Section 1: Lecture 2

Object Oriented Concepts

Access Modifiers: Controlling access to a class, method, or variable (public,

protected, private, package), Other Modifiers, Polymorphism: Overloading,, Inheritance,

Overriding Methods, Abstract Classes, Reusability, Class's Behaviors.

Introduction

Any object oriented language compose of some common features one should know

- **Classes**
- **Objects**
- **Programming Encapsulation (Information**
- **Hiding)**
- **Inheritance**
-

Classes & Objects

Class is a collection of member data and member functions. objects contain data and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data typ with the help of a class. In fact, objects are variables of type class. Once a class has -been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created: A class is thus a collection of objects of similar type. For example, mango, apple and orange are members of the class fruit: Classes are user-defined data types and behave like the built-in types of a programming language. For example, the syntax used to create an object is no different than the syntax used to create an integer object in C. If fruit has been defined as a class, then the statement fruit mango; will create an object mango belonging to the class fruit

Data Abstraction and Encapsulation

- The wrapping. up of data and functions into a single unit (called class) is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. This insulation of the data from direct access by the program is called data hiding.
- Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes. They encapsulate all the essential properties of the objects that are to be created. Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

Inheritance

- Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. For example, the bird robin is a part of the class flying bird which is again a part of the class bird. As illustrated in Fig.l.8, the principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived. In, OOP, the concept of inheritance provides the idea of reusability .This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined features of both the classes. The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class that is almost, but not exactly, what he wants; and to tailor the class in such a way that it does not introduce any undesirable side effects into the rest of the classes.
- Note that each sub class defines only those features that are unique to it. Without the use of classification, each class would have to explicitly include all of its features.

Polymorphism

- Polymorphism is another important OOP concept Polymorphism means the ability to take more than one form.. For example, an operation may exhibit different behavior m different instances.. The behavior depends upon the types. of data used in the operation For example, consider the operation of addition. For two numbers, the operation will generate a sum lf the operands are strings, then the operation would produce a third string by concatenation. Figure, l.9 illustrates that a single function name can be used to handle different number and different types of arguments. This is something similar to a particular word having several different meanings depending on the context.
- Polymorphism plays an important in allowing objects having different internal structures to share the same external interface. This is means that a general class of operations may be accessed in ~?the same manner even though specific actions associated with each operation may differ: Polymorphism is extensively used in implementing inheritance.

Access Specifiers

- **Access specifiers** defines the access rights for the statements or functions that follows it until another access specifier or till the end of a class. The three types of access specifiers are "private", "public", "protected".
- **<u>private:</u>** The members declared as "private" can be accessed only within the same class and not from outside the class.
- **<u>public:</u>** The members declared as "public" are accessible within the class as well as from outside the class.
- **<u>protected:</u>** The members declared as "protected" cannot be accessed from outside the class, but can be accessed from a derived class. This is used when inheritance is applied to the members of a class.
-