or GUIs.
- Reusable: Each module works independently of the surrounding module. This allows you take take one section, such as the user login, and use it for other projects.

Advantages and Disadvantages

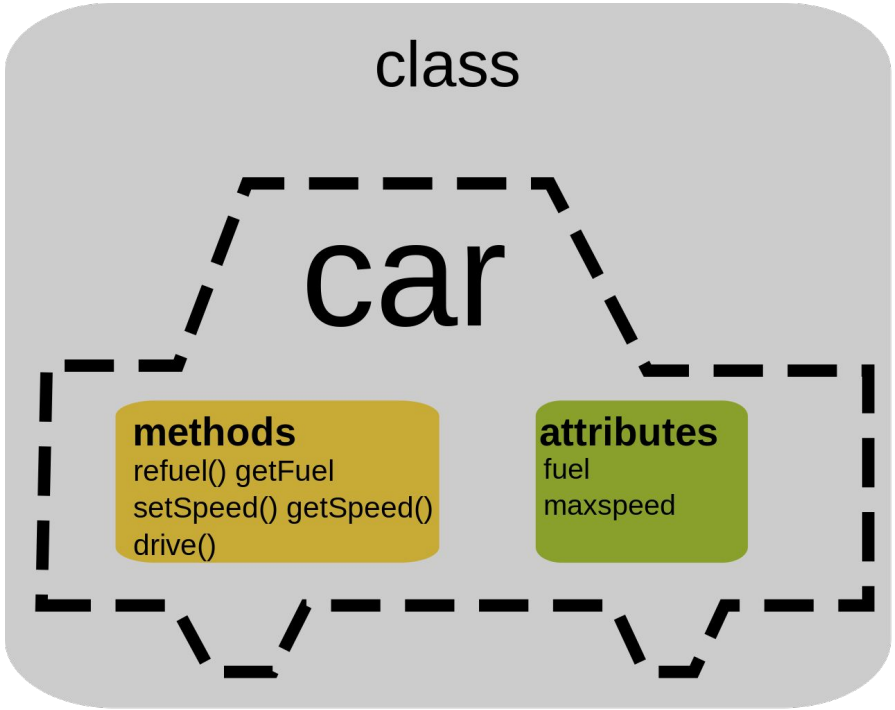
Disadvantages include:

- The real world refuses to divide up into neat classes and subclasses.
- Sometimes several objects will interact in complex ways - maybe even ways we didn't necessarily anticipate when writing the program.
- Can require more code and be more complicated for smaller projects or project in which there are very few repeated tasks.
- Decreased performance. This is one of the most heated debates. Although a well designed procedural site CAN slightly out perform a well designed object-oriented site, there are many factors to consider and this should not be your main concern.

Definitions

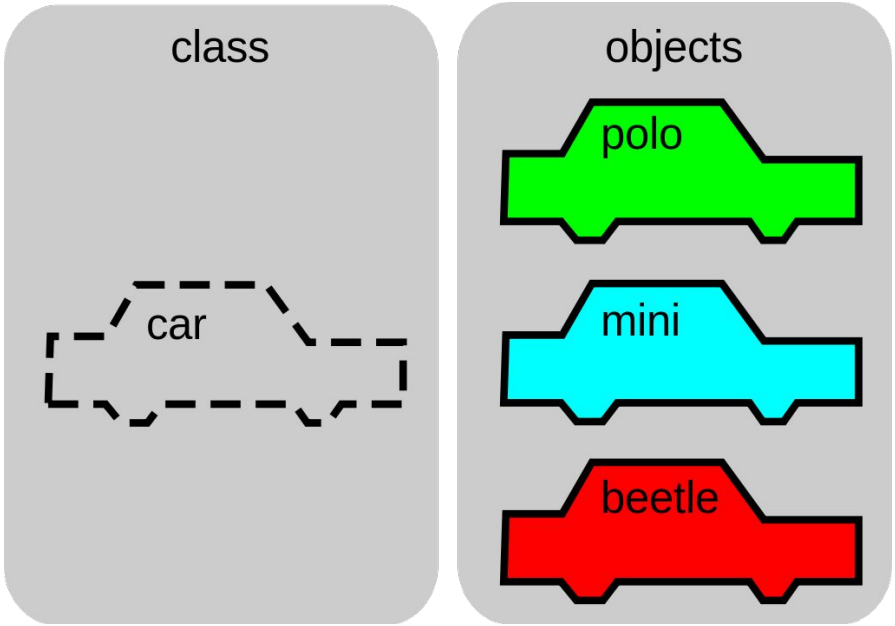
Class

In object-oriented programming, a class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).



Object

The user-defined objects are created using the class keyword. The class is a blueprint that defines the nature of a future object. An instance is a specific object created from a particular class. Classes are used to create and manage new objects and support inheritance—a key ingredient in object-oriented programming and a mechanism of reusing code.

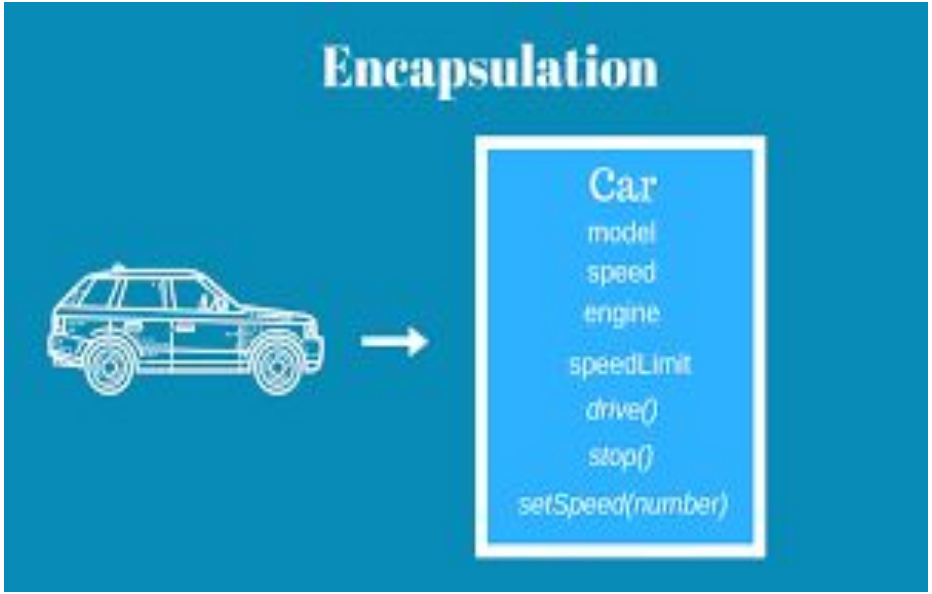


Instance

An instance is a unique copy of a Class that represents an Object. When a new instance of a class is created, the program will allocate a room of memory for that class instance.

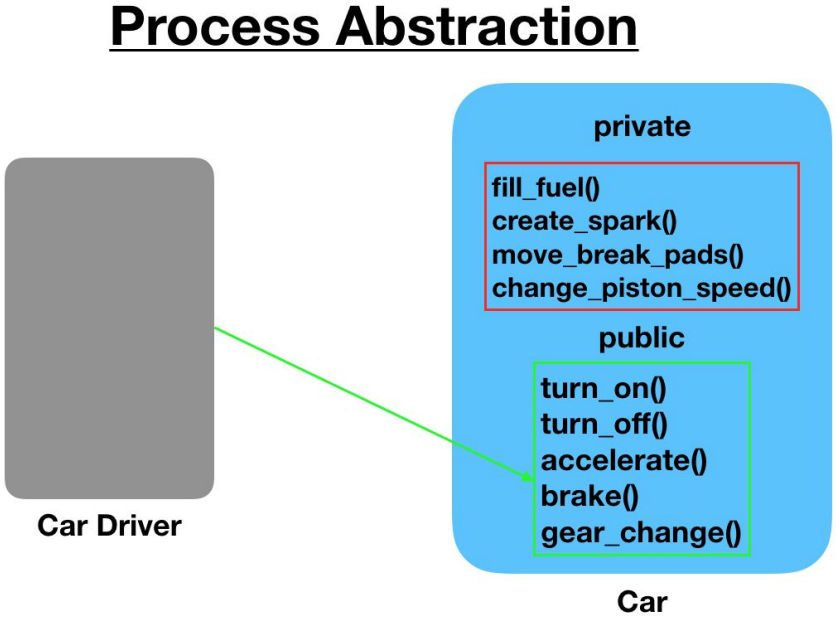
Encapsulation

Encapsulation refers to the bundling of data with the methods that operate on that data, or the restricting of direct access to some of an object's components. Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them.



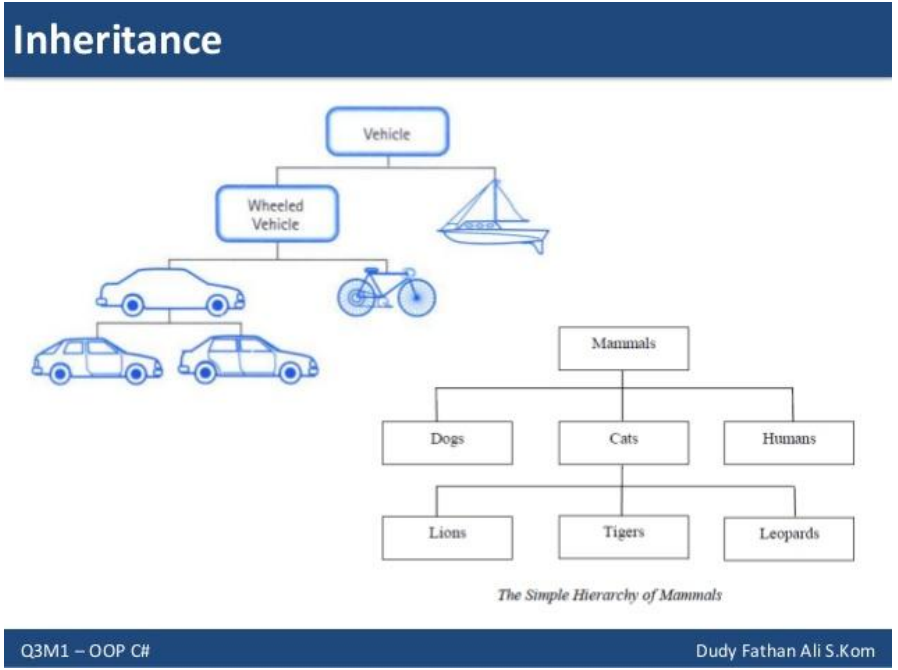
Abstraction

Abstraction involves the facility to define objects that represent abstract "actors" that can perform work, report on and change their state, and "communicate" with other objects in the system.



Inheritance

It provides a mechanism for establishing relationships and building hierarchies of class in object composition. Inheritance means the use of code that is pre-written or created previously.



Static class

Statics are like little gatekeepers or the mediators between the internal communication of source code and its data propagating throughout a program at runtime. Therefore, statics, especially - play an important role in Object Oriented Programming (OOP).

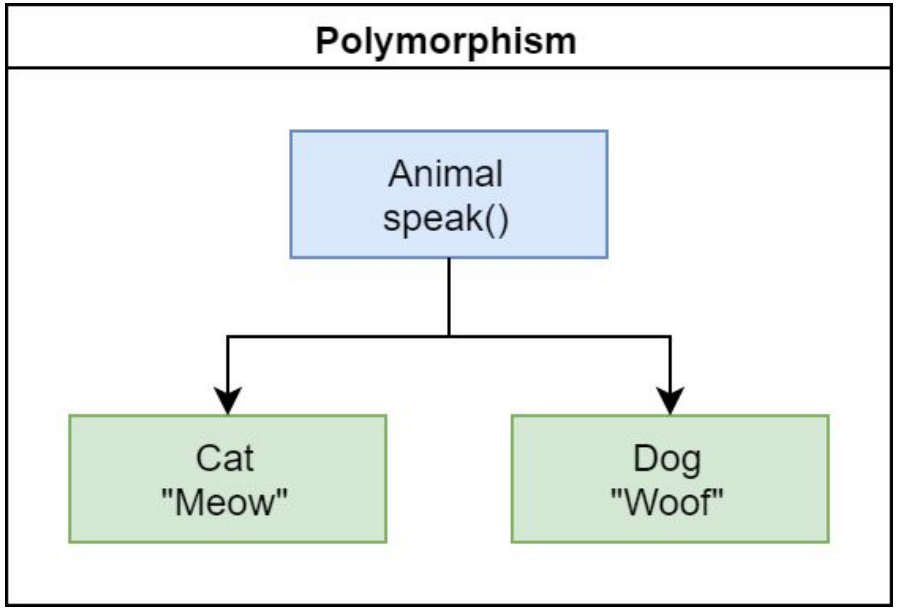
Non-Static Vs Static Class

- Non-Static: need New() to instantiate / create an object – like what you see just now
- Static: no need to use New() to use, there is just one copy of the class. This type of class basically to provide special functions for other objects
- So if you see a class being used without New(): it is a static class
- Eg Math class

```
age = Math.Round(18.5);  // Math rounding
```

Polymorphism

polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes.



Overloading and overriding

When two or more methods (functions) in the same Class have the same name but different parameters is called method overloading.

When two or more methods (functions) have the exact same method name, return type, number of parameters, and types of parameters as the method in the parent class is called method Overriding.