IT2205: Programming I

Section 5

Object Orientation (13 hrs)

Understanding Object-Orientation

- Object Orientation has taken the software world by storm.
- As a way of creating programs it has number of advantages.

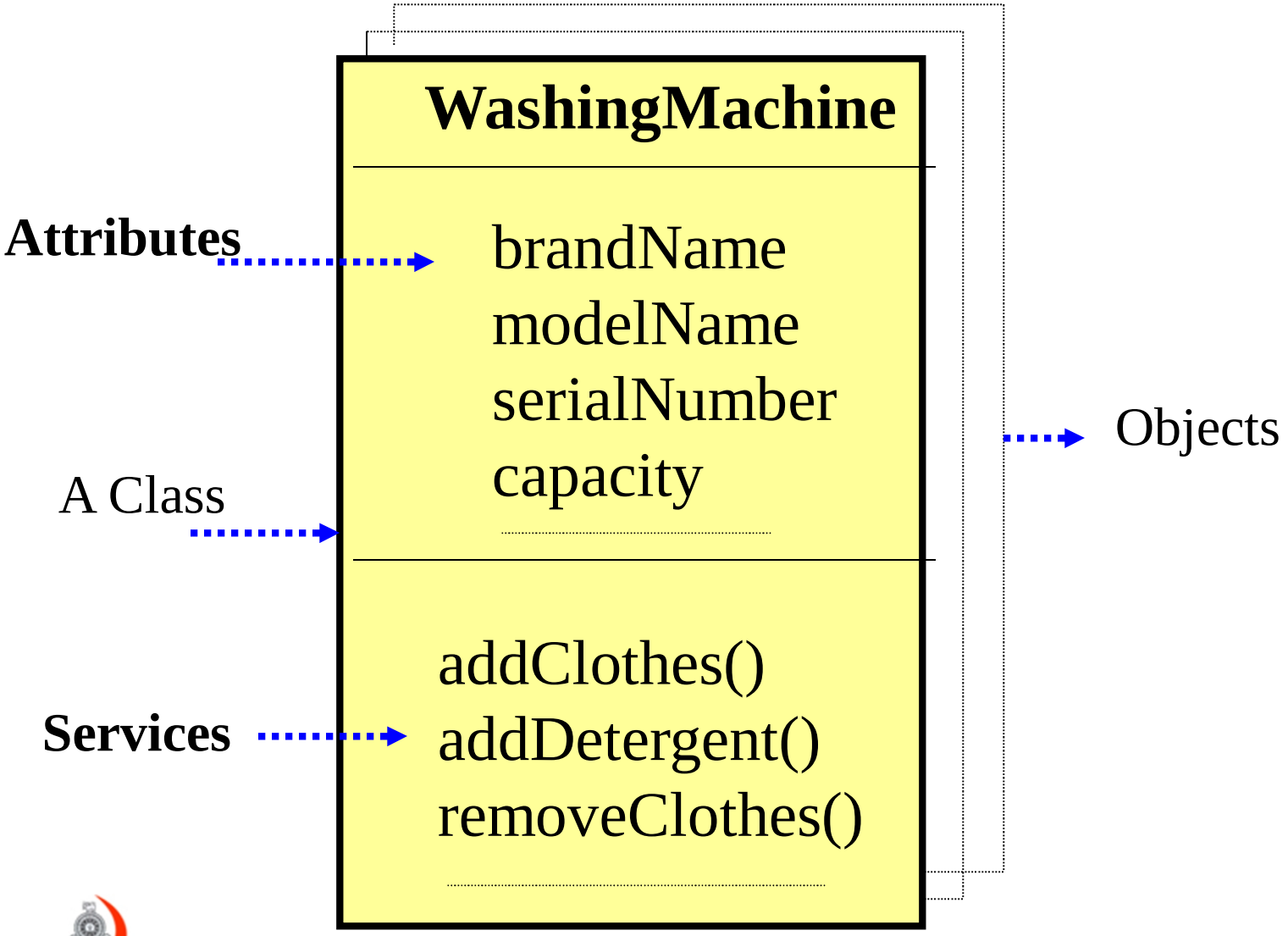
Understanding Object-Orientation cont..

- It promotes the component- based approach to software development.
  - First create a system by creating a set of objects.
  - Then, you can expand the system by adding capabilities to components you have already built *or* by adding new components.
  - Finally, you can reuse the objects you created for the system when you built a new system. This will substantially reduce the system development time.

Understanding Object-Orientation cont..

- UML (Unified Modeling Language) plays an important role in Object Orientation.
- UML allows you to built easy to use and easy to understand models of objects so that programmers can easily write software.
- Object Orientation depends on a few fundamental principles.(see later)

Attributes and Services



Attributes and Services Contin...

- A principle used to derive *attributes* and *services*

  * What the specific concept knows ?
    - *attributes*
  * What are the responsibilities of a class ?
    - *services*

Services operate when they are involved through message connections.

An object

Knows how to draw a rectangle

but he will not draw unless somebody ask him to draw

UCSC

BIT

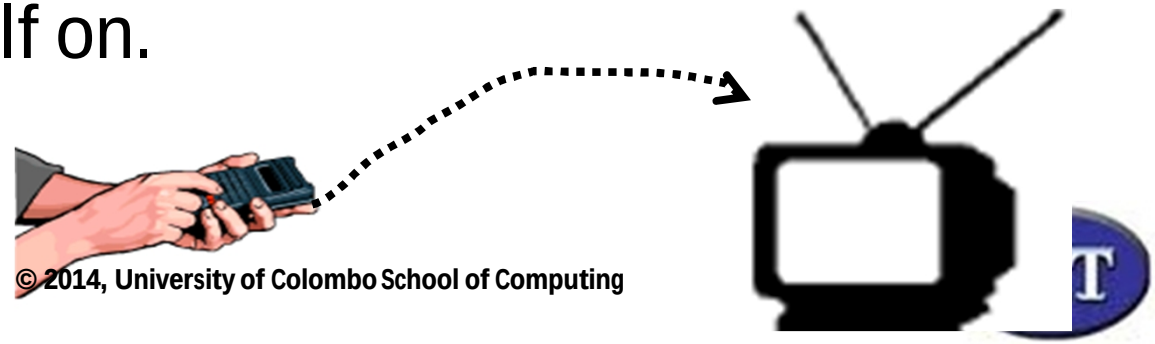
Message Interactions

- In a system, objects work together.
- They do this by sending messages to one another.
- One object sends another a message to perform an operation.
- The receiving object performs that operation.

  eg. A TV and a Remote

Message Interactions cont...

- When you want to watch a TV show,
  - You hunt round for a remote,
  - Settle into your favorite chair and
  - Push the *On* Button.
- What happens?
  - The remote object sends a message (literally!) to the TV object to turn itself on.
  - The TV object receives the message.
  - It knows how to perform the turn-on operation, and turns itself on.



Message Interactions cont.

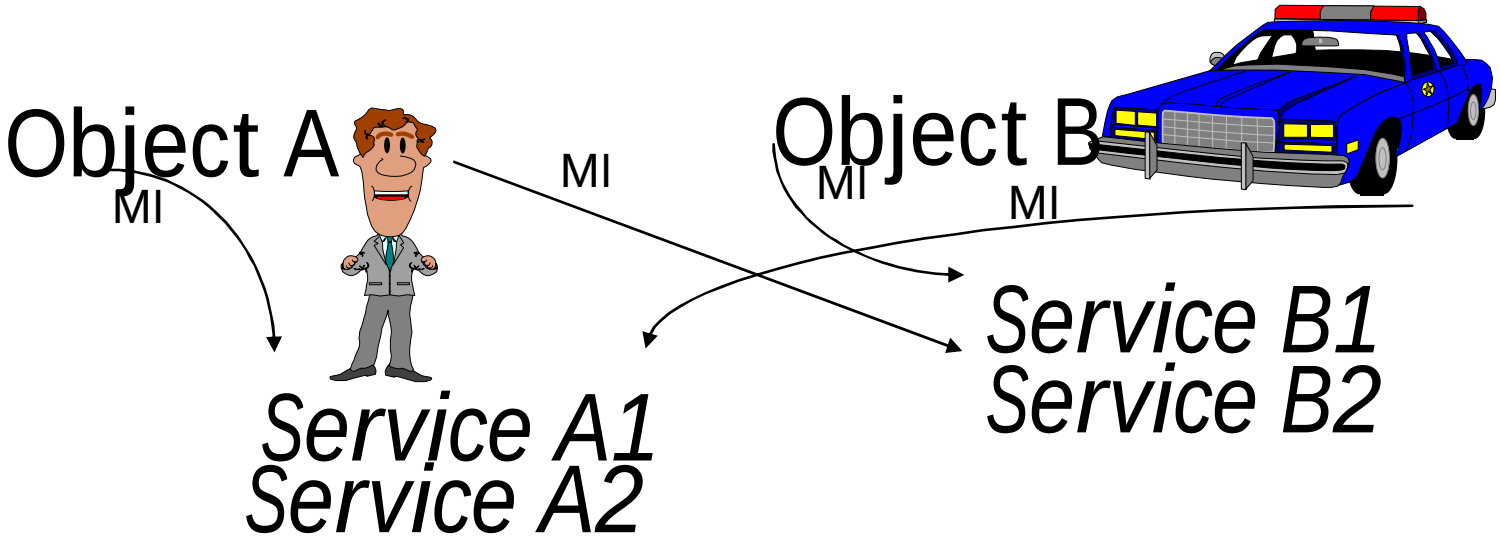
- When you want to watch a different channel,
  - You click the appropriate button on the remote,
  - Remote object sends a different message *change channel* to the TV object.
- The remote can also communicate with the TV via other messages for *changing the volume, muting the volume etc.*.

Message Interactions cont..

- Let's go back to interfaces for a moment.
- Most of the things you do from the remote, you can also do by *getting out of the chair*, *going to the TV*, and *clicking buttons on the TV.*
- The interface the TV presents to you is obviously not the same interface it presents to the remote.

Message Interactions (MI) cont..

- Each object offers services within the realm of its responsibility.



* Objects use other objects services through *message Interactions*

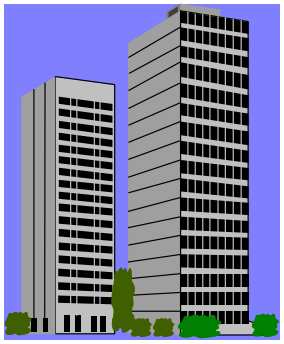
* An object may even call its own services

Relationships

- Three types used in OOA

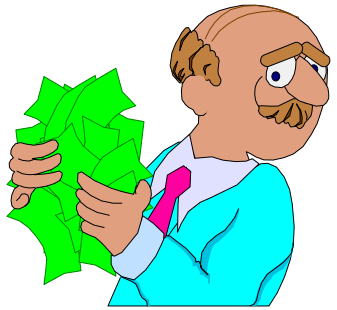
Instance connection (association), Whole parts (aggregation), Inheritance (Generalisation - Specialisation)

Association



Association

1 1..*



Malik
Sunil

Person

Organization *IBM*

Association represent a ***'uses a'*** relationship

Association cont...

- When you turn on your TV, in object oriented terms, you are in an *association* with your TV.
- An association is unidirectional (one way) or bi-directional (two way).

  eg. *is married to*
- Some times an object might be associated with another in more than one way.

  Gihan *is a co-worker of* Damith

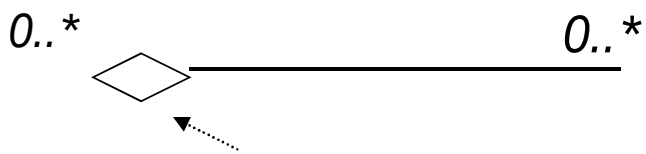
  Gihan *is a friend of* Damith

Relationships - Cont...

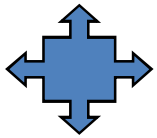
- ***Aggregation***

Represent *Whole Part* or Composition Relationship.

Aircraft



UML notation



Engine

School

Student

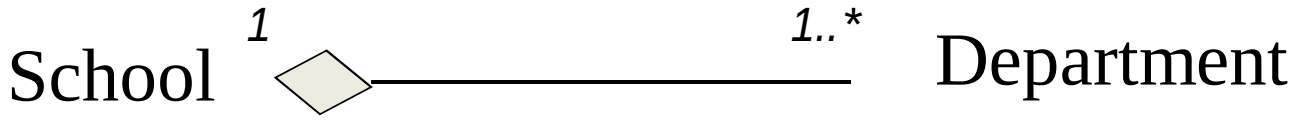
Whole object

Part Objects

Represent **'has a'** relationships.

Aggregation cont..

- ***Composition*** – is a variation of simple aggregation. It is a strong type of aggregation.



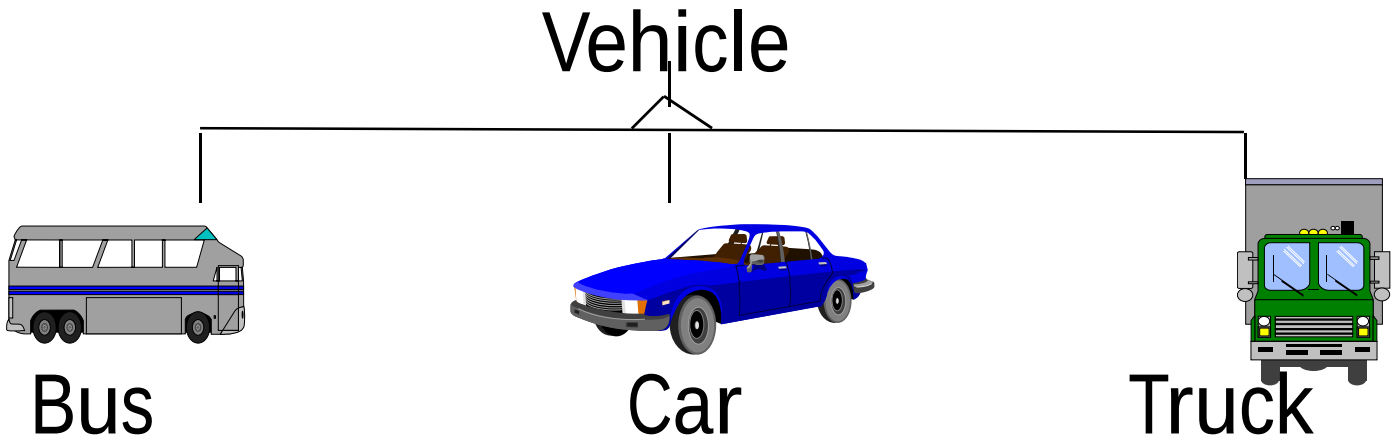
Each component in a composite can belong to just one whole.

Components will live and die with the whole object

Relationships - Cont...

- **Generalization and Specialization (Inheritance)**

Is an attempt to abstract the common features and place in a *parent class*



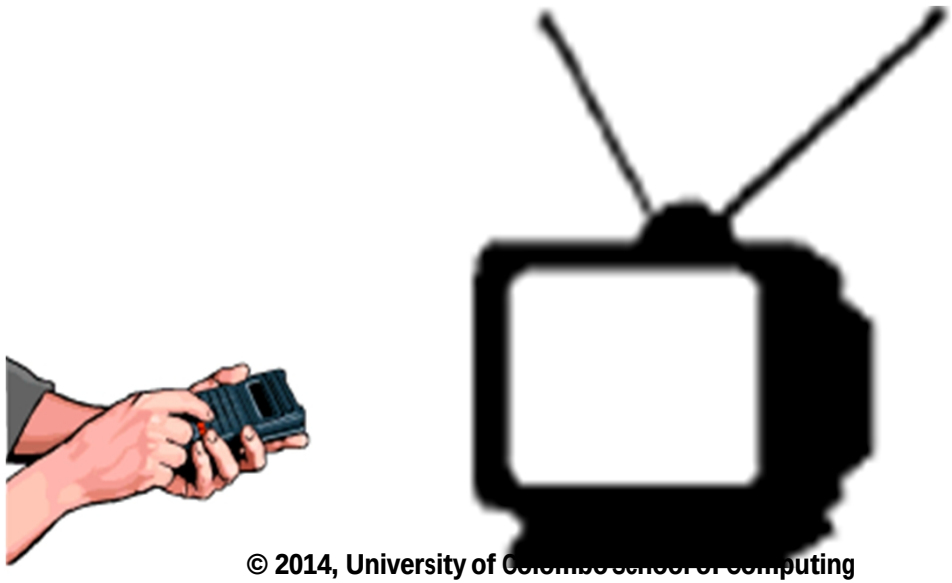
* Specialised classes inherits from the parent class
* Highest level of hierarchy is the *base* class
* Represent '**is a**' relationships

Encapsulation

- When an object carries out its operations, those operations are hidden.

Eg. When most people watch a television show,

- they usually don't know or care about the complex electronics that sit in back of the TV screen
- or the operations that are happening.



The TV hides its operations from the person watching it.

Encapsulation cont..

- Objects *encapsulates* what they do.
- That is, they hide the inner workings of their operations
  - from the outside world
  - and from other objects

Encapsulation cont..

- *Why is this important?*
  - In the software world, encapsulation helps cut down on the potential for bad things to happen.
  - In a system that consists of objects, the object depends on each other in various ways.
  - If one of them happen to malfunction, software engineers have to change it in some way.
  - Hiding its operations from other objects means it probably won't be necessary to change those other objects.

Encapsulation cont..

- Turning from software to reality, you see the importance of *encapsulation*.

eg. Your computer monitor, hides its operations from your computers CPU.

When something goes wrong with your monitor, you either fix or replace it.

You probably won't have to fix or replace the CPU along with it.

Encapsulation cont..

- Encapsulation is also called *information hiding*. (An object hides what it does from other objects and from outside world)
- But an object does have to present a "face" to the outside world, so you can initiate those operations.

eg. A TV, has a set of buttons either on the TV or on a remote.

The TV's buttons are called *interfaces*.

Encapsulation cont..

- *Traditional Programming*
  - Main program is the focus.
  - It brings in data and perform processing.
- *OO Pogramming*
  - Each object contains
    . Information it is responsible for
    . Services it supports

Encapsulation - cont...

- Encapsulation is done through the definition of *region of access*.
- *Region of access* defines the accessibility of the *services* or *attributes* in a class.
- In C++ (OO programming Language)
  - 3 types of access regions
    private, public, protected
- *In Java*
  Access modifiers: default, public, private, protected
  Static modifier, final modifier, synchronized modifier, native modifier

Polymorphism

- It allows different forms of the same service to be defined.
- Sometimes an operation has the same name in different classes.

  eg. You can open a door, you can open a window, a bank account or a conversation.

  In each case, you are performing a different operation.

Polymorphism

- In object-orientation, each class "knows" how that operation is supposed to take place.
- This is polymorphism.

Related terms used in OO programming language
Function Overloading,
Operator Overloading,
Method Overriding

Object Oriented Programming

Object Oriented Programming Introduction

- Three basic concepts underlining Object Oriented Programming
  - Data Abstraction with Encapsulation
  - Inheritance
  - Polymorphism

Object Oriented Programming Introduction cont..

- Data Abstraction in programming - process of defining a data type, often called Abstract Data Type
- **Encapsulation** - Enables you to hide, inside the object, both the *data* fields and the *methods* that act on that data.

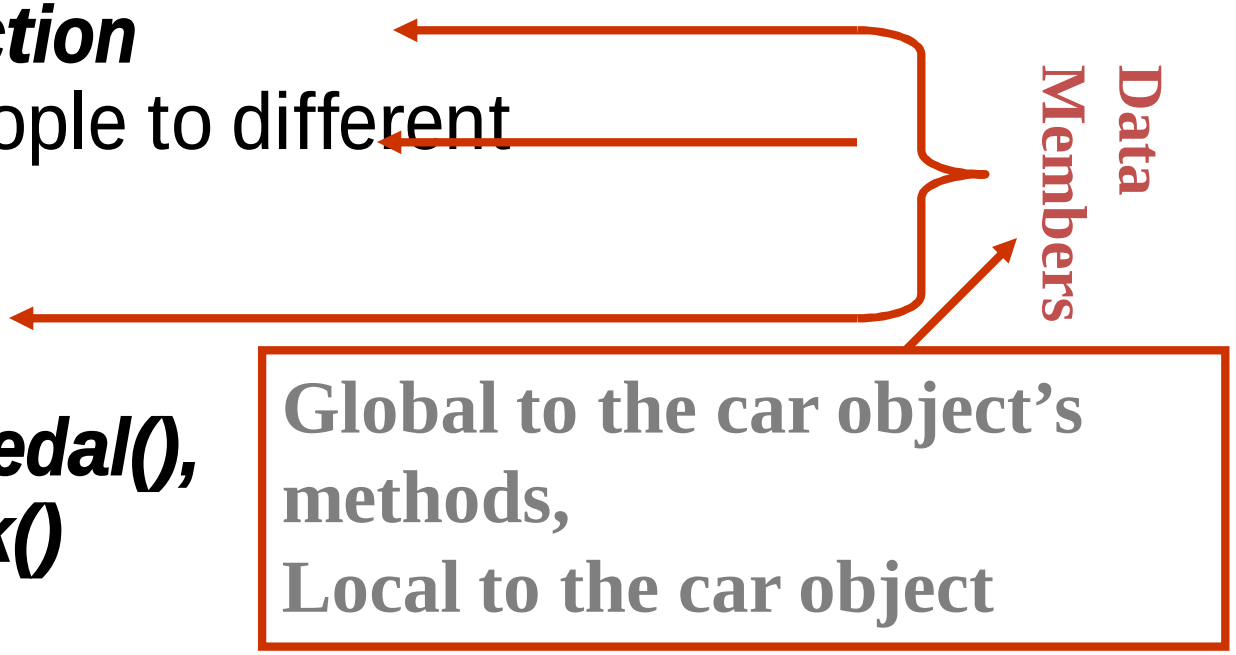
Object Oriented Programming Introduction cont..

- **Object – CAR**

  Manipulate its ***speed***, ***direction*** to transport people to different ***location***.

  Methods :

  Steer() , PressGasPedal(), PressBreak()



Object Oriented Programming cont...

- **Encapsulation** is achieved by using ***Modifiers***
  - ***Modifiers*** are Language Keywords that modify the definition of a ***Class, Method*** or a ***Variable***

Object Oriented Programming cont..

- **Inheritance** -
  - Concept of one class of objects inheriting the data and behaviour from another class of objects.

eg.

Object Oriented Programming cont..

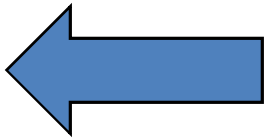
- *class* for a regular car.

Class **Car**

```
{
 direction;
 position;
 speed;

 Steer();
 PressGasPedal();
 PressBreak();
}
```

Data members



Methods

Object Oriented Programming cont..

Suppose you want to create a new car which has a special passing gear

Class PassingCar inherits from *Car*

```
{
  Pass();
}
```

Pass() is a method that implements the new functionality.

Object Oriented Programming cont..

- **Inheritance**
  - Through inheritance you can express the differences among related classes
  - You share the functions and member variables that implement the common features

Object Oriented Programming cont..

- **Inheritance** cont…
  - *Inheritance* also helps you to reuse existing code from one or more classes by simply deriving a new class from them.

Object Oriented Programming cont..

- **Inheritance** cont...
  - *Inheritance* is the creation of one class by extending another class so that instances of the new class automatically inherit the fields and methods of its parent class

Object Oriented Programming cont..

- **Inheritance contd...**

Inheritance promotes the class Reusability

It is because existing class can be sub classed through inheritance and any modifications necessary, could be applied to sub class (Overriding)

Object Oriented Programming cont..

- **Polymorphism**
  - Combines Greek Words Poly , Morphism

    Poly : Many

    Morphism : Forms
  - The quality of having more than one form.

Object Oriented Programming cont..

- In the context of Object Oriented Programming, polymorphism refers to the fact :
  - a single operation can have different behaviour in different objects

Say now you want a new kind of Car

- Having all the characteristics of the *PassingCar*
- Except that its passing gear is *twice* as fast as *PassingCar*

Class FastCar ...also inherits from *Car*

{

Pass();

}

- *Looks exactly like the original passing car*

Object Oriented Programming cont..

➢Polymorphism

It allows different forms of the same service to be defined.

There are two common ways of implementing Polymorphism. ***Overloading, Overriding***

Overloading - Using the same method name with different parameter type lists

Overriding - Using different implementations of the same method in sub classes.

Object Oriented Programming cont..

- **Encapsulation in Java**
- Access to variables and methods in Java classes is accomplished through ***Access Modifiers***
- Access modifiers define varying levels of access between class members and the outside world (other objects)

Object Oriented Programming cont..

- Access modifiers are declared immediately before the type of a member variable or the return type of a method.
- There are four access modifiers :
  - default (package access)
  - public
  - protected
  - and private

Object Oriented Programming cont..

- *Default access modifier*
  - There is no actual keyword for declaring default access modifier
  - It is applied by default in the absence of an access modifier.
  - Specifies that only classes in the same package (groups of related classes and interfaces) can have access to *class's variables and methods*

Object Oriented Programming cont..

- *Default access modifier cont...*

eg.

```
long length;
long getlength() {
   return length;
}
```

– Notice that neither the member variable nor the method supply an access modifier.

– So they take on the default access modifier implicitly

Object Oriented Programming cont..

- ***public*** access modifier
  - Specifies that class variables and methods are accessible to anyone, both inside and outside the class
  - This means that *public* class members have global visibility and can be accessed by any other objects.

  eg.

  ```
  public int count;
  public boolean isActive;
  ```

Object Oriented Programming cont..

- ***protected*** **access modifier**
  - **Specifies that class members are accessible only to methods in that class and subclasses of that class.**
  - **This means that protected class members have visibility limited to subclasses.**

```
eg.     protected char middleinitial;
        protected char getMiddleInitial() {
              return middleInitial;
            }
```

Object Oriented Programming cont..

- ***private*** access modifier
  - Specifies that class members are only accessible by the class they are defined in.
  - This means that no other class has access to private class members, even subclasses.

```
private String firstname;
private double howBigIsIt;
```

Object Oriented Programming cont..

- The ***static*** Modifier
  - The *static* modifier specifies that a variable or method is the same for all objects of a particular class.
  - When a variable is declared as being *static*, it is only allocated once regardless of how many objects are instantiated. (Typically new variables are allocated for each instance of a class)

Object Oriented Programming cont..

- The ***static*** Modifier cont..
  - All instantiated objects share the same instances of the *static* variable.
  - When you declare a method *static*, it is not instantiated with each object, but is part of the entire class.
  - Therefore you invoke the method by preceding it with the class name.

Object Oriented Programming cont..

- The *static* Modifier cont..
  - eg. (static member variable and a static method)

```
static int refCount;
static int getRefCount() {
  return refCount
  }
```

Object Oriented Programming cont..

- *Final* modifier
  - All methods and instance variables may be overridden by default.
  - If you wish to declare that you want to no longer allow subclasses to override your variables or methods, you can declare them *final*.

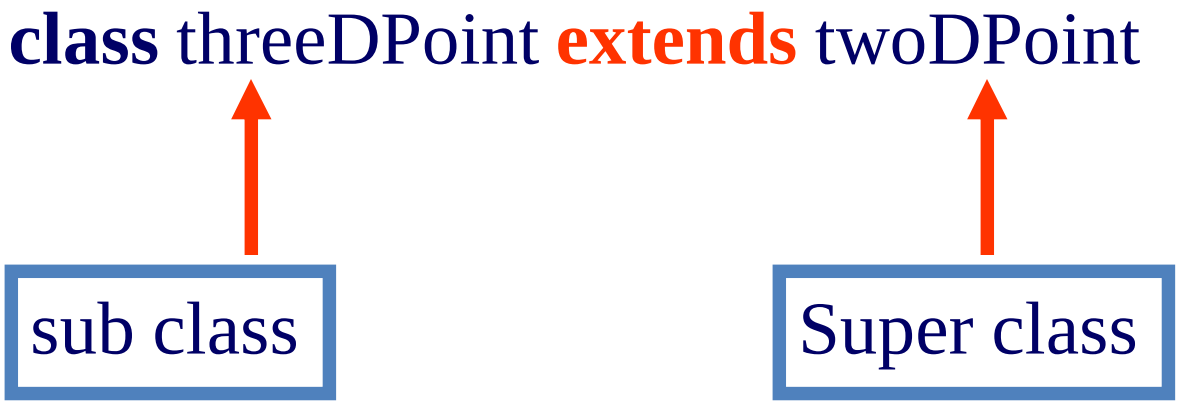
eg. final public int numDollars = 25;

Object Oriented Programming cont..

- *Other modifiers*
  - abstract,
  - *synchronized* ,
  - volatile,
  - native, etc.

Object Oriented Programming cont..

- **Inheritance** in Java
  - In Java the ***extends*** Keyword is used to achieve Inheritance



Object Oriented Programming cont..

➢ Inheritance Java cont...

A sub class can have its own implementation of the Super class methods.(Method Overriding)

```
public class threeDPoint extends twoDPoint {
        int z;
        public void Show( ) {
                super.Show();
                ……
        }
}
```

Show Method From super class is overridden

Accessing the super class Show Method

Object Oriented Programming cont..

➢ Inheritance Java contd...

A class that cannot be instantiated is known as an ***Abstract Class***

An Abstract Class

- is defined to be extended to sub classes
- uses ***abstrac***t Keyword in the Definition

```
abstract class Shape {
        abstract Point Center( );
        abstract double Diameter( );
        abstract double Area( );
}
```

Can be used to create any shape Such as Circle or Square

The End of Section