
# CMP2103:

# Object Oriented Programming (OOP)


# OOP Basic Concepts

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

# Abstraction

- Abstraction is an OOP design principle whose main focus is for simplification.
- It is the process of removing characteristics from something in order to reduce it to a set of essential characteristics.
- Through the process of abstraction, a programmer hides all but the relevant data about a class in order to reduce complexity and increase reusability.
- In summary, abstraction is the basic representation of a concept.

# Abstraction cont'd

- Abstraction allows programmers to represent complex real world problems in the simplest manner.
- It is the process of identifying the relevant qualities and behavior possessed by an object.
  - ✓ It thus allows for the extraction of the necessary features of an object without representing the background details.
- In OOP, abstract classes are defined as a framework for later extensions.

# Encapsulation

- Is the inclusion of property and method within a class/object for it's proper functioning.
  - Both the data and the functionality that could affect or display that data are included under a unified name (the object name itself)
- In the classic definition of encapsulation, the data elements or properties of the object) are not directly available to the outside world.
  - Instead, methods are created to give access to these values outside of the object.
  - This is made possible through the declaration of properties as public or private.


# Encapsulation cont'd

- Remember:
  - A class is a prototype, idea and blueprint for creating objects.
  - An object is an instance of a class.
- Encapsulation enables reusability of an instance of an already implemented class within a new class while hiding & protecting the method and properties from the client classes.


# Encapsulation - Benefits

- Ensures that structural changes remain local.
  - ✓ Changing the class internals does not affect any code outside of the class.
  - ✓ Changing a method's implementation does not reflect the clients using them.
- Encapsulation allows adding some logic when accessing client's data.
  - ✓ E.g. validation on modifying a property value.
- Hiding implementation details reduces complexity
  - ✓ Thus easier maintenance.

# Inheritance

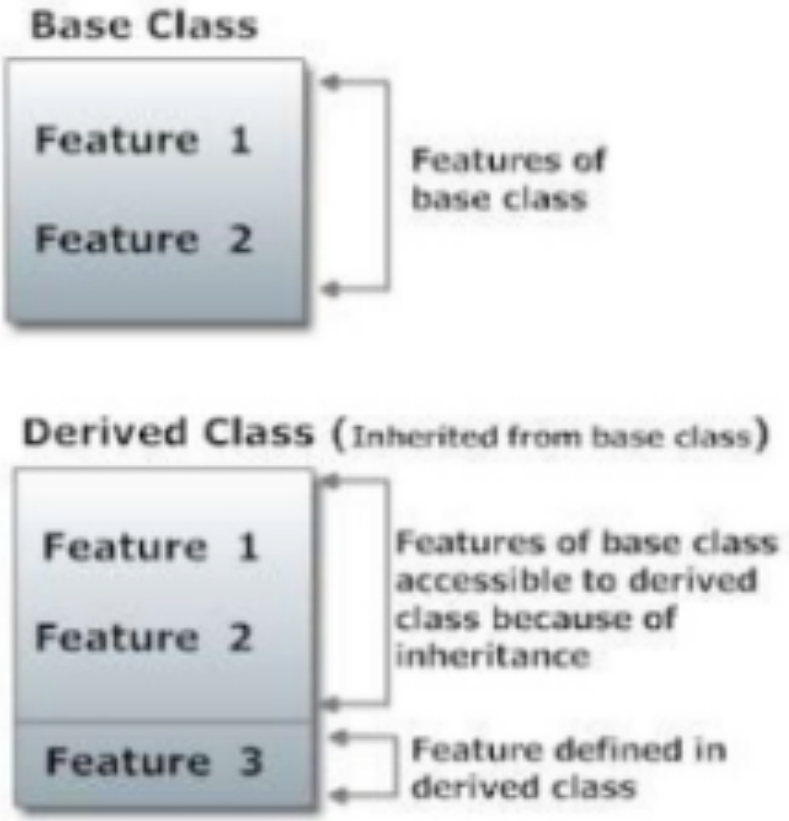
- The term “inheritance” in OOP keeps it’s regular meaning to indicate the transfer of traits like eyes, hair, genetics, etc.
- Inheritance allows developers to define objects in a hierarchy, much like a taxonomy chart or may be organizational chart.
- As a result, each level of the hierarchy defines a more specific object than the parent level.
- Each level also inherits every property and method of its parent object.

# Inheritance cont'd

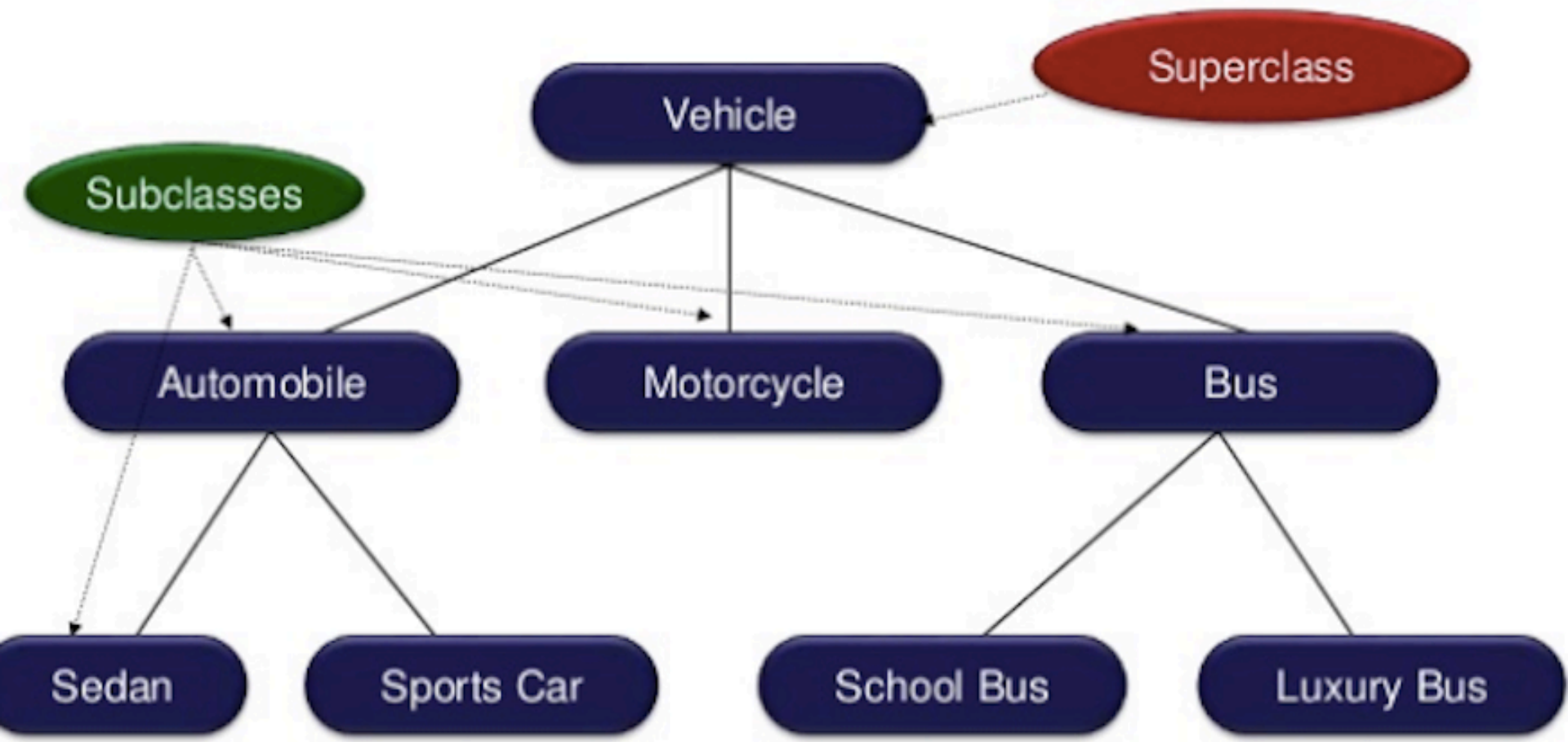
- Inheritance is thus a way of organizing classes.
- Classes with properties in common can be grouped so that their common properties are only defined in a parent class.
  - ✓ Superclass – transfer their attributes and methods to the subclass(es)
  - ✓ A subclass can inherit all its superclass attributes and methods besides having its own unique attributes and methods.

# Inheritance cont'd

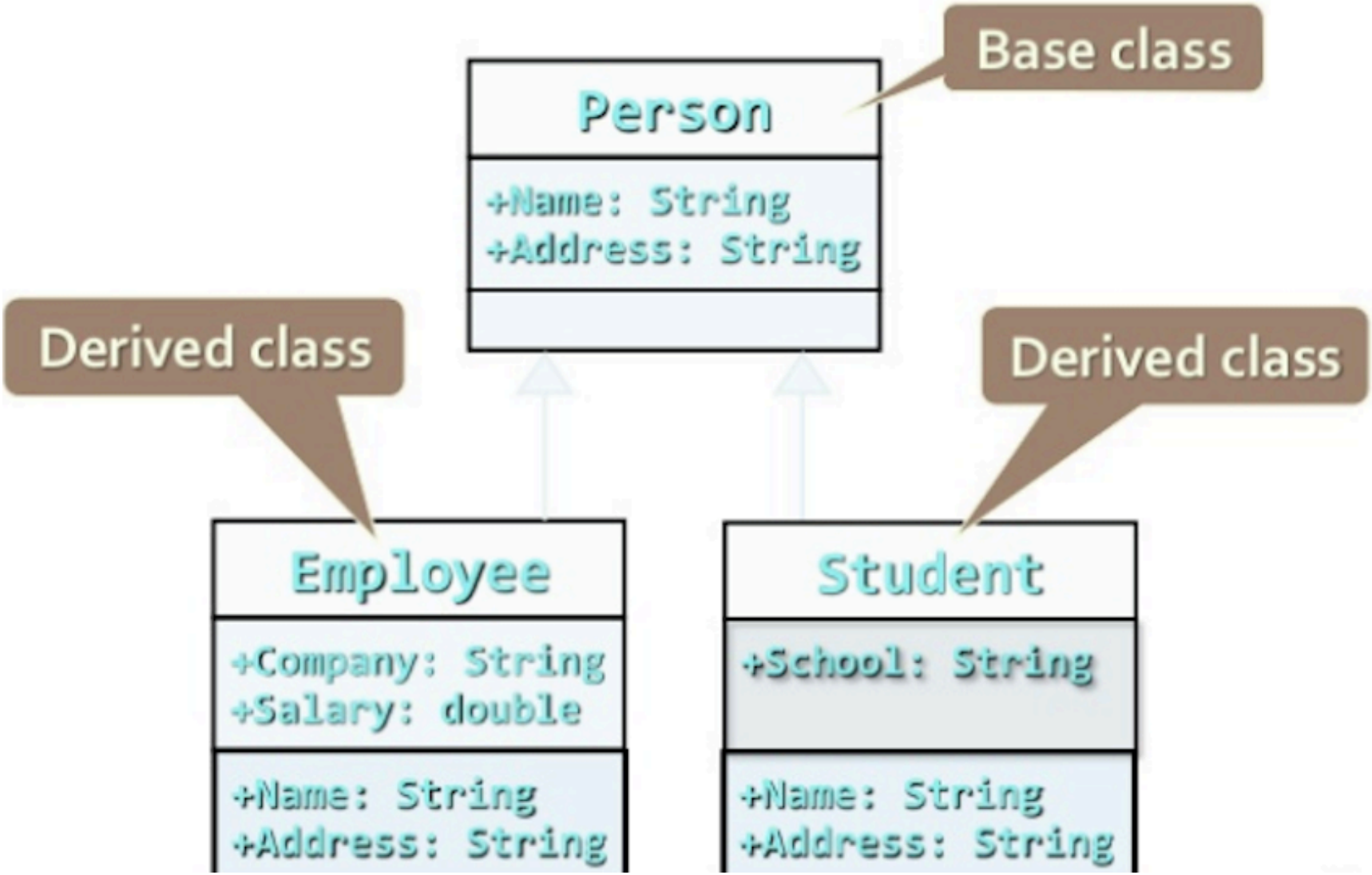
- In summary, inheritance:
- Expresses commonality among classes/objects.
- Allows for code reusability
- Highlights relationships
- Helps in code organization.



# An Inheritance Hierarchy



# An Inheritance Example



# Polymorphism

- In OOP, polymorphism means the ability to request that the same methods be performed by a wide range of different types of things.
  - ✓ This meaning does in fact try to maintain the generic meaning of polymorphism (many shapes).
- At every level of an object hierarchy, each object could have a method with the same name and because of the level at which the method resides, it would know which of the procedures or functions to call.

# Polymorphism cont'd

- As a technical issue and principle in OOP, polymorphism is achieved by using different techniques named method overloading, operator overloading and method overriding.
- For example:
  - If you had a Shape object with a Draw method, you could define a circle, triangle, polygon objects as Shape objects and override the Draw method to specifically draw the desired shape.
  - Any of the derived Shape objects would have the Draw method but only the right method for the right shape can be called.

# Polymorphism Cont'd

- Encapsulation, inheritance and abstraction concepts are very related to polymorphism.
- An object has "multiple identities", based on its class inheritance tree.
- It can be used in different ways.

# OOP Languages

- Two main subdivisions: Pure and Hybrid OOP languages.
- Pure OOP languages:
  - ✓ Eiffel
  - ✓ Actor
  - ✓ Emerald
  - ✓ JADE
  - ✓ Obix
  - ✓ Ruby
  - ✓ Python
  - ✓ Scala
  - ✓ Smalltalk
  - ✓ Self
- Hybrid OOP languages:
  - ✓ Delphi/Object Pascal,
  - ✓ C++
  - ✓ Java
  - ✓ C#
  - ✓ VB.NET
  - ✓ Pascal
  - ✓ Visual Basic
  - ✓ MATLAB
  - ✓ Fortran
  - ✓ Perl
  - ✓ PHP
  - ✓ ABAP

# Advantages of OOP

- ✓ Code reuse and recycling
- ✓ Improved software-development productivity
- ✓ Improves software maintainability
- ✓ Faster development
- ✓ Lower cost of development
- ✓ Higher-quality software
- ✓ Encapsulation


# Disadvantages of OOP

- Steep learning curve
- Could lead to larger program sizes
- Could produce slower programs

# Suitability of OOP

- OOP is good in complex projects or modular types of systems. It allows for simultaneous system development teams and also could aid in agile system development environments like Xtreme Programming.

# EXTRA MATERIAL

## Language translators:

- **Assembler** – a program that translates an assembly language program (written in a particular assembly language), into a particular machine language.
- **Compiler** – a program that translates a high-level language program (written in a particular high-level language), into a particular machine language.
- **Interpreter** – a program that translates a high-level language, one instruction at a time, into machine language.
- Interpreted languages are generally slower than compiled ones since they can be optimized to get faster execution.


# EXTRA MATERIAL

## Compilation Process:

- In the traditional compilation process, the compiler produces machine code for a specific family of processors.
- A main disadvantage of this is that the code produced is not portable.
- The concept of virtual machines mitigates this.
- The Java JVM (Java Virtual Machine) is a byte code interpreter that translates byte code into machine code.
- Instead of producing processor specific code, Java compilers


# EXTRA MATERIAL

## Java Virtual Machine (JVM):

- Instead of producing processor specific code, Java compilers produce an intermediate code called bytecode.
- Bytecode is also binary code but is not specific to a particular processor. The Java bytecode is then interpreted by the JVM.
- A java compiler will produce exactly the same bytecode irrespective of the computer system used.
- Java achieves compatibility by having a separate interpreter for each computer system.

# EXTRA MATERIAL

**Objects:**

- An object is an instance of a class. A object represents an entity – physical, conceptual, or software.
- Almost everything in the world can be represented as an object – a student, doctor, course unit, degree program, country, city, etc.
- More formally, an object is a computational entity that encapsulates some state, is able to perform actions or methods on this state and communicates with other objects.