SLIIT
Discover Your Future

IT2030 - Object Oriented Programming

Lecture 02

Java Classes and Objects

Learning Outcomes

At the end of the Lecture students should be able to get a revision on OOC you learnt in Year 1

- 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.

SLIIT
Discover Your Future

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, eta (2008), Object Oriented Analysis and Design with Applications 3rd Edition, pg 41)

SLIIT
Discover Your Future

Object Oriented Programming

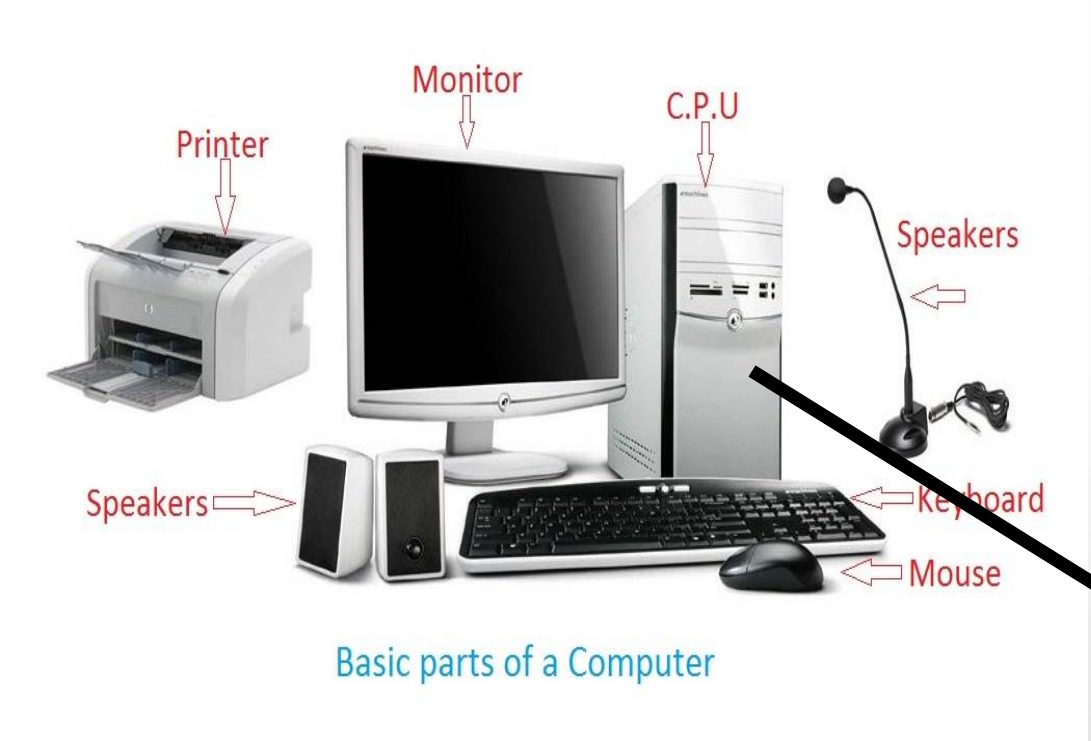
- Object Oriented Programming is a method of implementation in which programs are organized as a collection of objects which cooperate to solve a problem.
- Allows to solve more complex problems easily

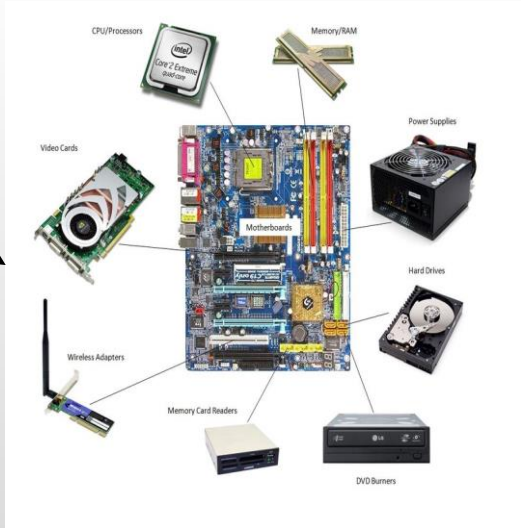
SLIIT
Discover Your Future

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

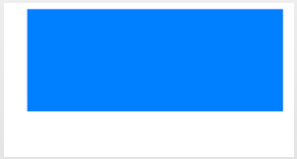
A Computer System

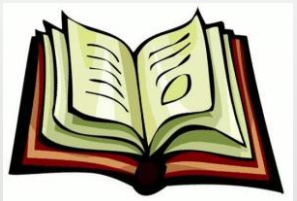




Classes

- 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.

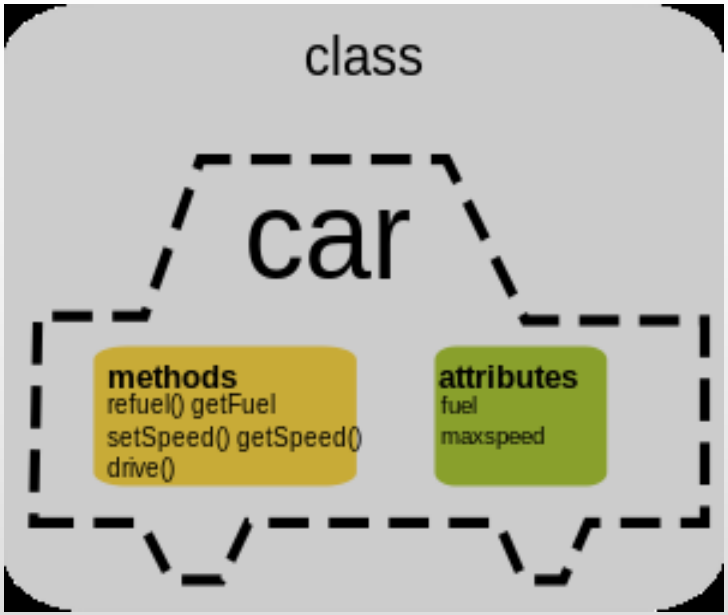



Classes

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

Class Name
Attributes/Properties
Methods

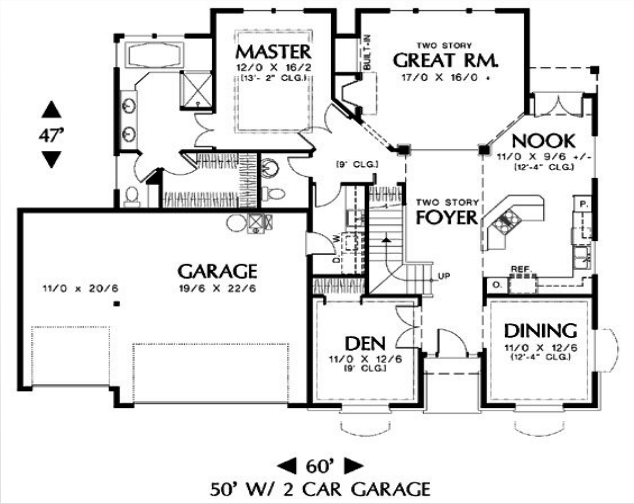
e.g.:



SLIIT
Discover Your Future

Classes and Objects

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



Class House

House1

House2

House3

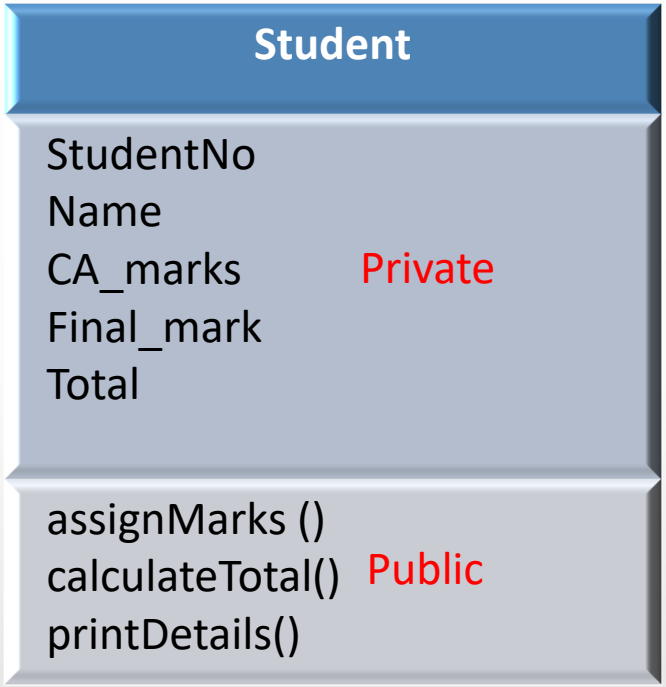
Objects

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 and then create as many objects which has the same structure of the class (House example)

SLIIT
Discover Your Future

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();
};
```

SLIIT
Discover Your Future

Exercise 1

- Identify the classes, attributes and what sort of objects that will have for hospital management system

- Write the classes and possible methods in simple English language

SLIIT
Discover Your Future

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++

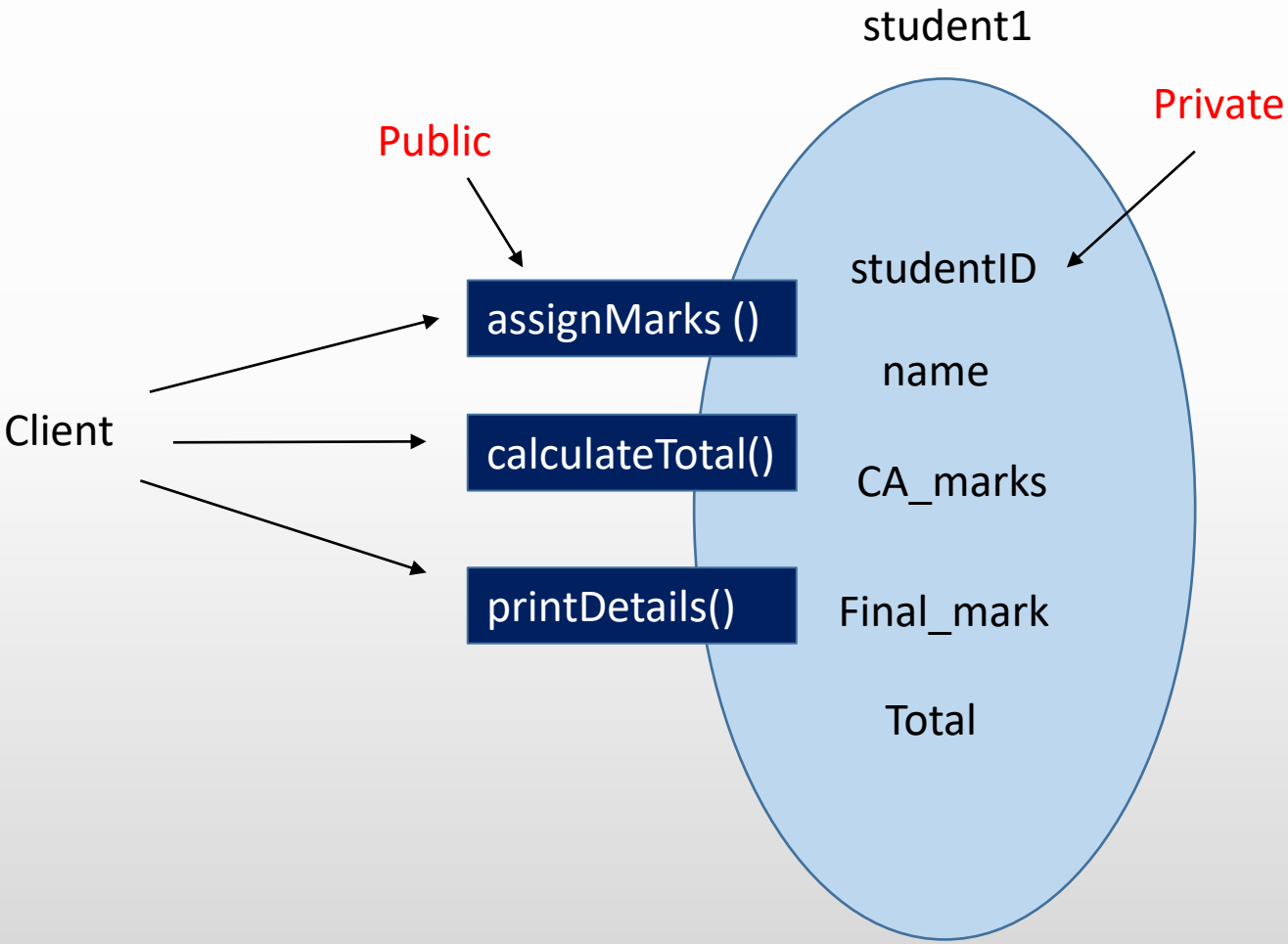
- All methods are implemented in the class definition in Java.
- Each property, method needs a specific access modifier e.g. private, public, protected
- In Java there is no semi colon at the end of the class
- In Java you only have dynamic objects.
- Since Java has an automatic garbage collector, you 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 from outside the class and can only be accessed though the operations defined for the class
- The public part of the definition specifies the operations as function prototypes
- These operations, or methods as they are called, can be accesses by the main program.

SLIIT
Discover Your Future

Private & Public



SLIIT
Discover Your Future

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

Exercise 2

- Select one class that selected from the hospital management system. Ex:- receptionist
- Write the relevant class using the java language

Methods and Properties

- 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

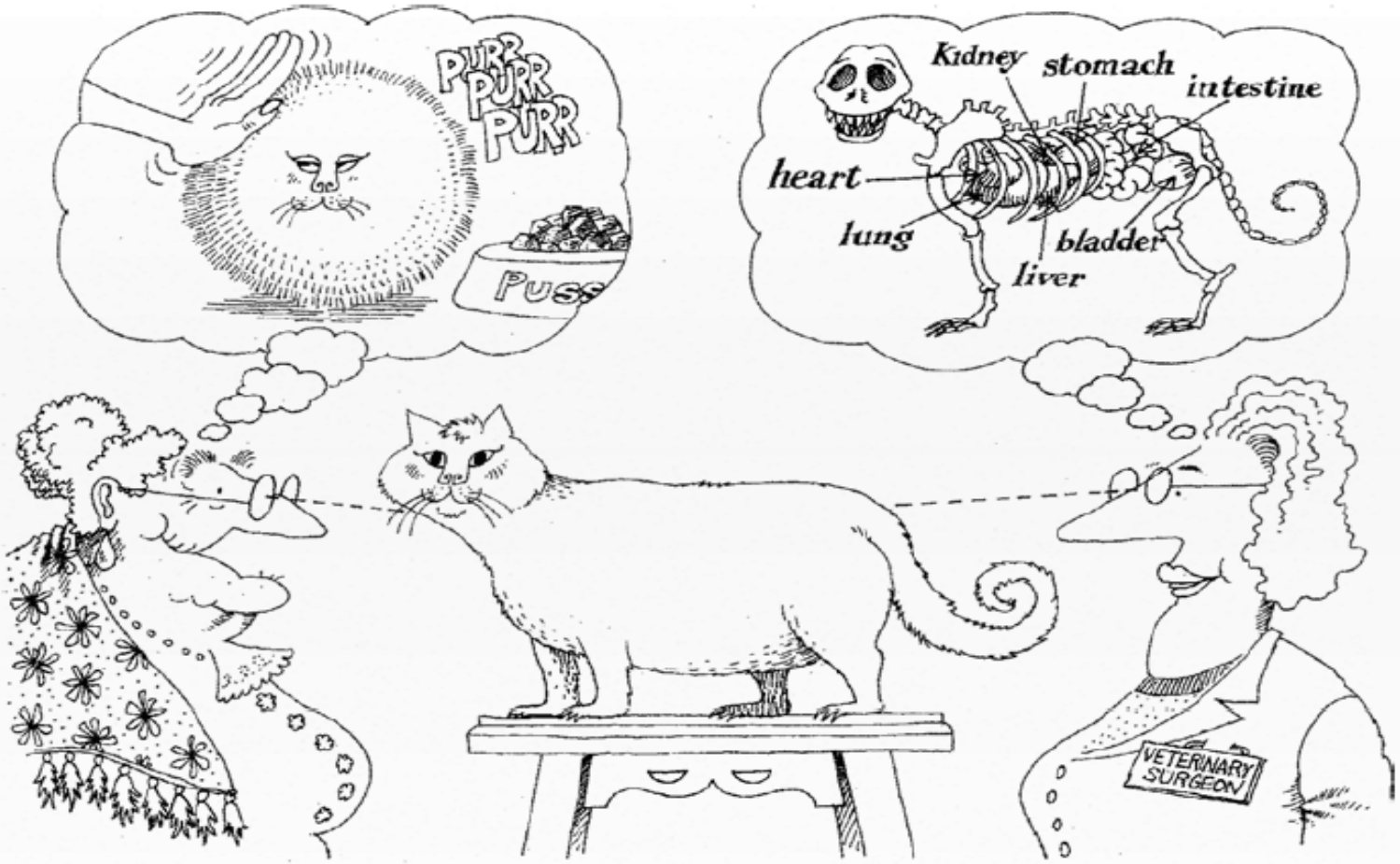
- An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to perspective of the viewer.

- (Reference : Grady Booch, eta (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.

SLIIT
Discover Your Future

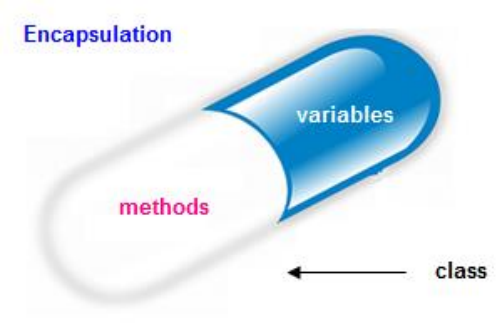
Exercise 3

Think of the class you selected in Hospital Management System.

1. What are the attributes needed ?
2. What are the attributes that can be omitted?

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.

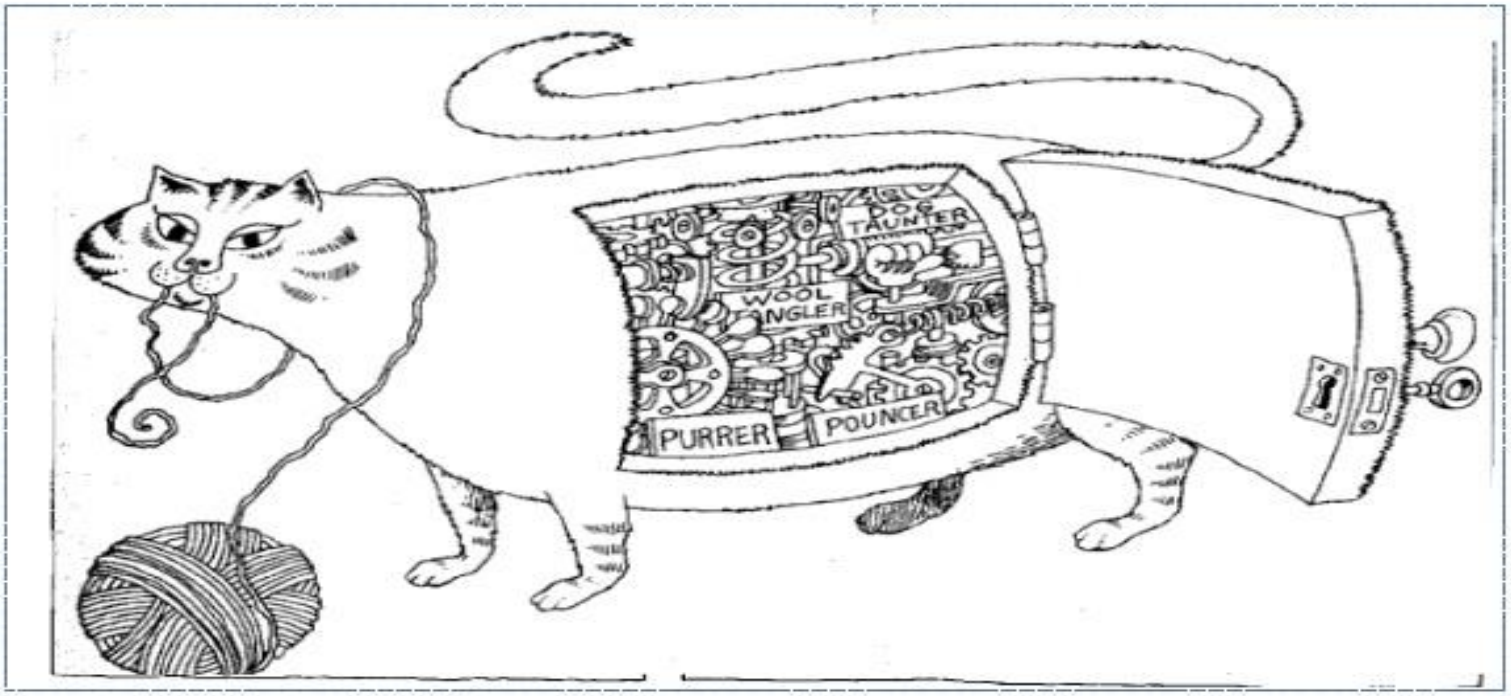


- (Reference : Grady Booch, eta (2008), Object Oriented Analysis and Design with Applications 3rd Edition, pg 52)

SLIIT
Discover Your Future

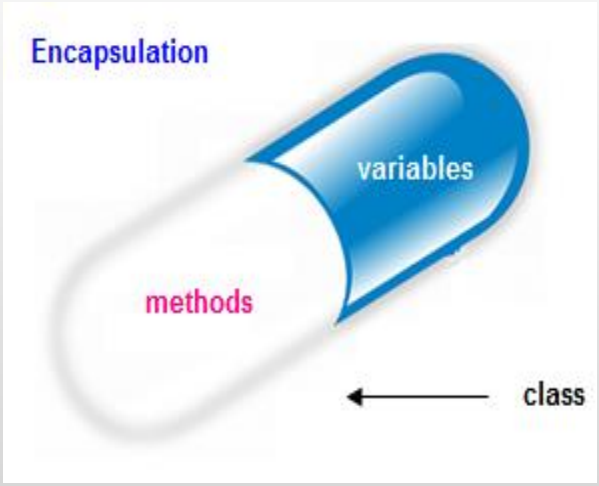
Encapsulation

Encapsulation hides the details of the implementation of an object



Encapsulation

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



SLIIT
Discover Your Future

Exercise 4

- Explain what is encapsulation with related to your class in the Hospital Management System.

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 – anyone can access / see it
  - Private – no one except the class can see/ use it

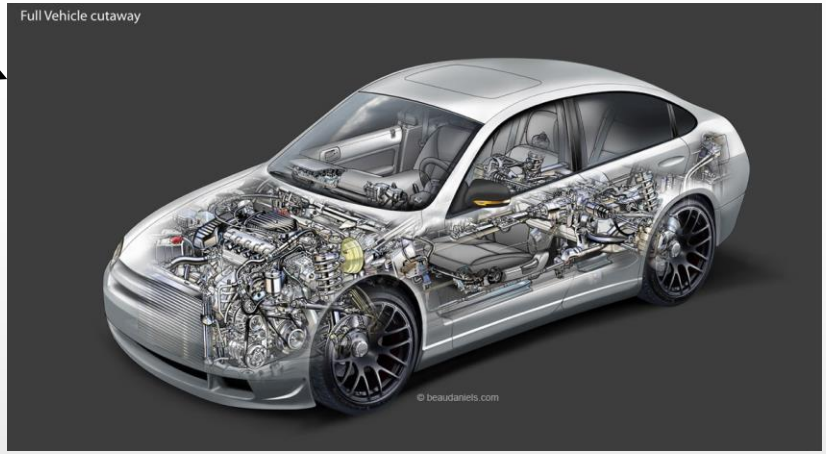
SLIIT
Discover Your Future

Exercise 5

- Modify your java classes so that it support for information hiding

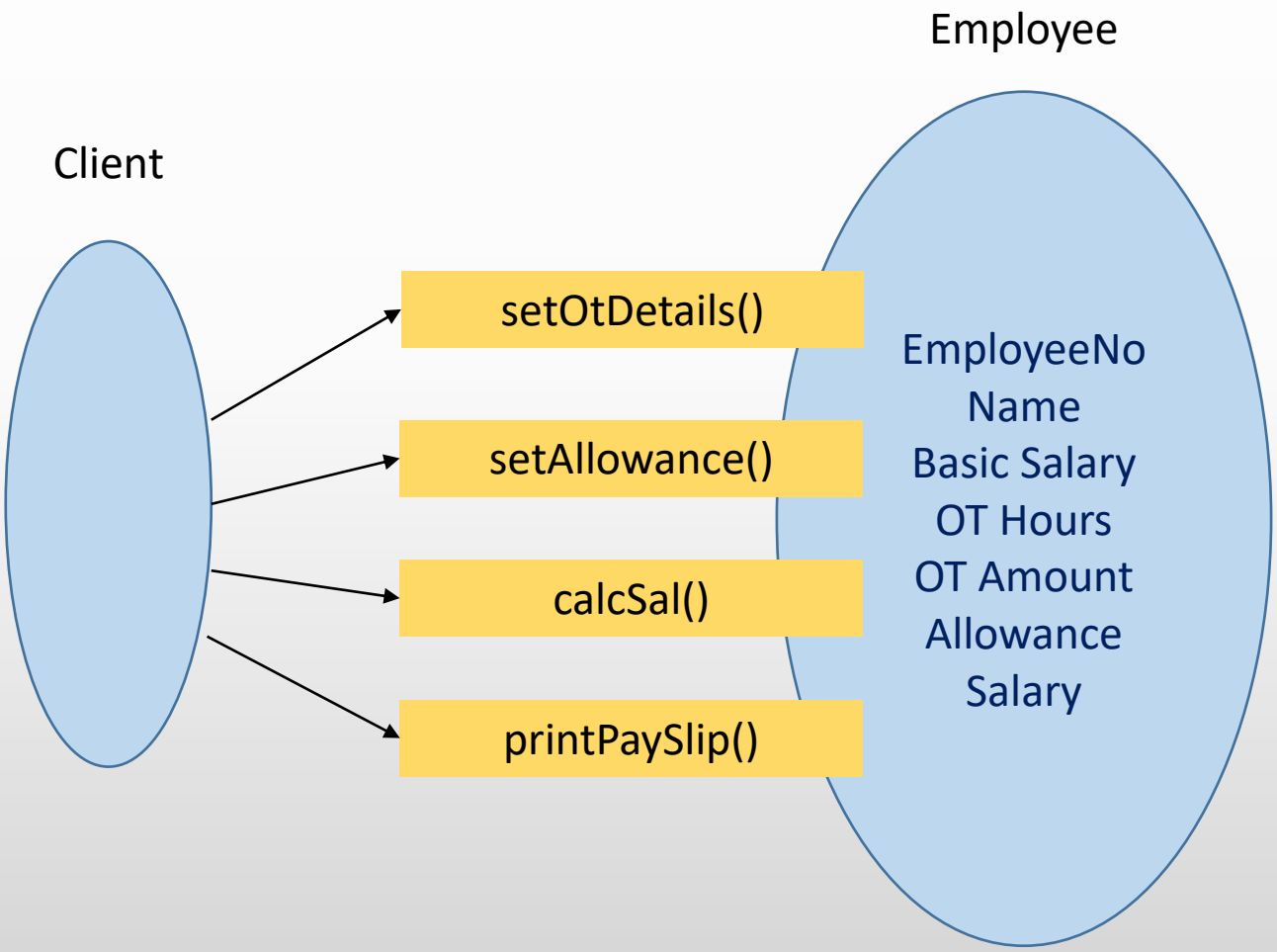
SLIIT
Discover Your Future

Interface



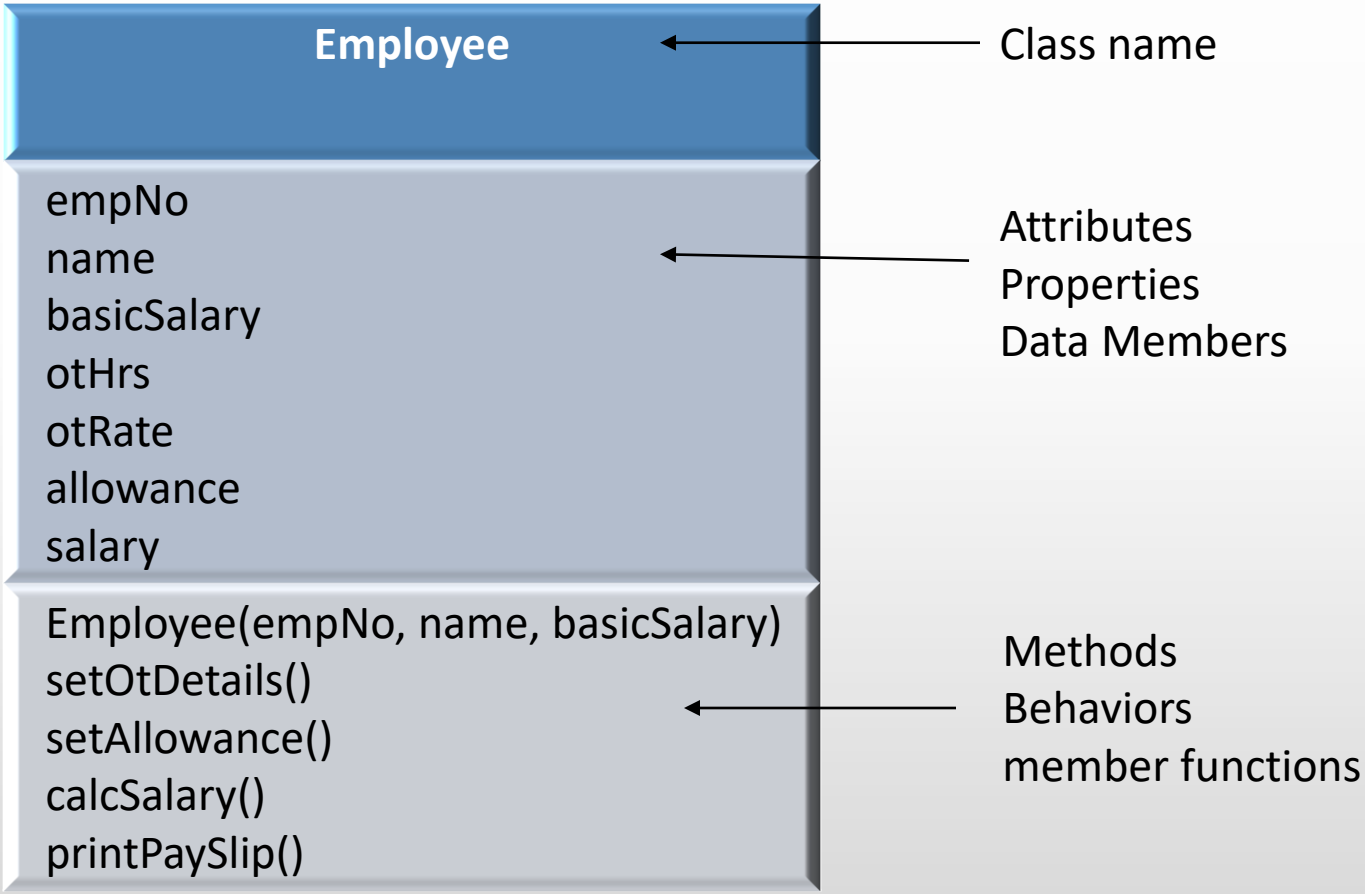
SLIIT
Discover Your Future

Interface



SLIIT
Discover Your Future

Terminology



Exercise 6

Implement a class called MyMain with a main method and Create Objects of the class to demonstrate the methods.

Constructor

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

SLIIT
Discover Your Future

Constructor

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

// Constructor with parameters
public Rectangle (int w, int l) {
    width = w;
    length = l;
}
```

```
public static void main(String args[]) {
   // default constructor called
    Rectangle r1 = new Rectangle();
     // Second constructor called
    Rectangle r2 = new Rectangle(10,20);
}
```

Constructors

- Default Constructors
  - Can be used to initialized 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;
}
```

SLIIT
Discover Your Future

Exercise 7

Take one class from your topic

1. Add the default constructor
2. And a parameterized constructor to initiate all attributes

Getters and Setters

- In general properties are declared as private preventing them from being accessed from outside the class.
- Typically an attribute will have a getter (accessor) (A get method to return its value) and a setter (mutator) (A set method to set a value).
- e.g. a property called length will have a getter defined as int getLength() and a setter defined as void setLength(). Both these methods will be declared as public methods.

Getters and Setters

```
4       private int width;
5       private int length;
6
7       // Constructors
8       public Rectangle() {…
11      }
12      public Rectangle(int w, int l) {…
15      }
16      // public
17      public void setWidth(int w) {
18          if (w > 0)
19              width = w;
20          else
21              width = 10;
22      }
23      public int getWidth() {
24          return width;
25      }
```

this keyword

- The this **keyword** can be used to refer to any member of the current object from within an instance Method or a constructor.

```
public Employee(int pempno, String name, double pbasicSal) {
    employeeNo = pempno;
    this.name = name;
    basicSalary = pbasicSal;
}
```

- We need to use this.name to refer to the property name to distinguish it from the parameter name.

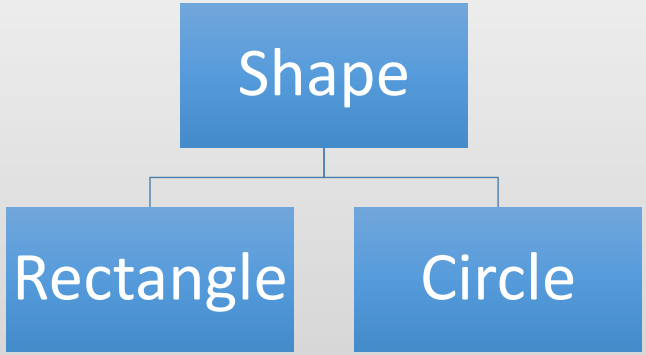
SLIIT
Discover Your Future

Exercise 8

- Modify your class with getters and setters

Generalization/Inheritance

- Child class **is a** type of the parent class
- Used to showcase reusable elements in the class diagram
- Child classes “inherit” the attributes and methods defined in the parent class



SLIIT
Discover Your Future

C++ vs Java - Inheritance

C++

```
class Circle : public Shape {
```

Java

```
class Circle extends Shape {
```

Java has a simpler inheritance mechanism where base class is extended as public.

C++ has multiple inheritance compared to Java's Single Inheritance.

C++ vs Java - Inheritance

C++

```
Circle (string tname, int r) : Shape ( tname) {
        radius = r;
}
```

Java

```
public Circle (String tname, int r) {
    super(tname);
    radius = r;
}
```

When you want to call a base class constructor C++ Requires to explicitly name the base class. In Java we use the super keyword to access the direct descendent class.

However this implies that in Java you can't directly call a class higher in the hierarchy e.g. the Grandfather class which is not in C++

C++ vs Java - Inheritance

C++

```
virtual void speak() {}
```

Java

```
public void speak() {}
```

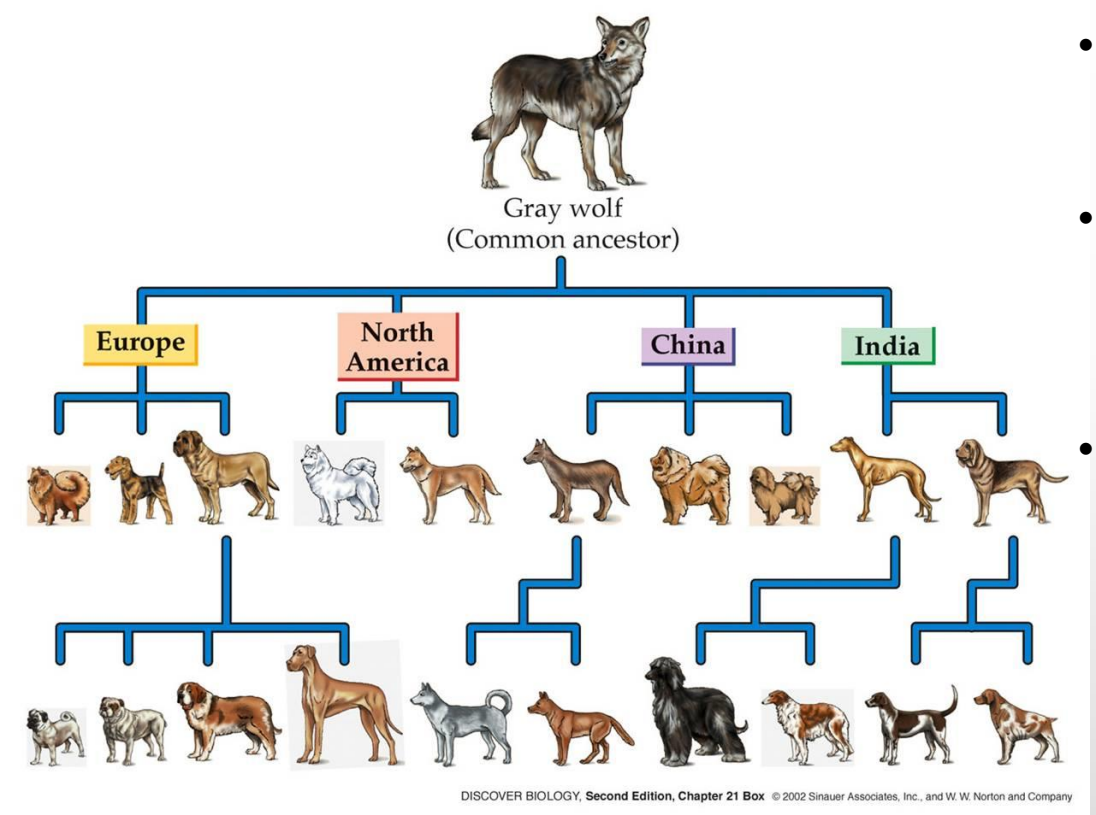
All methods in Java are virtual by default. In C++ we need to explicitly define polymorphic methods.

SLIIT
Discover Your Future

Exercise 9

Think of a way of implementing generalization in you selected class

Object Class



- A class hierarchy is similar to the taxonomy of animals that shows their ancestry.
- There are many breeds of dogs. A well known fact is that the common origin of a dog is the Gray Wolf.
- All Java classes are derived from a class called Object. This includes all the existing Java built in classes and the classes that you write. The methods and properties of the Object class is accessible to any Java class that you create.

Inheritance Shapes Example C++

```
23    class Rectangle: public Shape{
24            protected:
25            int width;
26            int height;
27            public :
28            Rectangle (string tname, int w, int h) : Shape ( tname)
29            {
30                    width = w;
31                    height = h;
32                }
33            int area( )
```

Shape_example.cpp

SLIIT
Discover Your Future

Inheritance Shapes Example Java

```
1  class Shape {
2      protected String name;
3      public Shape() {};
4      public Shape (String tname) {…
6      }
7      public void print() {…
9      }
10     public int area(){ return 0;}
11 }
12 class Rectangle extends Shape {…
32 }
33 class Circle extends Shape {…
47 }
48 class ShapeApp {
49     public static void main(String args[]) {
50     Rectangle R = new Rectangle("Rectangle", 4 , 6);
51     Circle C = new Circle("Circle", 3 );
```

Shape_example.java

SLIIT
Discover Your Future

Exercise 10

Write the java code in order to implement the inheritance

Polymorphism

- Greek meaning *"having multiple forms"*
- Ability to assign a different meaning or usage to something in different contexts
- Specifically, to allow an entity such as a variable, a function, or an object to have more than one form

An example

- Consider the request (analogues to a method) *"please cut this in half"* taking many forms

SLIIT
Discover Your Future

"please cut this in half"

- Imagine this task is automated…
- Without polymorphism
  - Need to tell the computer how to proceed for each situation
- With polymorphism
  - Just tell *please cut this in half*
  - The computer will handle the rest!

SLIIT
Discover Your Future

Animal Example

```
 2  class Animal {…
20  }
21  class Cat extends  Animal {…
30  }
31  class Dog extends Animal {…
40  }
41  class Cow extends Animal {…
50  }
51  class AnimalApp {
52      public static void main(String args[]) {
53      Animal ani[] = new Animal[4];
54      ani[0] = new Cat("Micky the Cat");
55      ani[1] = new Dog("Rover the Dog");
56      ani[2] = new Cow("roo the Cow");
57      ani[3] = new Animal("no name");
58      for (int r=0;r<4; r++)
59          ani[r].song();
```

Animal_example.cpp

Animal_example.java

SLIIT
Discover Your Future

Exercise 11

- Think of a way to implement Polymorphism in your solution

Exercise 12

Implement the necessary classes, along with their required attributes and methods, based on the following description.

You have been assigned the task of developing a simple system to manage the employees of a company. The system should consist of two classes, Employee and Manager. The Employee class should have three data members, EmpId, name, and address, which are all of string data type. It should also have a default constructor and a parameterize constructor to initialize these attributes.

Create a getter and a setter to the attribute EmpId. Add two methods, Read () and Print(). The Read () should accept inputs for the above mentioned attributes through the keyboard and Print() should display their values.

The Manager class should extend from the Employee class and should have two data members Department number (int), Department name (String). It should have a parameterized constructor to initialize all these attributes.

Furthermore, you need to create a class called EmployeeApp with a main method where you can create 2 objects each from Employee and Manager classes.

Thank you!