SLIIT

Discover Your Future

Object Oriented Concepts

Lecture-04

Classes & Objects – Part 2

Learning Outcomes

- At the end of the Lecture students should be able to
  - Understand, identify and describe Classes, Objects, Properties and Methods
  - Describe Encapsulation, Information Hiding, and Interfaces

How do we develop an OO Program ?

We look at the problem that needs to be solved

This is the building we want to build

e.g. In a real world scenario this could be a Student Information System (SIS)

Identifying Objects needed

These are the Blocks (Objects) that we need.

e.g. In SIS objects could be details of OOC, IWT, students called Manoj, Gayani

How do you create these Blocks (Objects)?

What if we needed to manufacture these blocks. How could we do this? What do we need to make first?

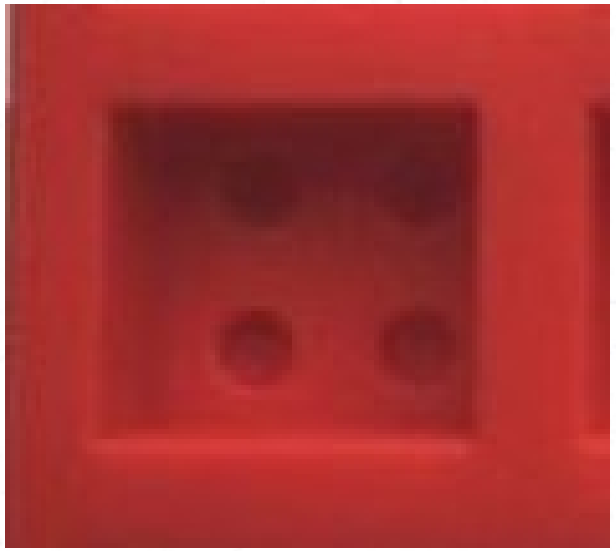
A Mould (Class)

Once we make a Mould (Class) we can make as many blocks (Object) that we need.

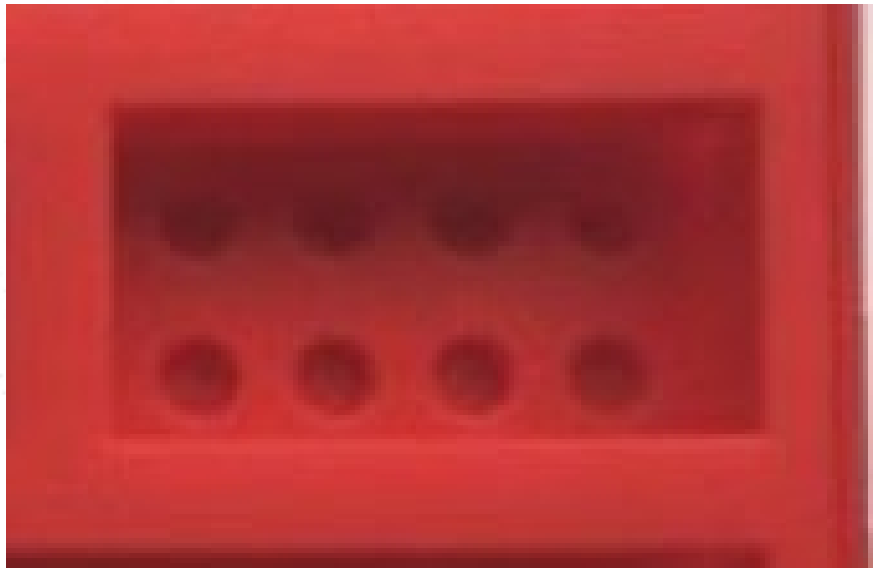
How to group these Blocks (Objects)?

How can we group these Blocks?

We could do it by Shape



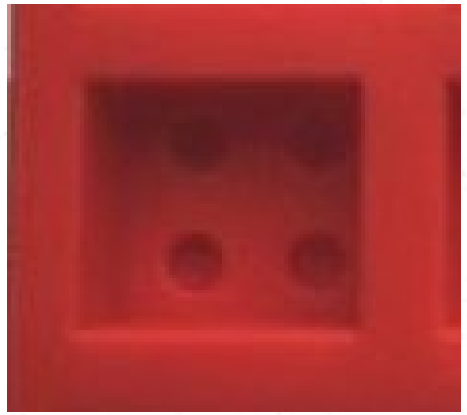
A Square Mould (class)



A Rectangle Mould (class)

e.g. In SIS it could be Student Class, Subject Class

Creating Blocks (Objects) from Moulds (Classes)

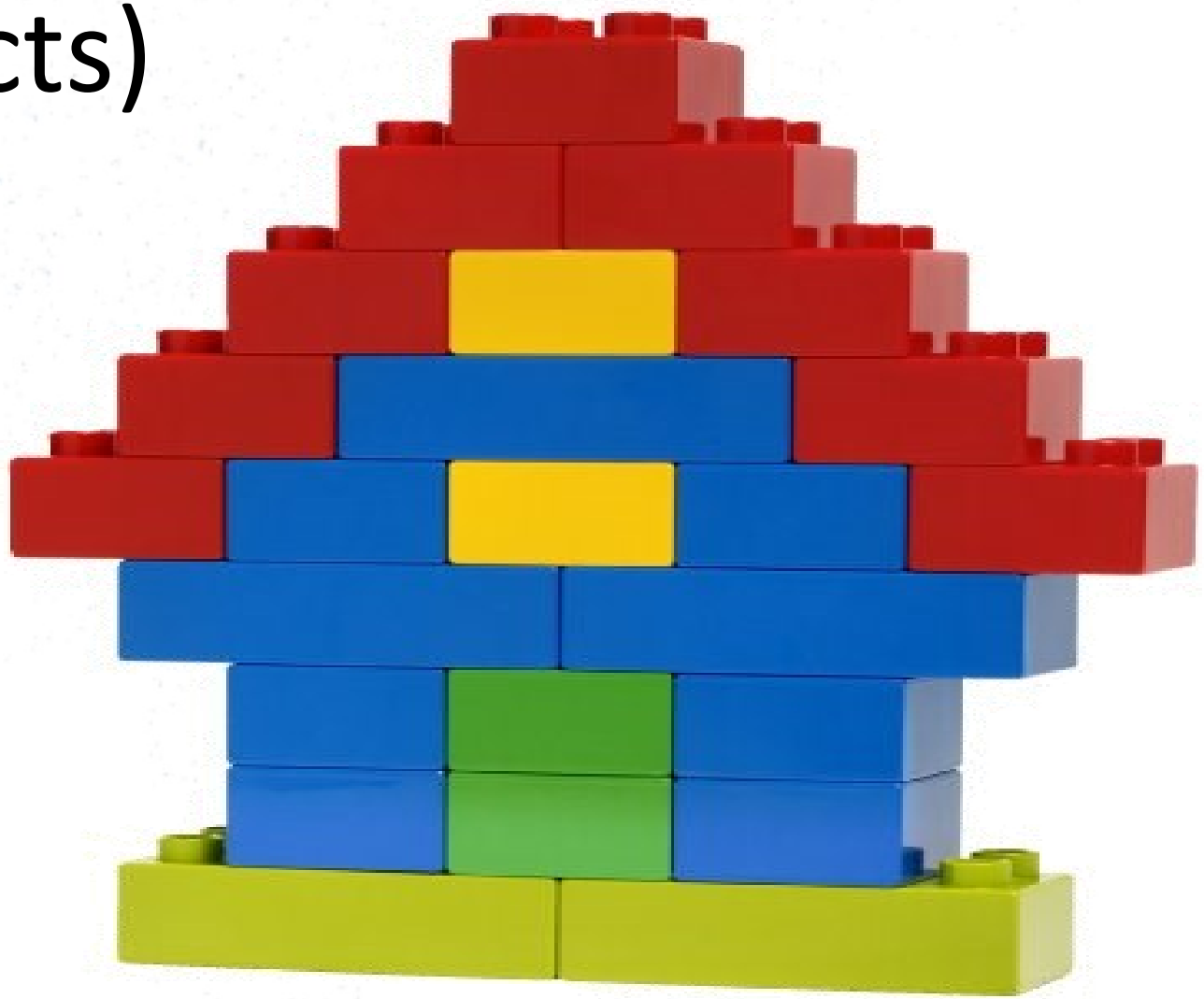


A Square Mould (class)

A Rectangle Mould (class)

Blocks (Objects) made

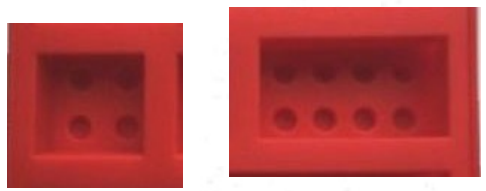
Final Product – Assembling Blocks (Objects)



We can now assemble the Objects and create our final solution.

Problem to Solve

Identify Objects that are needed

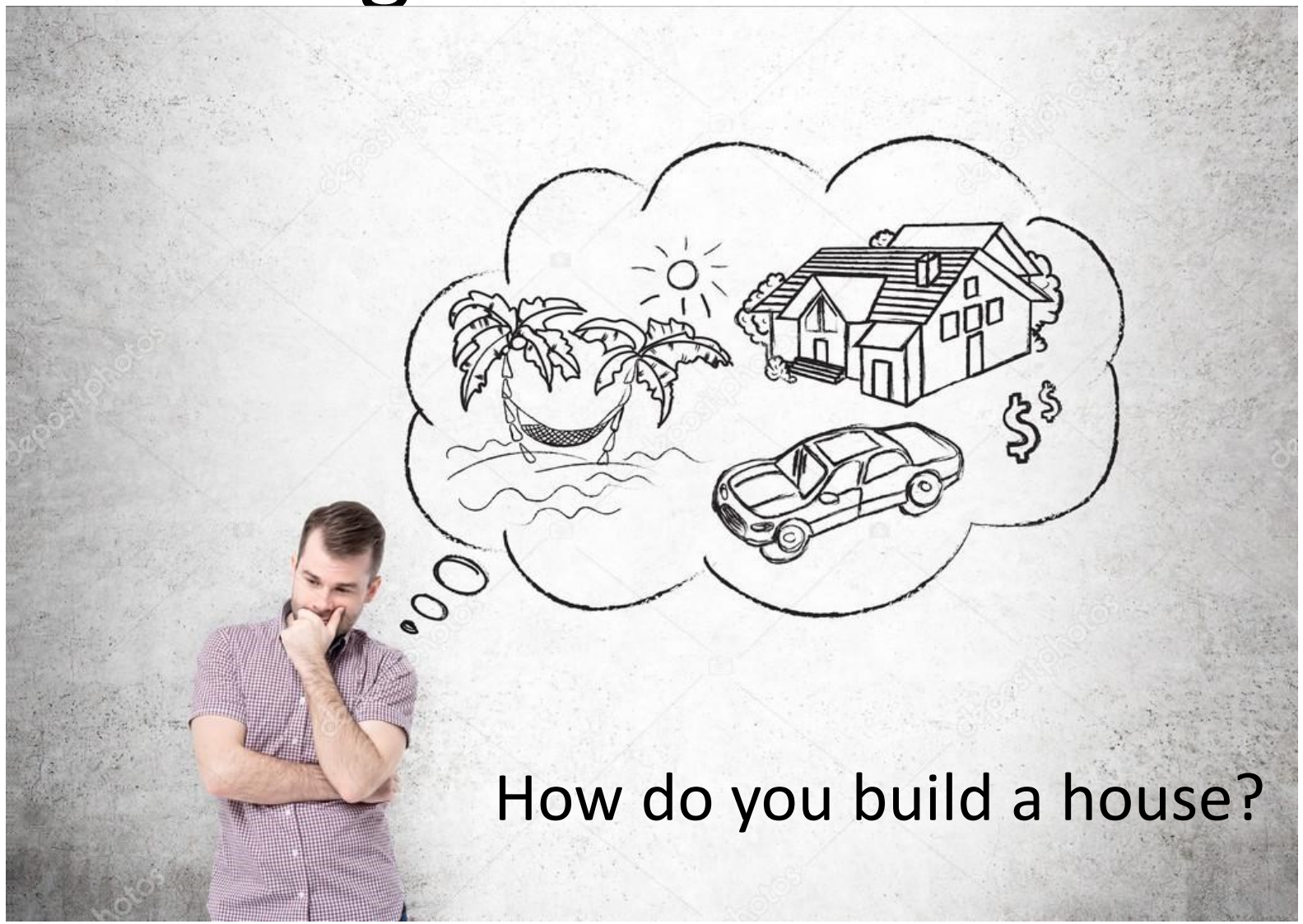


Identify Classes through Abstraction

Assemble Objects to create the solution

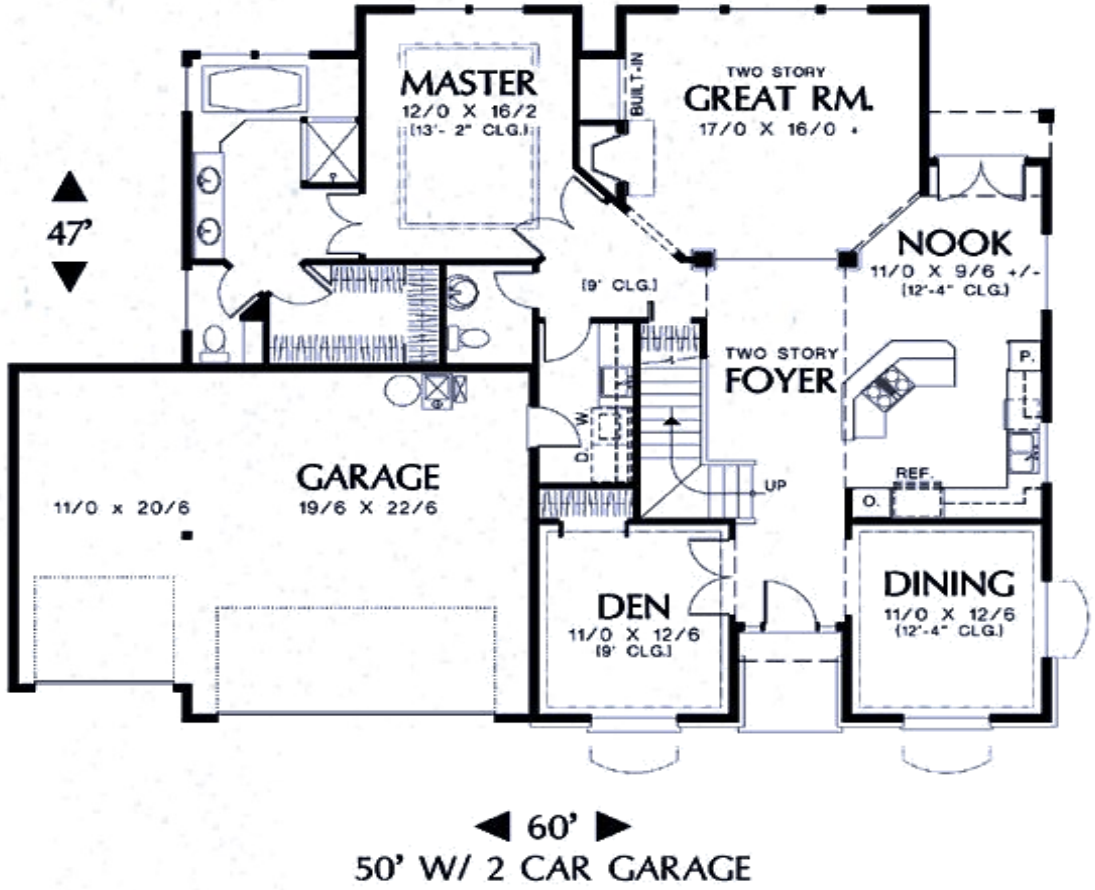
Create Objects from Classes

I am dreaming of



Meeting an Architect to make a House Plan (Blue Print)

Blue Print – House Plan (Class)

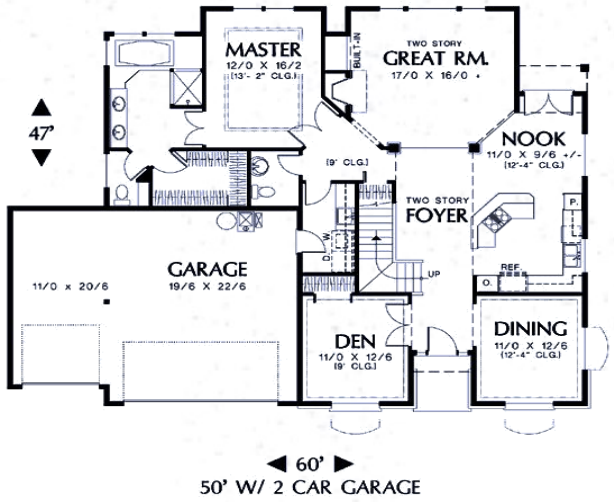


Your Dream House

Class House

Your Dream House

- With the House Plan – Blue Print (Class) you can now get a contractor to build your dream house (Object)

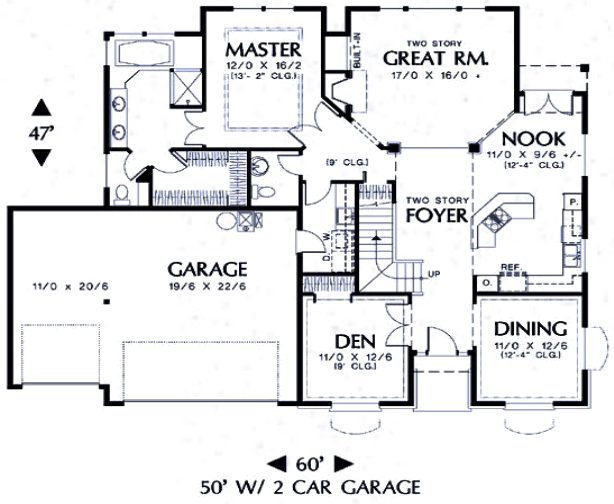


Class House

Object

Classes and Objects

- An Object is a specific instance (variable) of the data type (class)
- A class is a blue print of an object.

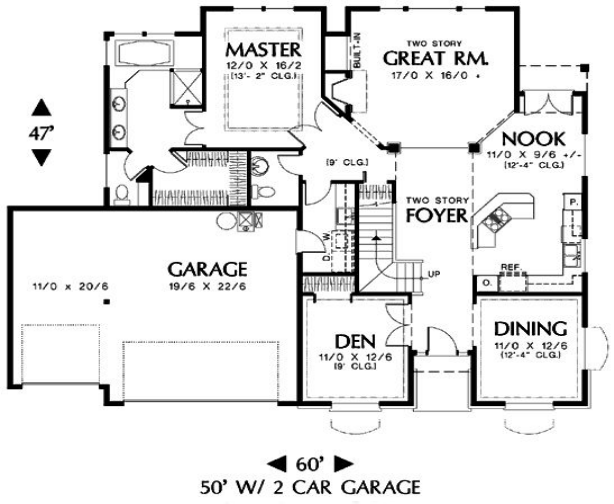


Class House

Object

Classes and Objects

- You can make as many houses as you want from a single house plan – Blue Print (Class)



Class House

House1

House2

House3

Objects

Classes – Captures Behavior as well

- A concept (class) has both properties and behavior.
- We know that dogs and cats behave differently

Dog Class

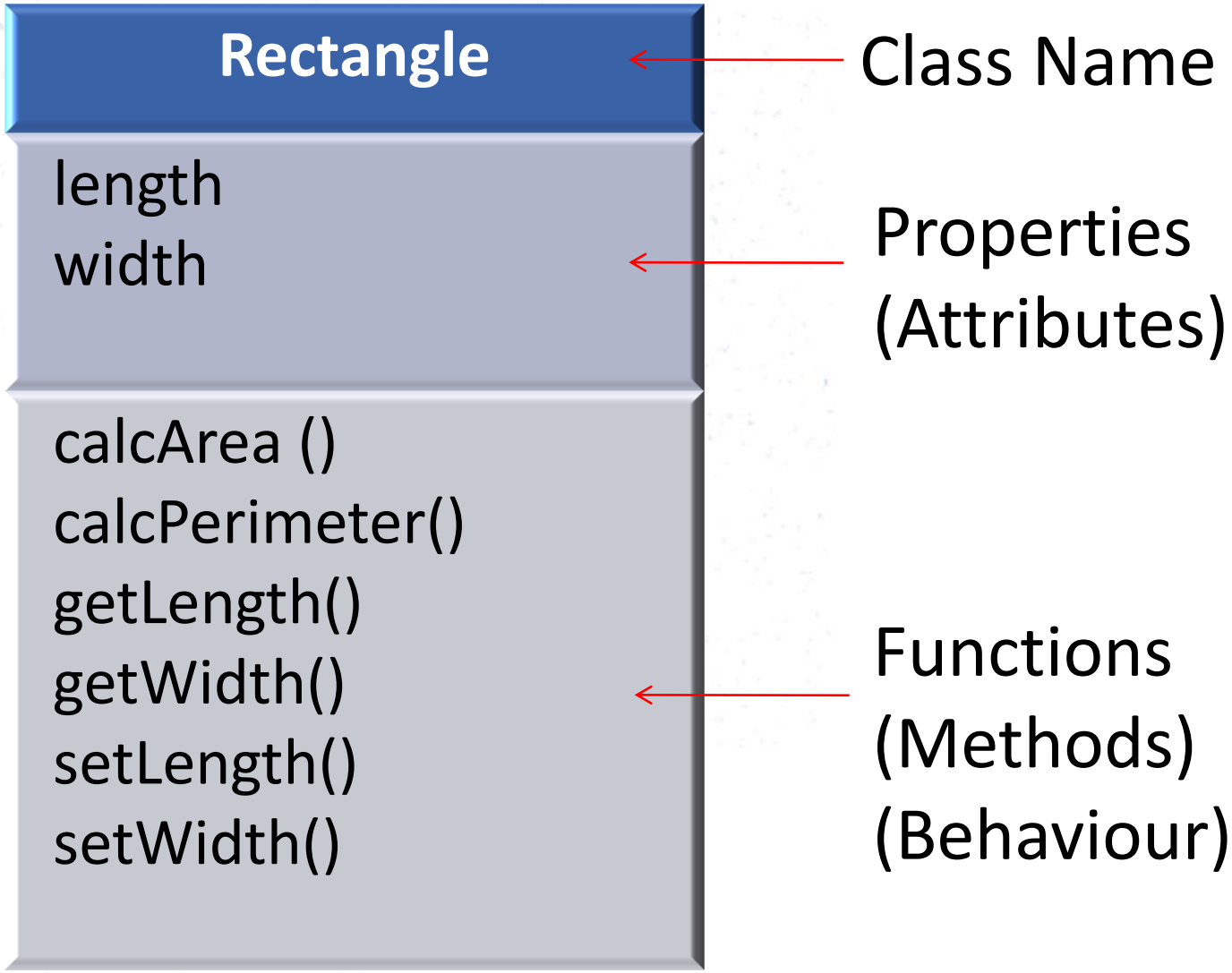
Cat Class

Example – Behaviour is captured as functions

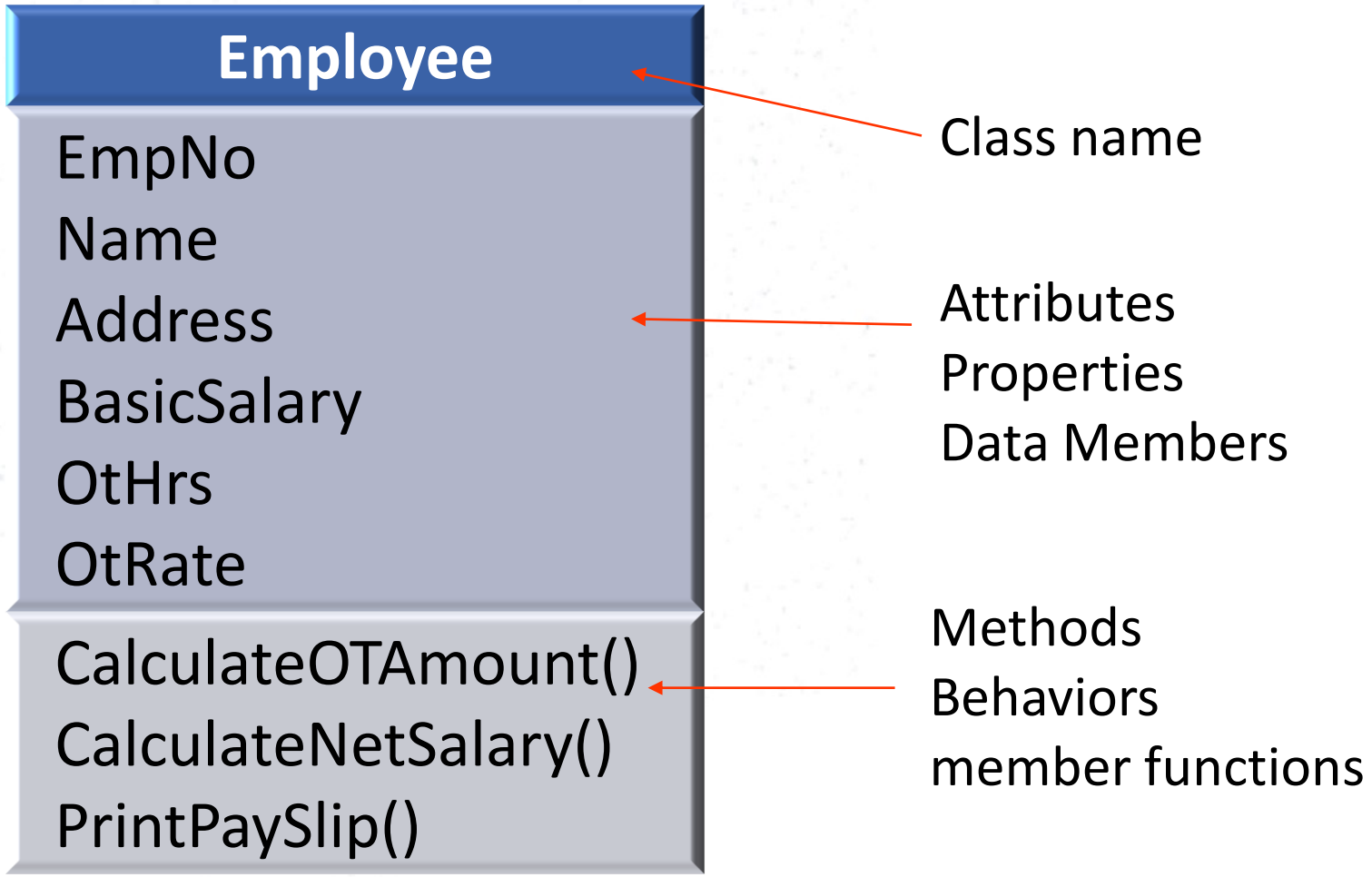
Dog Class	Cat Class
name owner breed work *(e.g. police dog)*	name owner breed
Bark() Fetch()	Meow() Purr()

Grouping properties and functions together is called Encapsulation

Rectangle Class



Employee Class



Restricted Access

- All properties and some functions of a Class have restricted access (private) and can be accessed only through public functions.
- Why is this necessary?
- Let's look at an example.

A Jewelry Shop

Jewelry can be accessed only through a Sales Person

JewelleryShop Class

Private
(Restricted)

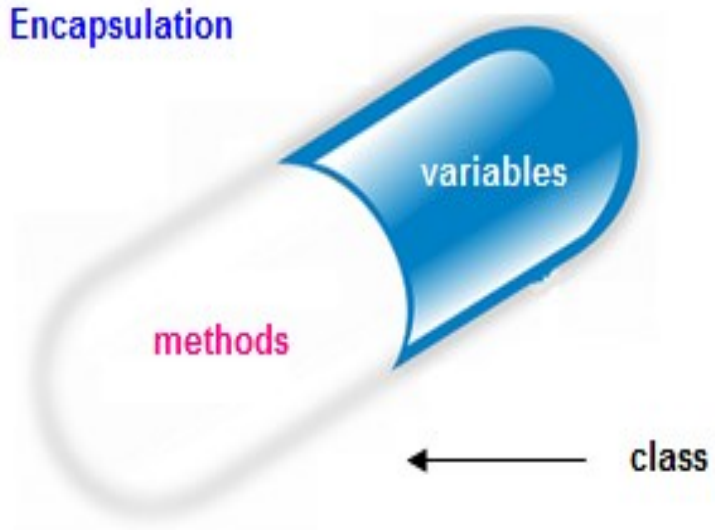
Public
(Salesman
do this)

Information Hiding

- Hide certain information or implementation decision that are internal to the encapsulation structure (class)
- The only way to access an object is through its public interface (public functions)
  - Public – anyone can access / see it
  - Private – no one except the class can see/ use it

Encapsulation

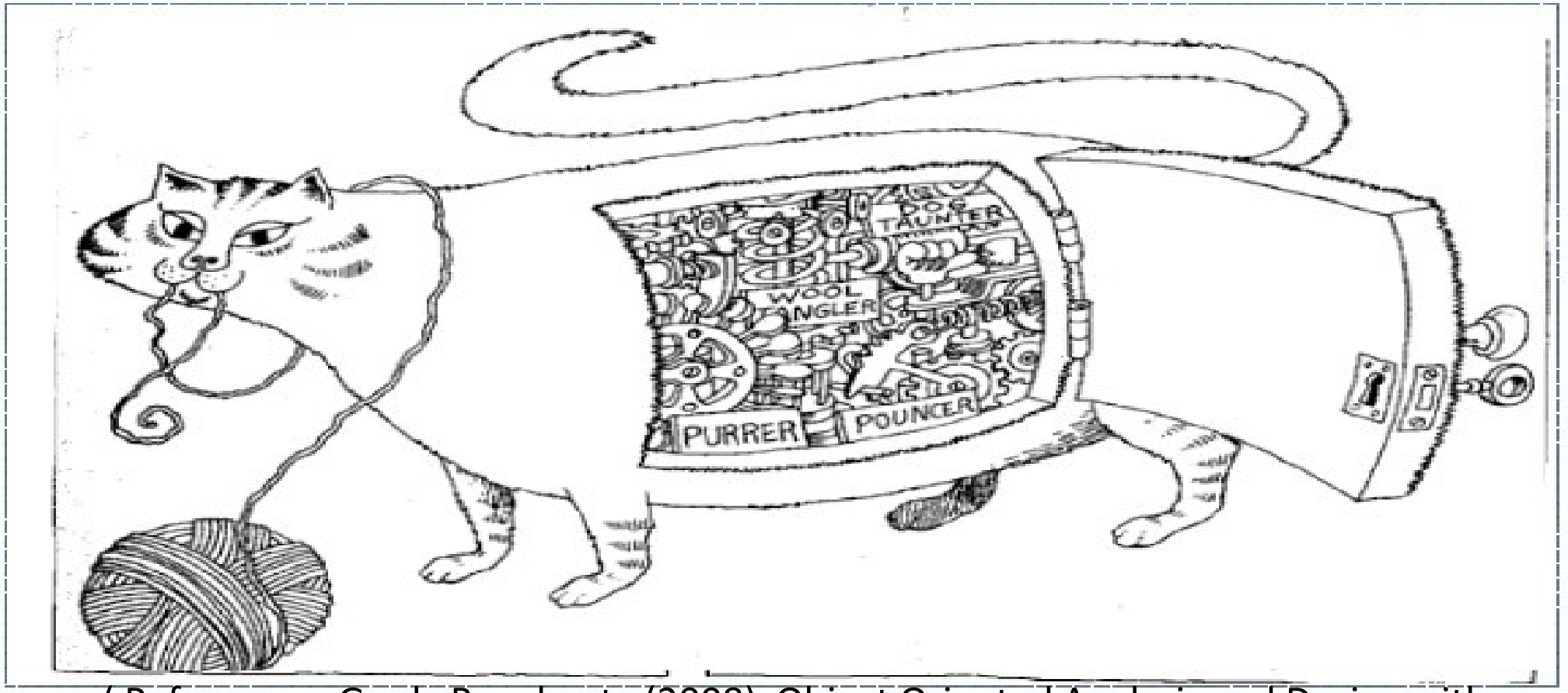
- It is the process of grouping related attributes and methods together, giving a name to the unit and providing an interface (public functions) for outsiders to communicate with the unit.



Encapsulation

- The implementation of the TV is hidden from us. Your TV could be OLED, LCD, Plasma or an old CRT one.
- You can control any such TV using the same commands in your remote (interface).

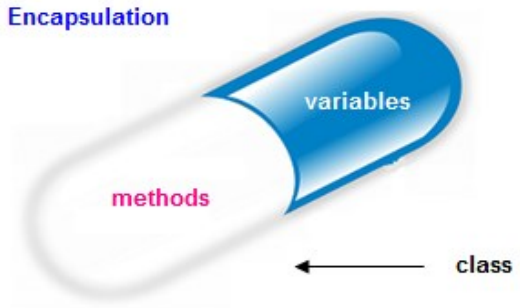
Encapsulation hides the details of the implementation of an object



(Reference : Grady Booch, eta (2008), Object Oriented Analysis and Design with Applications 3rd Edition, pg 52)

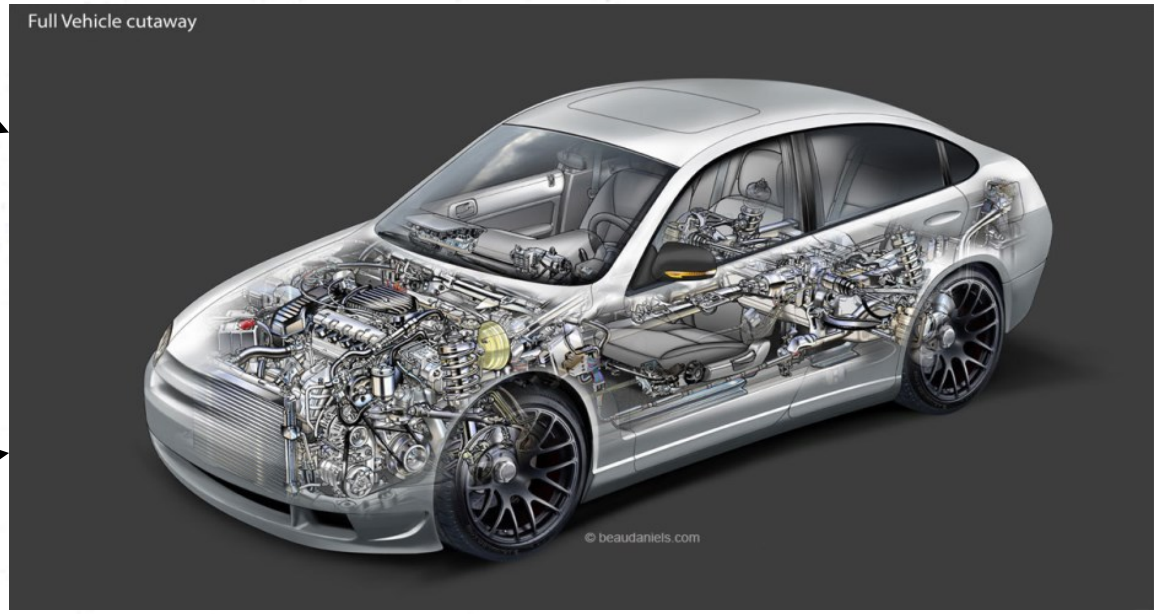
Encapsulation

- Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.



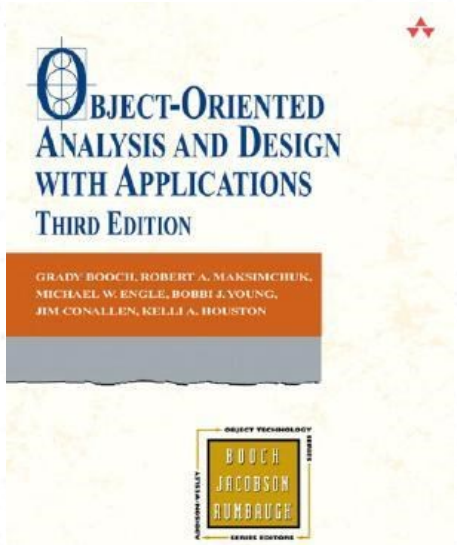
- (Reference : Grady Booch, eta (2008), Object Oriented Analysis and Design with Applications 3rd Edition, pg 52)

Interface – Public Functions



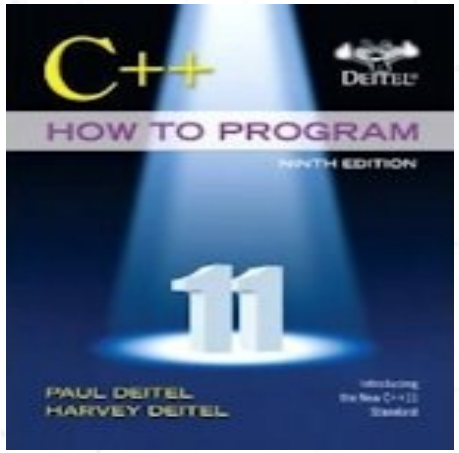
You interact with some object through its interfaces. Your car can be Gasoline, Hybrid or Electric but you drive it the same way.

Reference



Chapter 03

Grady Booch (2008), Object-Oriented Analysis and Design with Application,
3rd Edition



Chapter 09

Deitel & Deitel's (2016), C++ How to Program,
9th Edition