Chapter 1
Introduction to Object-Oriented Programming (OOP)

Objectives

After you have read and studied this chapter, you should be able to: -

- Differentiate procedural Programming and OOP.
- Name the basic components of object-oriented programming
- Differentiate classes and objects.
- Describe significance of inheritance in object-oriented programs.

Programming paradigms

There are different types of programming languages, such as:

Procedural Programming: is a list of instructions telling a computer, step-by-step, what to do, usually having a linear order of execution from the first statement to the second and so forth with occasional loops and branches. Procedural programming language includes C, FORTRAN, Pascal, and Basic.

Procedural languages are characterized by sequential sets of linear commands. The focus of such languages is on *structure*.

Object oriented programming: allow the programmer to break down the problem into *objects*: self-contained entities consisting of both data and operations on the data. E.g. java, C#, VB.NET

- The focus of OOP languages is not on structure, but on *modeling data*.

Advantage of OOPs over Procedure-oriented programming language

1) OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

2) OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.

3) OOPs provides ability to simulate real-world event much more effectively.

We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

1.1. Overview of Object Oriented Programming

The OO paradigm aims to eliminate some of the flaws of the procedural approach to programming. In OO, *data* is critical – data is not allowed to move freely around a system, but it is tied closely to the functions that operate on it. Data is also protected from unintentional modification by other functions.

In OO programming, a system or problem is decomposed into entities called ***objects***. An object has *data members* and functions – in Java, functions are known as *methods*.

Data is protected because the data belonging to an object can only be accessed and modified by the methods of that object.

Different objects can 'talk' to each other through their methods – in this way, Object A cannot directly modify data belonging to Object B – but it can call a method of Object B that in turn modifies the data.

Object means a real word entity such as pen, chair, table etc.

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some core concepts:

- ✓ Class
- ✓ Object
- ✓ Encapsulation
- ✓ Inheritance
- ✓ Abstraction
- ✓ Polymorphism

To summarise the OOP approach:

- ✓ Emphasis on data, not procedure
- ✓ Programs are made up of objects
- ✓ The data structures reflect characteristics of the objects
- ✓ Methods that operate on the data of an object are tied into the data structure
- ✓ Data is hidden from external functions
- ✓ Objects can communicate with each other through methods
- ✓ New data and methods can be easily added when necessary

✓ Takes a *bottom-up* approach to program design

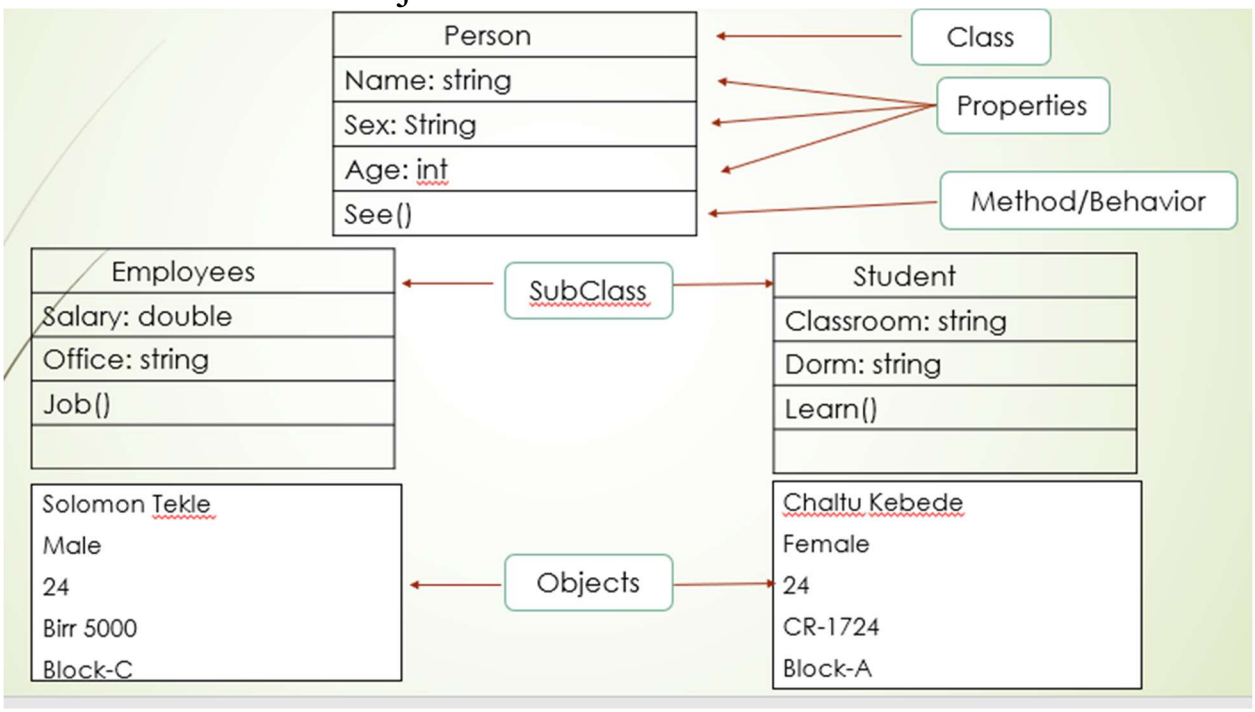
1.2. Classes and Objects

✓ Object-oriented programs use objects.

✓ Objects are the basic runtime entities in an OO program. An *object*:

- Is an identifiable thing or individual?
- Is of significance to the system
- Has characteristic behaviour (i.e. functionality)
- Has states reflected in the attributes (data) held about the object.

A *class* is the *definition of a set of objects* which are all similar in terms of their attributes, associations and functionality e.g. customer class defines all customer objects.



Methods can be *public* or *private*.

Public methods:

- Are triggered by receipt of a message from another object
- Have a public interface
- Can be defined at the class or object level

- Can be inherited (see Inheritance below)

Private methods:

- Are triggered by methods within an object
- Are internal to a class
- Can be changed without affecting any methods outside the class

A class is a bit like a data-type – once a class has been defined, any number of objects belonging to that class can be created. The objects then are a bit like variables – each object is associated with the class type from which it was created.

Technical contrast between Objects & Classes

CLASS	**OBJECT**
Class is a data type	Object is an instance of Class.
It generates OBJECTS	It gives life to CLASS
Does not occupy memory location	It occupies memory location.
It cannot be manipulated because it is not available in memory *(except static class)*	It can be manipulated.

1.3. Data Abstraction and Encapsulation

Encapsulation means the wrapping up of data and methods into a single unit (a class). It also means that the data inside a class is hidden from everything outside the class.

- ✓ The data can only be accessed by invoking the methods of the class. The internal workings of the methods are of no concern to other classes in the program.
- ✓ Information Hiding.
- ✓ Implementation can change without effecting any calling code.
- ✓ The class is kind of a container or capsule or a cell, which encapsulate the set of methods, attributes and properties to provide its indented functionalities to other classes.

Abstraction refers to the way in which a class represents only the essential features of the set of objects it models – a class does not include background or extra details that are not relevant to the system.
So a class is like a list of abstract attributes e.g. size, weight, cost, and the methods that operate on those attributes e.g. change the cost, increase the weight.

- ✓ Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.
- ✓ It is the process of removing characteristics from something in order to reduce it to a set of essential characteristics.

1.4 **Inheritance**

Inheritance is the way in which objects of one class get the properties of objects of another class. This includes the data and the methods.
For example, take a system in which there are various types of 'person' objects – they have some attributes in common, such as Name, Address and Date of Birth. The person that is an Employee is part of the class 'Person', but has some extra attributes such as Salary, Department and Rank.

- ✓ On the surface, inheritance is a code re-use issue. we can extend code that is already written in a manageable manner.

Benefits of Inheritance

- ✓ Used to avoid redundancy.
- ✓ Useful for easy modification (modification can be made only once)
- ✓ New classes can be easily added (sometimes may require restructuring)
- ✓ Used to generalize/specialize classes

1.4. **Polymorphism**

- ✓ The word polymorphism means the ability to take more than one form. In terms of the OOP, this means that a particular operation may behave differently for different sub-classes of the same class.

- ✓ Consider a class hierarchy for different shapes. A Shape class encapsulates the common features of shapes – such as drawing a shape and calculating the area of a shape (or, in other words, each Shape object can draw itself and can calculate its area).
- ✓ Deals with the ability of an object to be of different forms.
- ✓ Polymorphism comes in the form that a reference in an OO program can refer to objects of different types at different times

1.5. Java Technology

What is Java Technology?

- ❖ **The Java technology is:**
  - ✓ A programming language
  - ✓ A development environment
  - ✓ An application environment
  - ✓ A deployment environment
- ➢ **A programming language**
  - o As a programming language, Java can create all kinds of applications that you could create using any conventional programming language.
- ➢ **A Development Environment (JDK)**
  - o As a development environment, Java technology provides you with a large suite of tools:
    - ✓ A compiler (javac)
    - ✓ An interpreter (java)
    - ✓ A documentation generator (javadoc)
    - ✓ A class file packaging tool and so on...
  - o The JDK additionally contains the development tools necessary to create Java programs. The JDK therefore consists of a Java compiler, the Java virtual machine and the Java class libraries
- ➢ **An Application and Runtime Environment**
  - ✓ Java technology applications are typically general-purpose programs that run on any machine where the Java runtime environment (JRE) is installed.

- ✓ The JRE consists of the JVM and the Java class libraries. Those contain the necessary functionality to start Java programs.

Key features of Java technology…

Java Features…

- Java Virtual Machine (JVM)
  - ✓ an imaginary machine that is implemented by emulating software on a real machine
  - ✓ provides the hardware platform specifications to which you compile all Java technology code
- Bytecode
  - ✓ a special machine language that can be understood by the Java Virtual Machine (JVM).
  - ✓ Independent of any computer hardware, so any computer with a Java interpreter can execute the compiled Java program, no matter what type of computer the program was compiled on.
- Garbage collection thread
  - ✓ responsible for freeing any memory that can be freed.
  - ✓ This happens automatically during the lifetime of the Java program.
  - ✓ programmer is freed from the burden of having to deallocate that memory themselves
- Code Security
  - ✓ Code security is attained in Java through the implementation of its Java Runtime Environment (JRE).
  - ✓ JRE runs code compiled for a JVM and performs class loading (through the class loader), code verification (through the bytecode verifier) and finally code execution.
  - ✓ class files are loaded into memory on demand, as needed by the program.
  - ✓ The class responsible for finding and loading class files into memory at run time is called Class Loader.
  - ✓ *It adds security by separating the namespaces for the classes of the local file system from those that are imported from network sources.*

About Java programs, it is very important to keep in mind the following points.

- **Case Sensitivity -** Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java
- **Class Names -** For all class names, the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case. Example *class MyFirstJavaClass*
- **Method Names -** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case. Example *public void myMethodName()*
- **Program File Name -** Name of the program file should exactly match the class name. When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match your program will not compile). Example : Assume 'MyFirstJavaProgram' is the class name, then the file should be saved as *'MyFirstJavaProgram.java'*
- **public static void main(String args[]) -** Java program processing starts from the main() method, which is a mandatory part of every Java program. Java Identifiers: All Java components require names. Names used for classes, variables and methods are called identifiers. In Java, there are several points to remember about identifiers. They are as follows: All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_). After the first character, identifiers can have any combination of characters. A keyword cannot be used as an identifier. Most importantly identifiers are case sensitive. Examples of legal identifiers: age, $salary, _value, __1_value