SLIIT

Discover Your Future

IT2030
Object Oriented Programming

Lecture 02
Object Oriented Concepts Recap

Dr. Kalpani Manathunga

Learning Outcomes

At the end of the Lecture you should know

- Object Oriented Programming
- Classes and Objects
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Interfaces

We will also look at key differences between writing simple object-oriented programs in C++ and Java.

Object Oriented Programming

- Object Oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

(Reference : Grady Booch, et.al (2008), Object Oriented Analysis and Design with Applications 3rd Edition, pg 41)

Object Oriented Programming

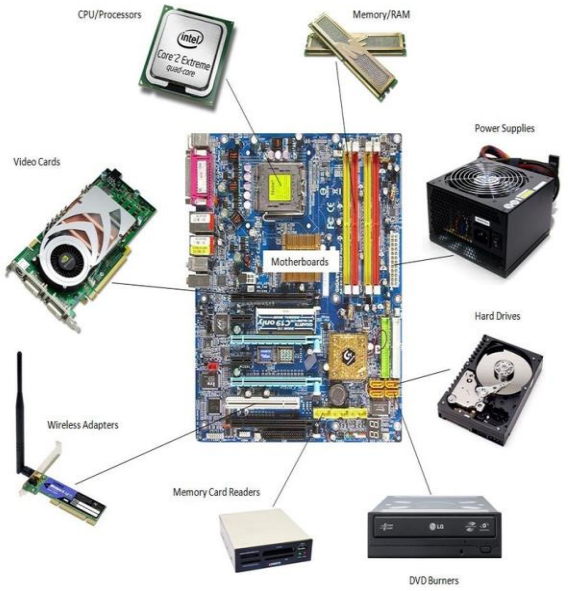
- Programs are organized as a collection of objects which cooperate to solve a problem.
- Allows to solve more complex problems easily.
- Objects contain both data and methods needed.

Object Oriented Programming

- A complex system is developed using smaller sub systems
- Sub systems are independent units containing their own data and functions
- Can reuse these independent units to solve many different problems

A Computer System

Basic parts of a Computer

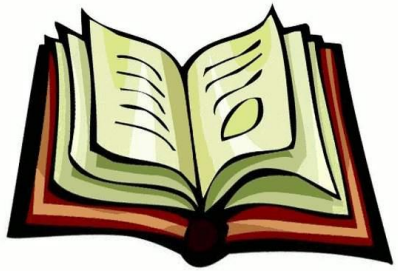


Classes

- A class is the abstract definition of the data type. It includes the data elements that are part of the data type, and the operations which are defined on the data type.
- It is an **Entity** which could be a thing, person or something that is imaginary.





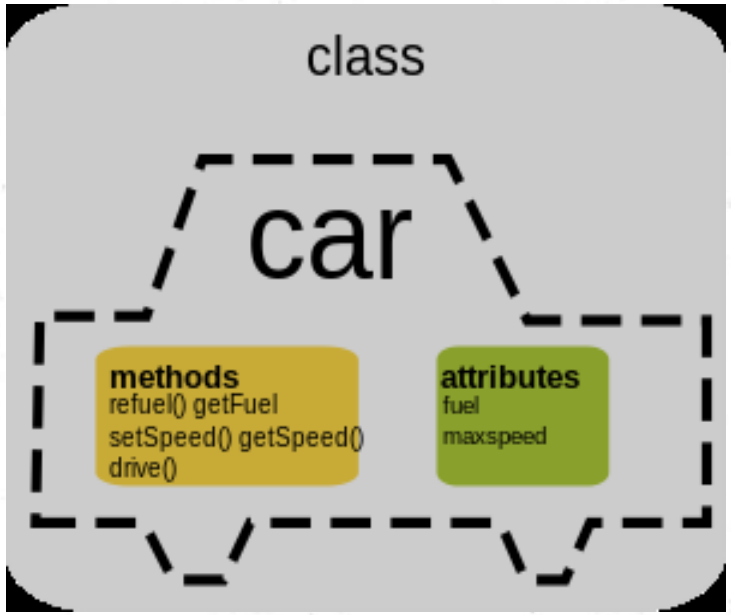




Classes

- An entity can be described by the data (properties) and its behavior (methods)

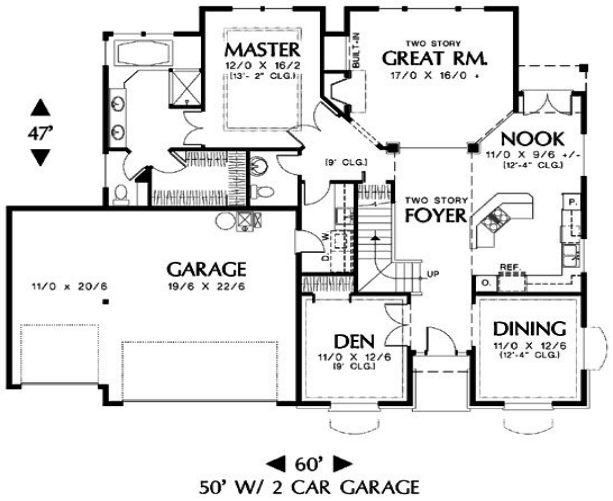
Class Name
Attributes/Properties
Methods

e.g.:

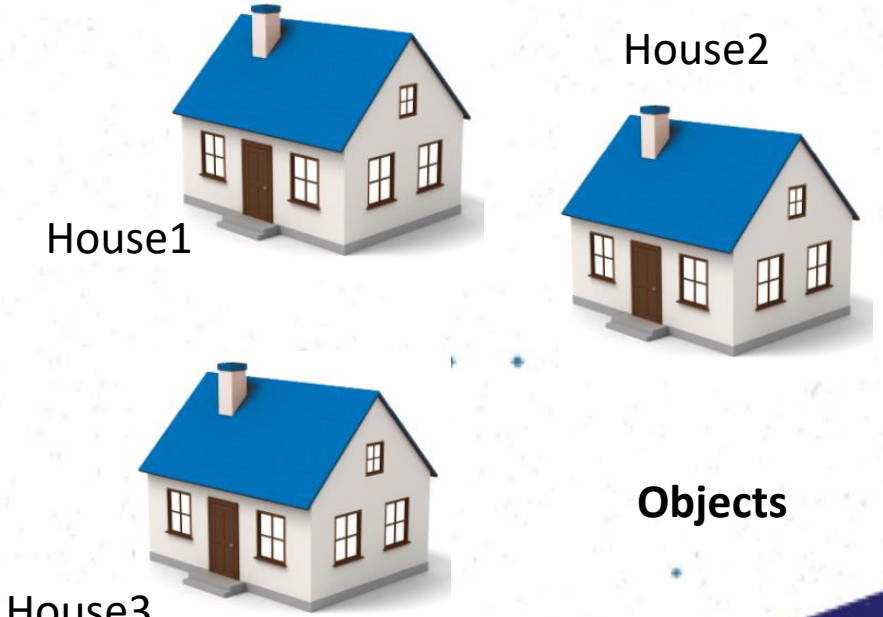


Classes and Objects

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



Class House

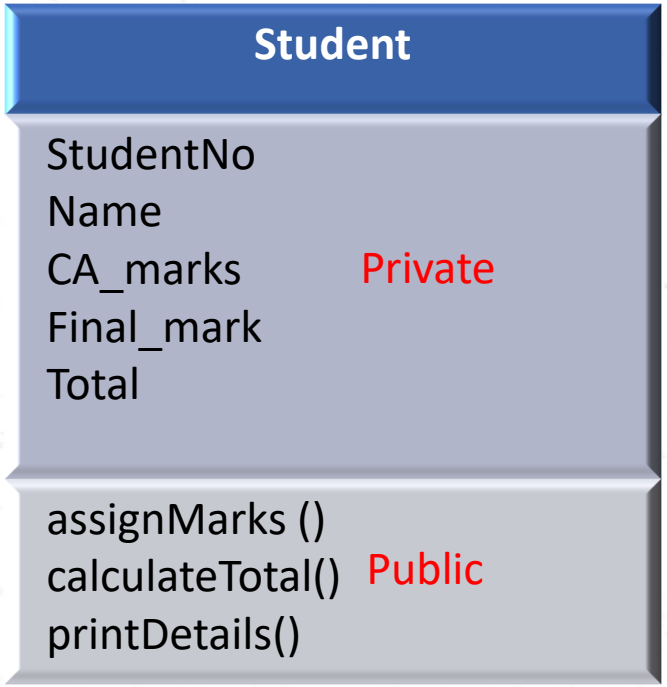


Objects

Objects

- Objects are instances of classes, which we can use to store data and perform actions.
- We need to define a class including the properties and methods.
- Then create as many objects as needed which have the same structure of the class (House example).

Class in C++




```
class Student{
   private :
            int studentNo;
            char name[30];
            int CA_mark;
            int Final_mark;
    public:
            void assignMarks(int pCA, int pFin);
            int calculateTotal();
            void printDetails();
};
```

Class in Java

```
class Student{
   private :
         int studentNo;
         char name[30];
          int CA_mark;
          int Final_mark;
    public:
         void assignMarks(int pCA, int pFin) {
          }
          int calculateTotal() {
          }
          void printDetails() {
          }
};
```

C++

```
class Student {
        private int studentNo;
        private String name;
        private int CA_mark;
        private int Final_mark;

         public void assignMarks(int pCA, int pFin)
             {   }
         public int calculateTotal() {
         }
         public void printDetails() {
         }
}
```

Java

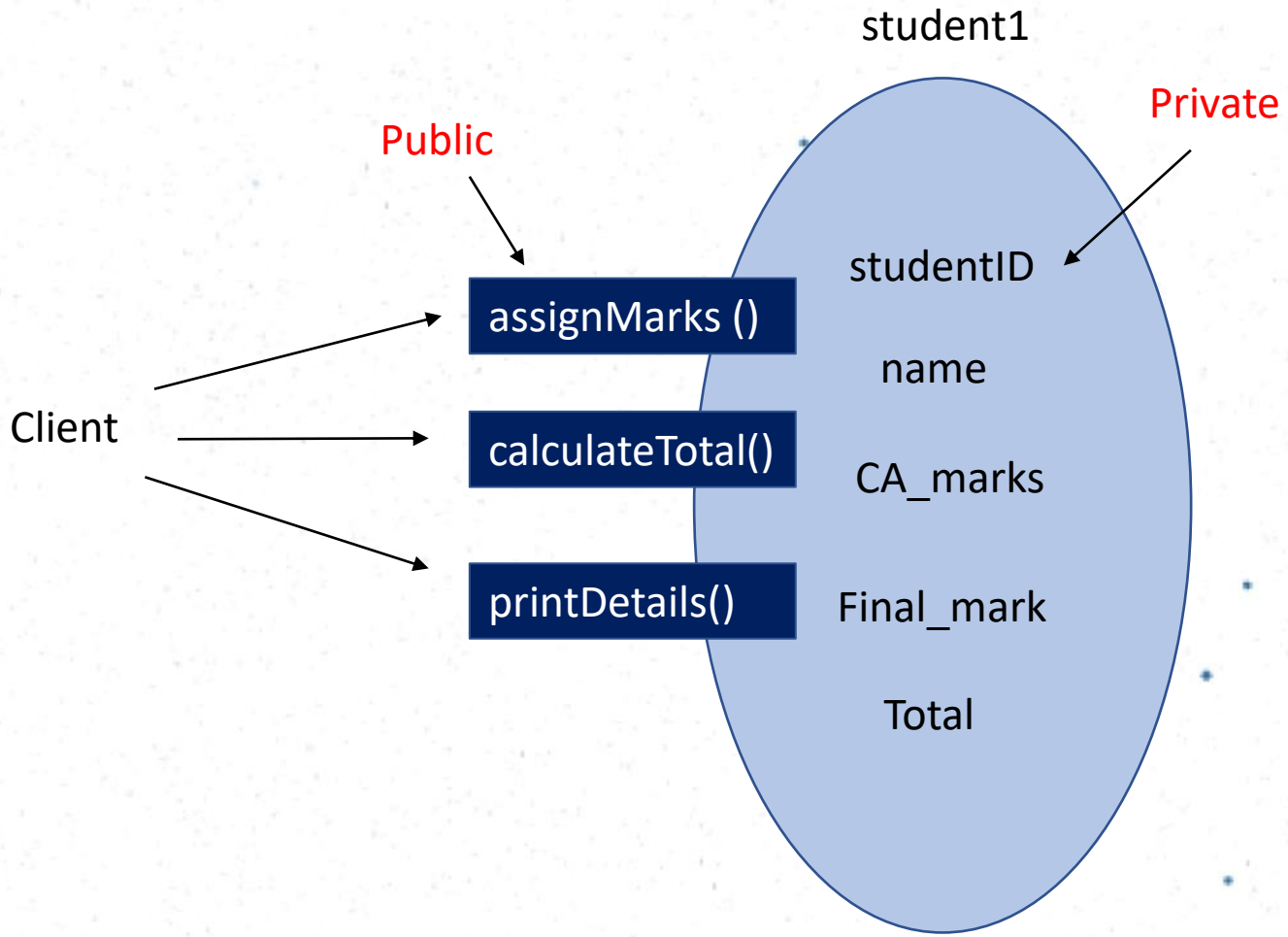
Java vs C++

- All methods are implemented in the class definition in Java.
- Each property and method require a specific access modifier (e.g. private, public, protected, friendly-default)
- In Java there is no semi colon at the end of the class
- In Java you only have dynamic objects
- Since Java has an automatic garbage collector, you do not need to use a command line delete to remove objects from memory.
- We use the “dot” operator instead of the “->” operator to access methods in Java.

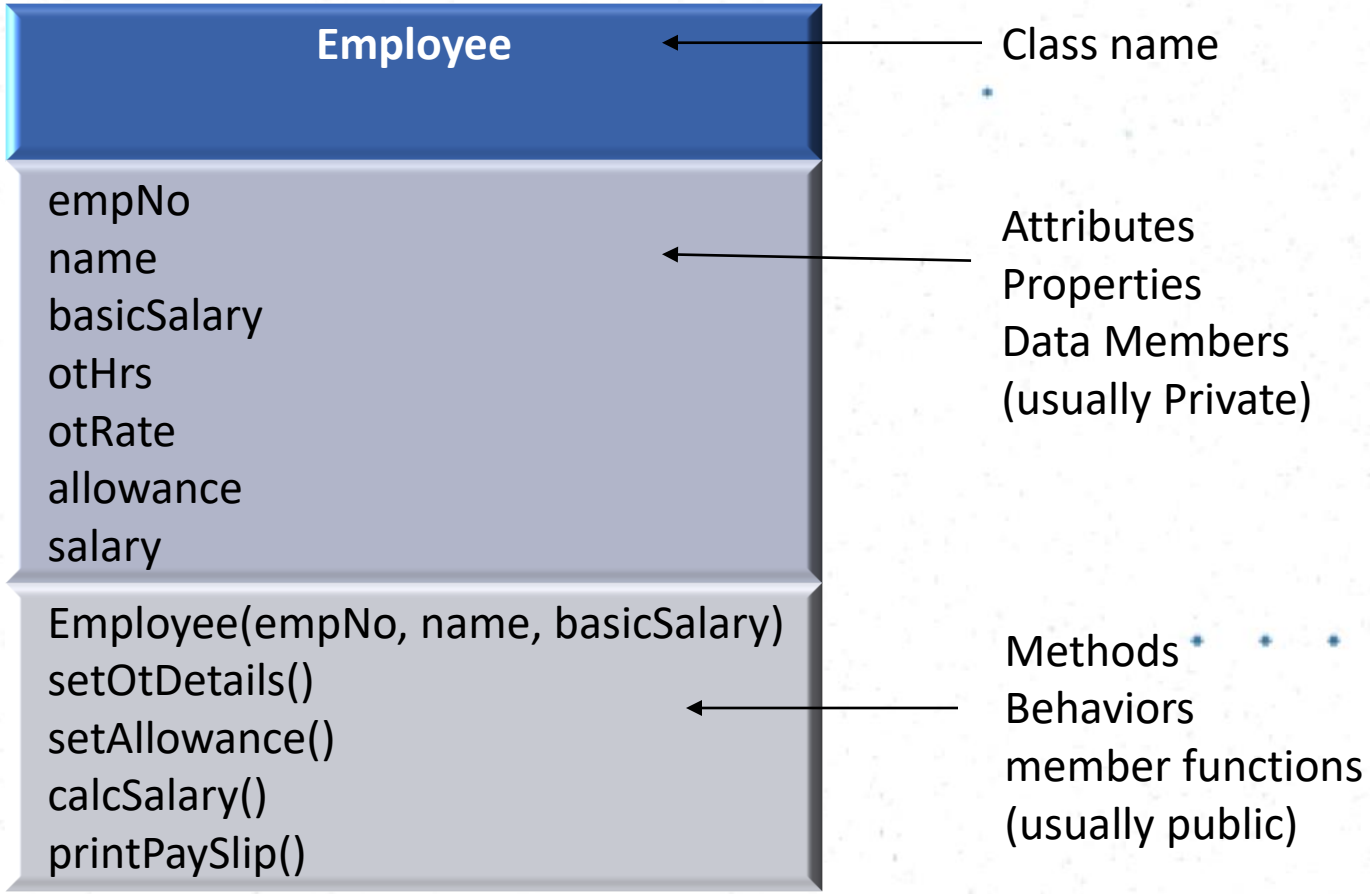
Private & Public

- The private part of the definition specifies the data members of a class
- These are hidden, not accessible outside the class and can only be accessed through the operations defined in the class
- The public part of the definition specifies the operations as function prototypes
- These operations, or methods as referred in Java, can be accessed by the main program

Private & Public



Terminology



Creating Objects

```
Student student1 = new Student();
Student student2 = new Student();
// We do not use * for pointers in Java
```

student1
studentNo – 1011 Name – Ajith Silva CA_mark - 56 Final_mark -60

student2
studentNo – I131 Name – Surani Fernando CA_mark - 70 Final_mark -65

Methods and Properties

C++	Java
Data members	Attributes (properties or variables)
Member functions	Methods (operations or behaviours)

- In C++ properties are called data members, other popular names for properties are attributes, variables.
- In C++ methods are called member functions, other popular names are operations, behaviors. These are really functions.

Abstraction

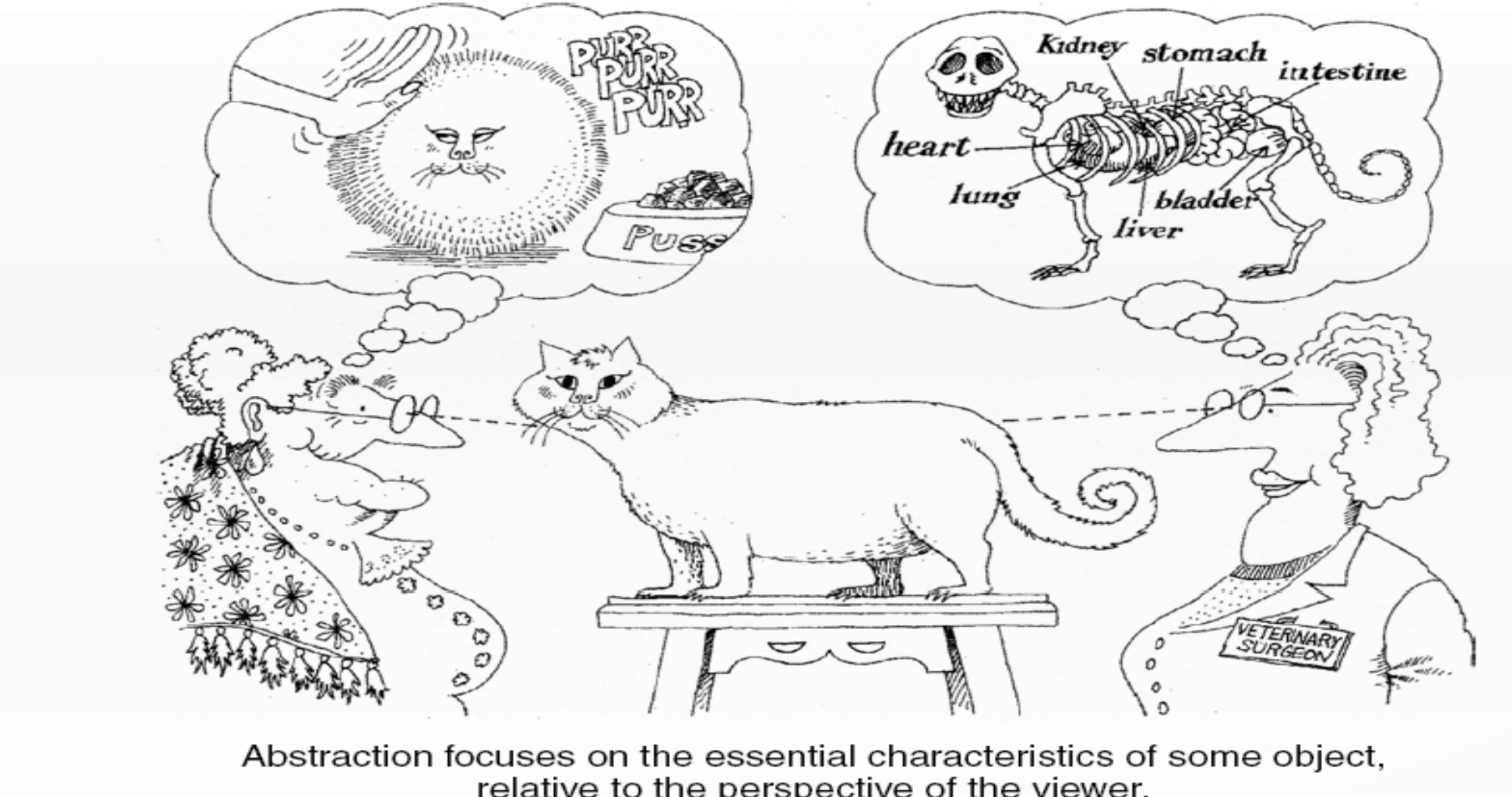
- An abstraction denotes the *essential characteristics* of an object that distinguish it from all other kinds of objects
- Thus, it provides crisply defined conceptual boundaries, relative to perspective of the viewer.

(Reference : Grady Booch, etal (2008), Object Oriented Analysis and Design with Applications 3rd Edition, pg 44)

Abstraction

- Abstraction is the process of removing characteristics from ‘something’ in order to reduce it to a ***set of essential characteristics*** that is needed for the particular system.

Abstraction



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

Example

Identify different classes that may exist in a Hospital Management System.

Can you figure out different attributes of objects that exist in this system?

What are the attributes that can be omitted?

Example contd.,

Hospital Management System

Object : Receptionist objects

Attributes: age,weight,height,bSalary,address,name, numberofChildern,staffID, allowance, telephoneNumber

What are the attributes that can be omitted?

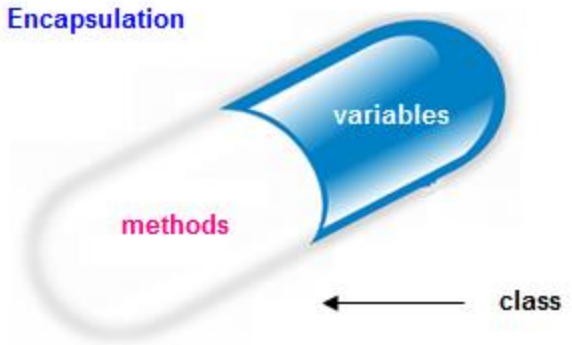
age,weight,height, numberofChildern, allowance

What are the attributes needed ?

name,staffID, address, telephoneNumber

Encapsulation

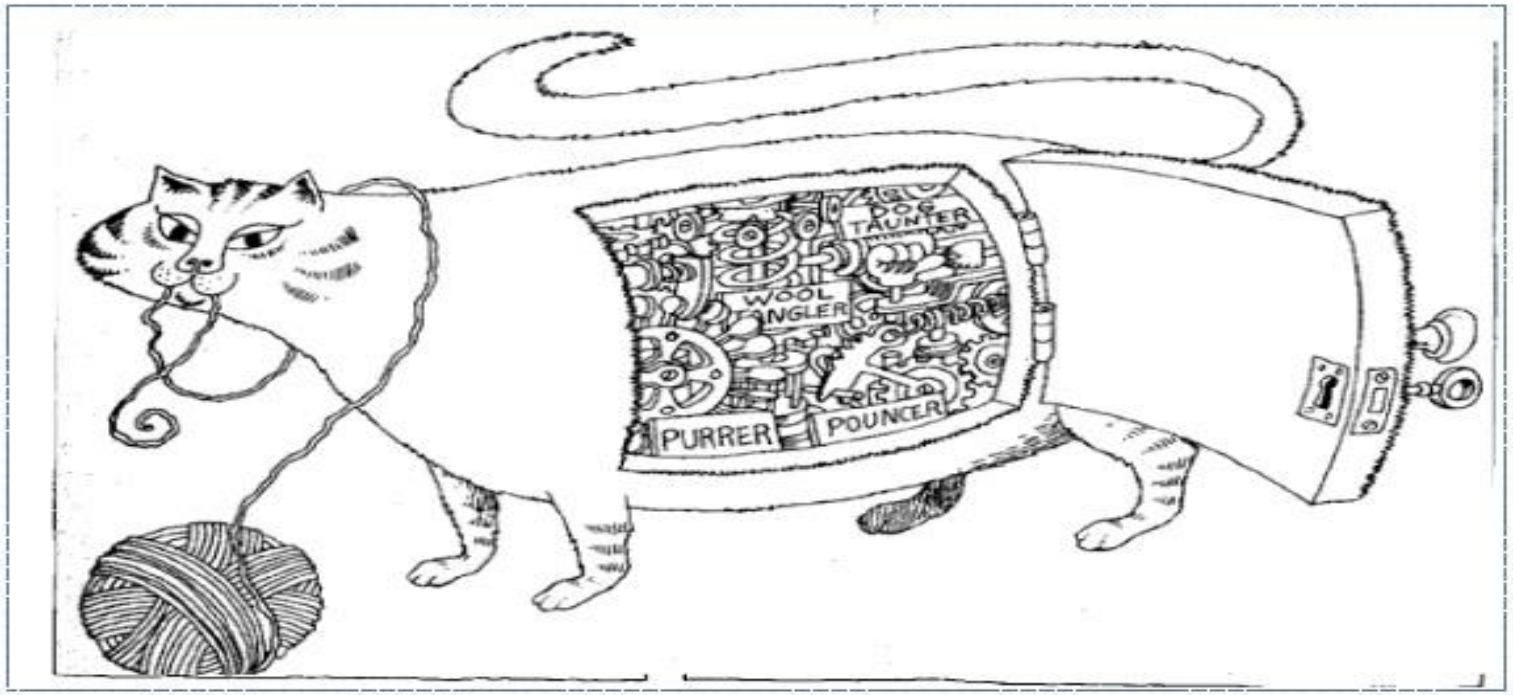
- Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior.
- Encapsulation serves to separate the contractual interface of an abstraction and its implementation.



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

Encapsulation

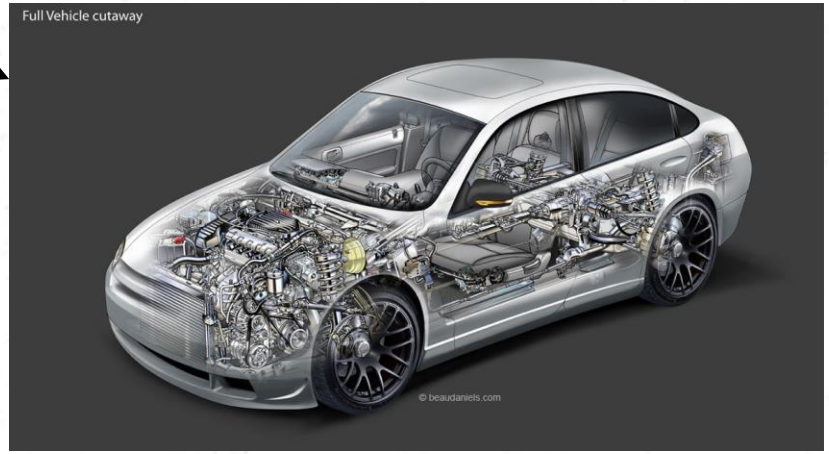
Encapsulation hides the details of the implementation of an object



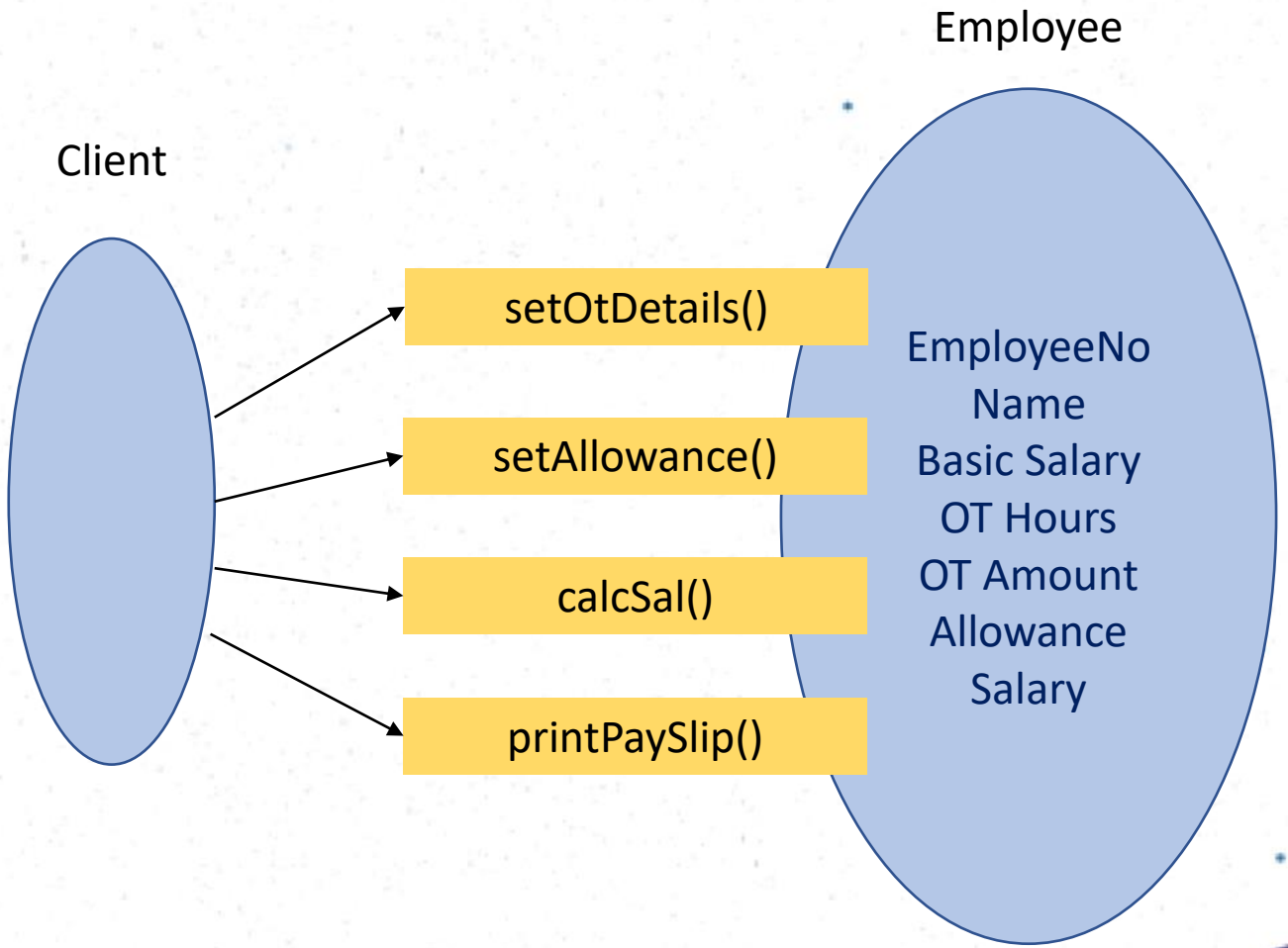
Information Hiding

- Hide certain information or implementation decision that are internal to the encapsulation structure (class)
- Objects can be made accessible through public modifiers.
  - Public – anyone can access / see it
  - Private – no one except the defining class can see/ use it
  - Java has two more modifiers as default and protected

Interface



Interface


Employee
Client
setOtDetails()
setAllowance()
calcSal()
printPaySlip()
EmployeeNo
Name
Basic Salary
OT Hours
OT Amount
Allowance
Salary

Example

- The Receptionist class define the structure and the behavior of the receptionist.
- ID, name, age define who the receptionist is.
- GenerateBill method will define what the receptionist will do.
- Such behaviors will be defined by the object which will be created from the receptionist class.
- This means the behavior of the Receptionist (or interface of a class) is defined by the methods that operate on its instance data.

Exercise 4 – Sample Answer Contd.

```
public class Recepitionist{
        private int staffID;
        private String name;
        private String telephoneNumber;


        public Boolean checkRoomAvailability(int
roomNumber){  }
        public double generateBill(){      }
        public void takeCustomerFeedback() { }
}
```

Creating objects

- Implement a class called MyMain with a main method.
- Create objects from Receptionist class to demonstrate the methods.

Example

```
class MyMain {
    public static void main(String args[]) {
        Receptionist recep1 = new Receptionist();

        boolean status = recep1.checkRoomAvailability(3);
        double bill = recep1. generateBill();
        recep1. takeCustomerFeddback();
}
```

Constructor

- Constructor is used to initialize the object when it is declared.
- Constructer is a method which has the same name as the class name.
- Constructor does not return a value, and has no return type (not even void)
- There can be default constructors with no parameters and constructors with parameters
- When an object is declared the appropriate constructor is executed.

Constructors

- Default Constructors
  - Can be used to initialize attributes to default values

```
public Rectangle () {
    width = 0;
    length = 0;
}
```

- Overloaded Constructors (Constructors with Parameters)
  - Can be used to assign values sent by the main program as arguments

```
public Rectangle (int w, int l) {
    width = w;
    length = l;
}
```

Example

- Add a default constructor to the Receptionist class
- And another parameterized constructor to initiate all attributes

Sample Answer

```
public class Receptionist{
        private int staffID;
        private String Name;
        private String TelephoneNumber;

        public Receptionist(){
                this.staffID = 0;
                this.name = “abc”;
                this. telephoneNumber = null;
        }
        public Receptionist(int pID, String pName, String pTelephoneNumber){
                this.staffID = pID
                this.name= pName ;
                this. telephoneNumber = pTelephoneNumber;
        }
        ……
}
```