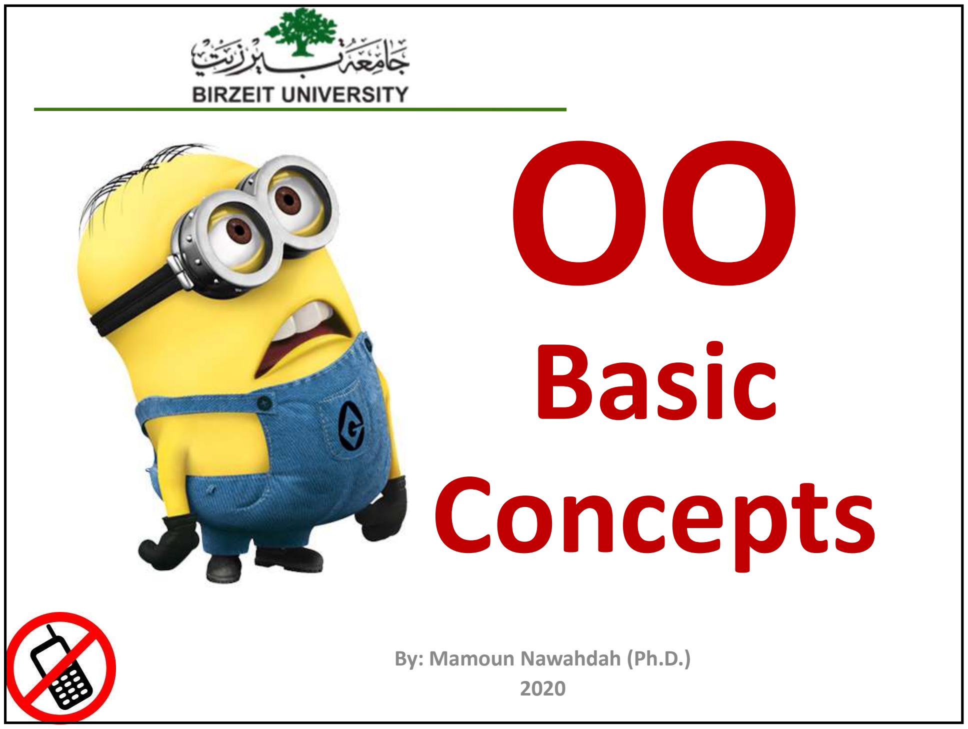
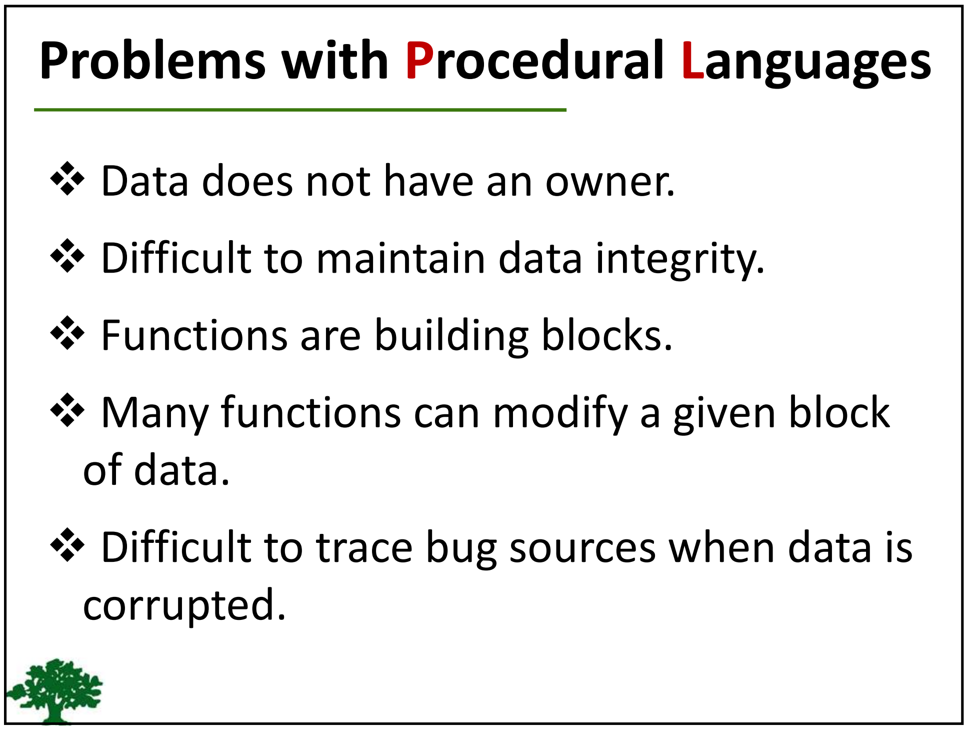


جامعة بيرزيت
BIRZEIT UNIVERSITY

OO Basic Concepts

By: Mamoun Nawahdah (Ph.D.)
2020

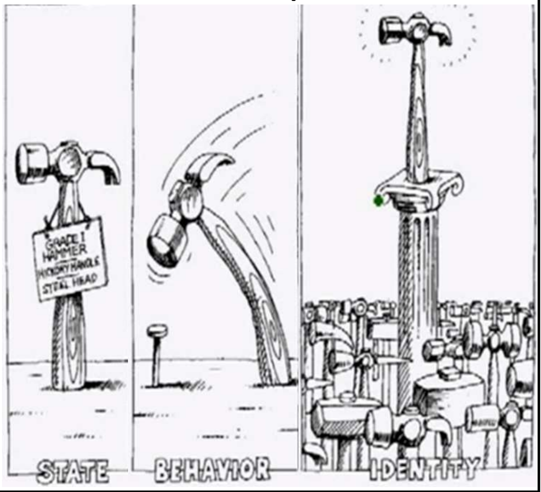


Problems with Procedural Languages

- ❖ Data does not have an owner.
- ❖ Difficult to maintain data integrity.
- ❖ Functions are building blocks.
- ❖ Many functions can modify a given block of data.
- ❖ Difficult to trace bug sources when data is corrupted.

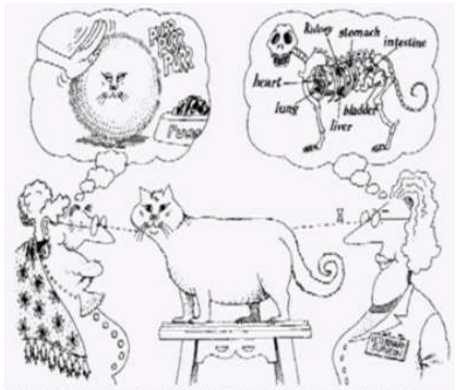
What is Object?

- ❖ An object has **state**, exhibits some well defined **behavior**, and has a unique **identity**.
- ❖ **State**:
  - Data members
  - Fields
  - Properties
- ❖ **Behavior**:
  - Member functions
  - Methods

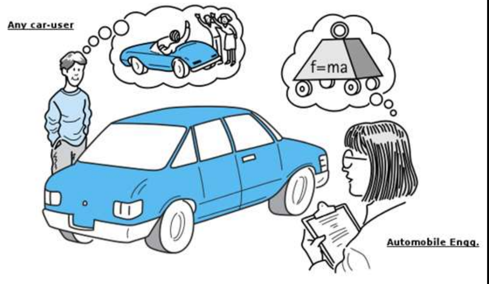


Abstraction - Modeling

- ❖ **Abstraction** focuses upon the **essential** characteristics of some object, relative to the perspective of the viewer.



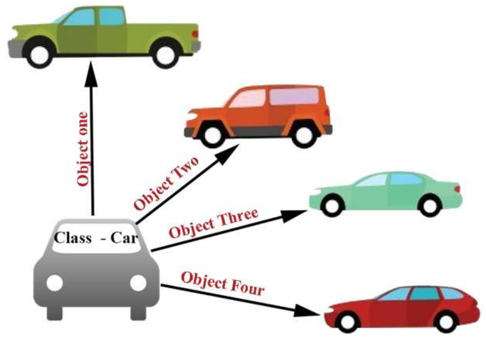
Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.



An abstraction includes the essential details relative to the perspective of the viewer

What is Class?

- A **class** represents a set of objects that share common structure and a common behavior.
- A **class** is a **blueprint** or **prototype** that defines the variables and methods common to all objects of a certain kind.



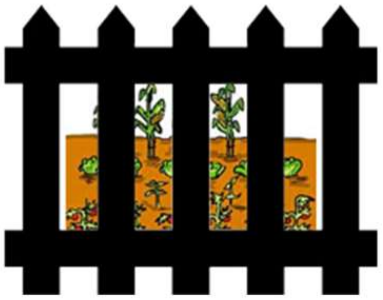
Class (Car)	
Attributes	Methods
Company	Move
Type	Stop
Model	Turn Right
Color	Turn Left

Class Access

- **Problem**: You have a garden and it is **public**. Anyone can take the properties of the garden when they want.

Class Access cont.

- **Solution**: Make it **private**, put a high fence around my garden, now it is safe!



- But wait, I can no longer access my own garden!!!!

Class Access cont.

- **Solution**: Hire a private guard and give him **rules** on who is able to access the garden.
- Anyone want use the garden must get permission from guard.
- Garden is now **safe** and **accessible**.

Class Access cont.

Setters and Getters to Safeguard Data
Data
Set Property
Get Property
Outside Requester
Private Property
Setter&Getter
Requester

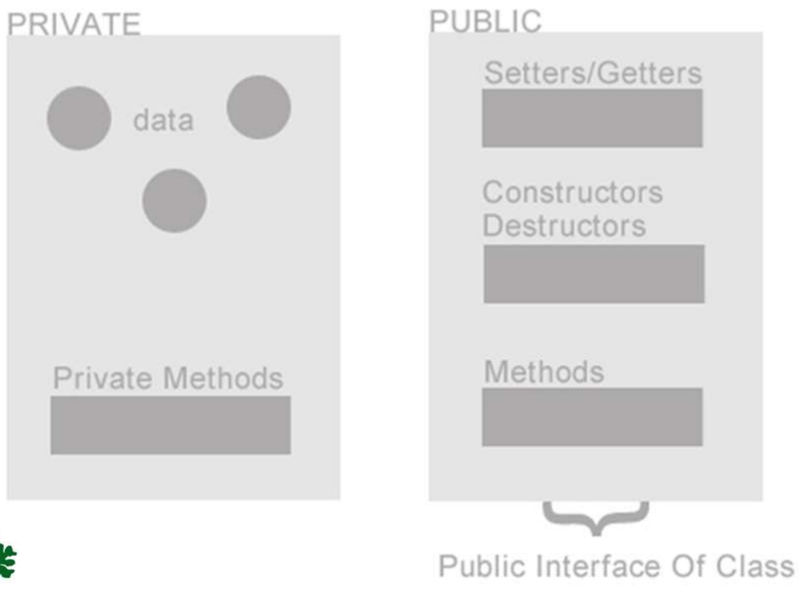
Initialization of Objects

- **Constructors** ensure correct initialization of all data. They are automatically called at the time of object creation.
- **Destructors** on the other hand ensure the de allocation of resources before an object dies or goes out of scope.

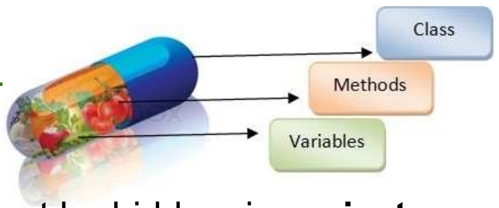
Lifecycle of an Object

- **Born Healthy**:
  - Using **constructors**
- **Lives Safely:**
  - Using **setters** and **getters**
- **Dies Cleanly:**
  - Using **destructors**

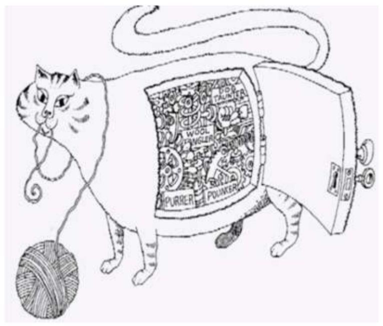
Anatomy of a Class



Encapsulation



- ❖ **1st law of OOP**: data must be hidden, i.e. **private**.
- ❖ Read access through read functions.
- ❖ Write access through write functions.
- ❖ For every piece of data, 4 possibilities:
  - Read and write allowed
  - Read only
  - Write only
  - No access



Encapsulation

- ❖ **Encapsulation** is used to hide unimportant implementation details from other objects.
- ❖ In real world, when you change gears on your car:
  - You don't need to know how the gear mechanism works.
  - You just need to know which lever to move.

Encapsulation cont.

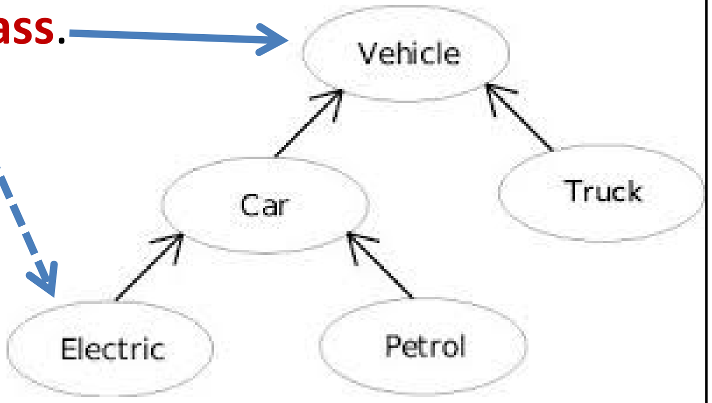
- ❖ In software programs:
  - ▪ You don't need to know how a class is implemented. (e.g. **Math** class)
  - ▪ You just need to know which methods to invoke. (e.g. **pow** method)
- ❖Thus, the implementation details can change at any time without affecting other parts of the program.

Inheritance

- ❖ **Extending** the functionality of a class or
- ❖ **Specializing** the functionality of the class.

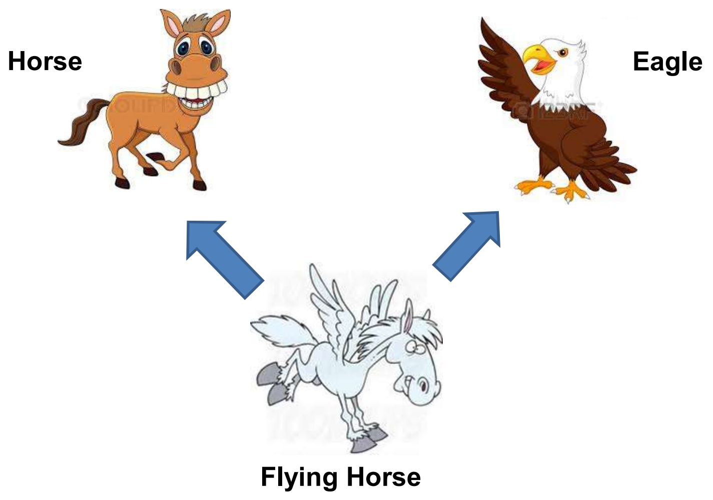
Inheritance cont.

❖ **Subclasses**: a subclass may inherit the structure and behaviour of it's **superclass**.



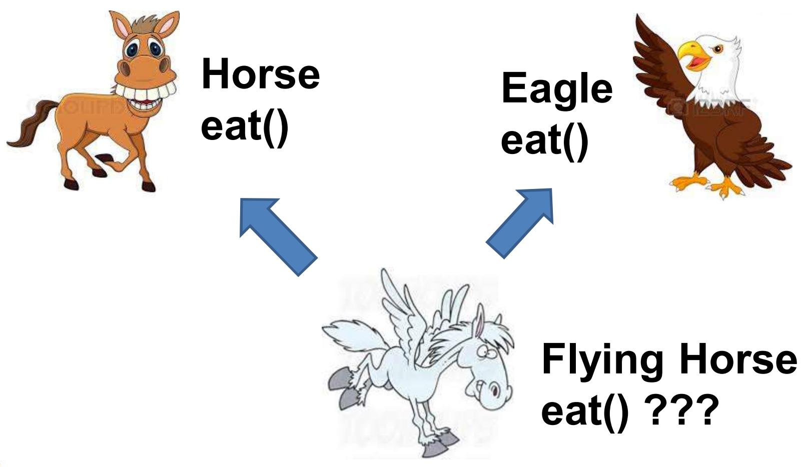
Multiple Inheritance

❖ One class have more than one **superclass**.



Multiple Inheritance cont.

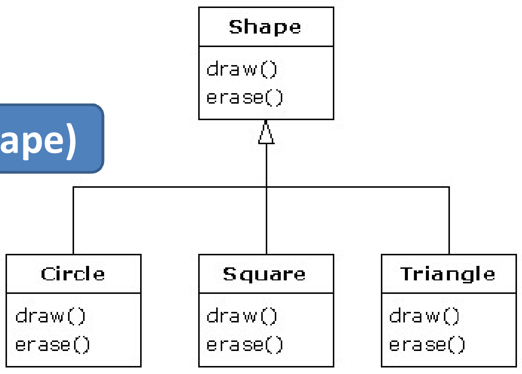
❖ **Ambiguity** in multiple inheritance:



Polymorphism

❖ **Polymorphism** refers to the ability of an object to provide different behaviours (use different implementations) depending on its own nature. Specifically, depending on its position in the class hierarchy.

drawShape (class Shape)


Shape
draw()
erase()
Circle
draw()
erase()
Square
draw()
erase()
Triangle
draw()
erase()