Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Introduction to Object Oriented Programming (OOP)
2. History of OOP
3. Object and Class
4. Characteristics
5. OOP and Procedure Oriented Programming

Introduction to OOP

- OOP is the term used to describe a programming approach based on objects and classes.
- Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming.
- Object-oriented paradigm allows us to organise a software as a collection of objects that consist of both data and behaviour.
- The main aim of OOP is to bind together the data and the functions that operate on them, so that no other part of the code can access this data except that function.

Introduction to OOP (Cont…)

- OOP allows decomposition of a problem into a number of entities called objects, and then, builds data and functions around these objects.
  - A software can be divided into a number of small units called objects. The data and functions are built around these objects.
  - The data of the objects can be accessed only by the functions associated with that object.
  - The functions of one object can access the functions of another object.

Introduction to OOP (Cont…)

- **Where we can use OOP?**
  - Where the application can be decomposed into modules, which is known as **modularisation**.
  - Where an application can be composed from existing and new modules, i.e. **software re-use**.

History of OOP

- Many people believe that OOP is a product of 1980s.
- Bjarne Stroustrup was trying to move from C language into the object-oriented world by creating the C++ language.
- However, SIMULA 1 (1962) and Simula 67 (1967) are the two earliest object-oriented languages.
- While most of the advantages of OOP were available in the earlier Simula languages, OOP received popularity from the inception of C++.

History of OOP (Cont…)

- The next step of development of OOP started in Jan 1991, when J. Gosling, B. Joy, P. Naughton, and several others met in Aspen, Colorado, to discuss ideas for the Stealth Project.
- This group wanted to develop intelligent electronic devices capable of being centrally controlled and programmed from a handheld device.
- However, they felt that C++ was not up to the job.
- The result was the Oak programming language, which eventually morphed into the Java.
- Java quickly grew in popularity, spurred by the growth of the World Wide Web.

Object and Class

Object

- It is a basic unit of OOP and represents the real-life entities.
- An object is an instance of a class.
- When a class is defined, no memory is allocated. However, when it is instantiated (i.e., an object is created), memory is allocated.
- An object has an identity, state (instance variable or member variable) and behavior (method).
- Each object contains data and code to manipulate the data.
- Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Object and Class (Cont…)

- **Object (Cont…)**

Fig. 1: Example of an object

Object and Class (Cont…)

- **Object (Cont…)**

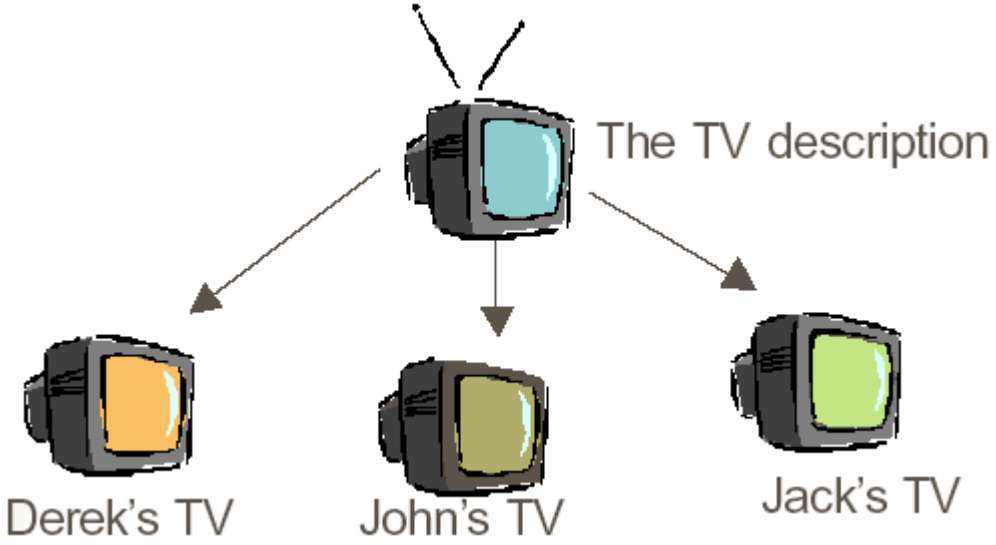

Fig. 2: Example of objects of a class

Object and Class (Cont…)

- **Class**
  - A class is a user-defined data type or a blueprint for an object.
  - It consists of data members and member functions, which can be accessed and used by creating an instance of that class.
  - It represents the set of properties or methods that are common to all objects of one type.
  - Class provide a well-defined interface.
  - When an object is created by a constructor of the class, the resulting object is called an instance of the class, and the member variables specific to the object are called instance variables.

Object and Class (Cont…)

Class (Cont…)

- Class in Java determines how an object behaves and what the object can contain.
- A class in Java can contain fields, methods, constructors, blocks, nested class and interface.
- For an example, consider the class, i.e., Car.
- There may be many cars with different names and brands.
- However, all of them share some common properties like all of them will have 4 Wheels, Speed Limit, Mileage Range, etc.
- Here, Car is the class, and wheels, speed limits, mileage are their properties.

Characteristics

Abstraction

- Abstraction is a general concept, which we can find in the real world as well as in OOP languages.
- An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects.
- Any objects in the real world like machine or classes in any software project that hide internal details provide an abstraction.
- It's the development of a software object to represent an object we can find in the real world.
- Encapsulation hides the details of that implementation.

Characteristics (Cont…)

- **Inheritance**
  - If we have several descriptions with some commonality between these descriptions, we can group the descriptions to provide a compact representation.
  - OOP approach allows us to group the commonalities and create classes that can describe their differences from other classes.
  - Classes may be created in hierarchies.
  - Inheritance lets the structure and methods in one class pass down the class hierarchy.
  - If a step is added at the bottom of a hierarchy, only the processing and data associated with that unique step must be added. Everything else above that step may be inherited.
  - Reuse is considered as a major advantage of object orientation.

Characteristics (Cont…)

- **Inheritance (Cont…)**

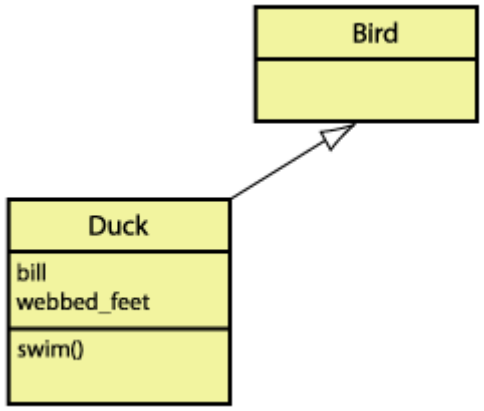

Fig. 3: Example of Inheritance

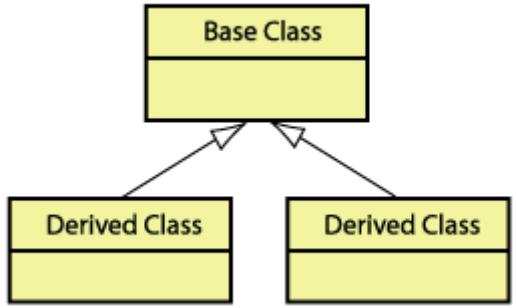

Fig. 4: Base class and derived class

Characteristics (Cont…)

- **Encapsulation**
  - Encapsulation is used to hide the mechanics of the object, allowing the actual implementation of the object to be hidden, so that we don't need to understand how the object works.
  - It refers to the creation of self-contained modules (classes) that bind processing functions to its data members.
  - The data within each class is kept private. Each class defines rules for what is publicly visible and what modifications are allowed.
  - Here, we need to understand the interface that is provided for us.

Characteristics (Cont…)

- **Encapsulation (Cont…)**
  - For an example, in a Television class, the functionality of the television is hidden from us.
  - However, we are provided with a remote control, or a set of controls for interacting with the television, providing a high level of abstraction.
  - Thus, there is no requirement to understand how the signal is decoded from the aerial and converted into a picture to be displayed on the screen.
  - There is a sub-set of functionality that the user is allowed to call, termed the interface.
  - In the case of the television, this would be the functionality that we could use through the remote control or buttons on the front of the television.
  - The full implementation of a class is the sum of the public interface plus the private implementation.

Characteristics (Cont…)

- **Encapsulation (Cont…)**

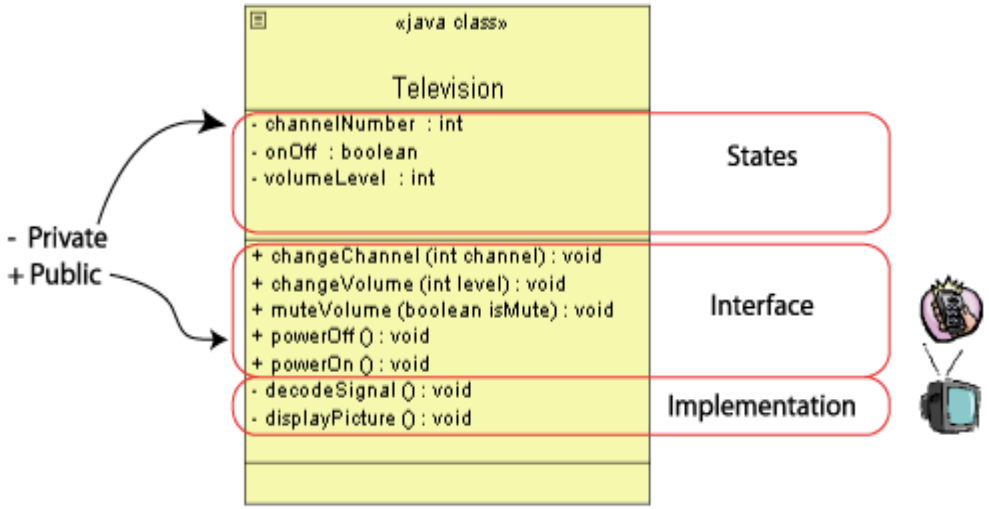

Fig. 5: Example of encapsulation

Characteristics (Cont…)

- **Polymorphism**
  - In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
  - Polymorphism means “multiple forms”.
  - Multiple forms refer to multiple forms of the same method, where the exact same method name can be used in different classes, or the same method name can be used in the same class with slightly different parameters.
  - There are two forms of polymorphism, over-riding and over-loading.
  - OOP lets programmers creating procedures for objects whose exact type is not known until runtime.
  - For example, a screen cursor may change its shape from an arrow to a line depending on the program mode. The routine to move the cursor on screen in response to mouse movement can be written for “cursor”, and polymorphism lets the right version for the given shape be called.

OOP and Procedure Oriented Programming

Procedural Oriented Programming	Object Oriented Programming
Here, a program is divided into small parts called **functions**.	In OOP, a program is divided into small parts called **objects**.
This follows **top down approach**.	OPP follows **bottom up approach**.
There is no access specifier.	It has access specifiers like private, public, etc.
Adding new data and function is not easy.	Adding new data and function is easy.
This is **less secure**.	It provides data hiding, so it is **more secure**.
Here, function is more important than data.	In OOP, data is more important than function.
It is based on **unreal world**.	OOP is based on **real world**.
Examples: C, FORTRAN, Pascal, Basic, etc.	Exemples: C++, Java, Python, C#, etc.

thank you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.