Object-Oriented Programming

OOP

What is OOP all About?

- Traditional Programing: A Process or a procedure that has been coded to be run on a computer.
- The original idea of a computer ("The Early Days") was to run, many times over, a well defined process or procedure.
  - ✓ such as computing payroll for a large company or computing interest on bank accounts
  - ✓ We will refer to this idea as "Procedural Programming" or "Top-Down Functional Decomposition"

So What is it all About?

- The traditional (or procedural) method of programming is fine
- However, as systems became more complex:

➡ System behavior became difficult to characterize as a procedure

➡ Requirements of performance and cost effectiveness increased

So What is OOP?

- Software is designed as a collection of objects that interact with each other through methods (functions)

➡ For example, in a payroll system the employees are an object and payroll checks are another object.

➡ A method (or function) associated with the payroll checks object could be to print the checks.

Object-Oriented

- Objects are the main feature of Object-Oriented design and programming
- In the traditional method of design and programming, the main feature of the design of a system (program) is the process (or procedure) on how to solve a problem or perform a task.
- The traditional method of design can be greatly affected by change.

Object-Oriented

- Objects are defined (declared) along with their appropriate functions. These functions (or methods) operate on the object.

➡ Hence we have an abstraction that is Not dependent on changing requirements of the application.

Object-Oriented

- Classes: Allow the software designer to view objects as entities (much like a relational database entities)
- When designing the software system, the developer specifies properties of objects that will be needed in the system

Object-Oriented Example

- for example, in a Library management system "books" are objects and their properties can be:
  - ✓ Catalogue Call Number
  - ✓ Title
  - ✓ Author
  - ✓ Is the book in or checked out

Abstract Specifications

- The properties, as we specified in the Library example, are *abstract*.

  ➡ abstract means there are no restrictions on how the functionality is developed.

  ➡ This specification is called *an abstract class*.

  ➡ An abstract class works as a "template" for the developer

UML

- The Unified Modeling Language (UML) is the standard tool (or "blueprints") for describing the final software product.

Standard Solutions

- Design Patterns: The Object structure allows the developers to build standard solutions to common problems. Design patterns are a common form of reuse of solutions.

Adaptability

- Software is flexible - Hardware is Not
  - That is, the developers (or maintainers) can modify objects to create new objects.
  - By using *Inheritance*, a new class can be created, called the *descendant class*, that can modify the features of an existing class (called an *ancestor class*).
  - This new class is a convenient way to specialize or extend the ancestor (or parent) class.

Inheritance - an example

- Given a class called *Window* (which displays some menu), the developer can specialize it through *inheritance* by deriving a descendent (or child) class called MenuWindow.
- Inheritance hence supports a form of code reuse.

Modular Design

- Designing a “Large” system by putting together a number of distinct software components.
- This concept makes a complex system easier to understand

More on Modular Design

- The objective is that a *module* clearly specifies what it does but does not uncover its implementation.
- This concept is called *encapsulation*. That is, the module hides the details of its implementation.

Encapsulation

- In Procedural Programming, the modules are procedures (functions) and data manipulated by procedures.
  - data are usually passed through arguments and a value returned by a function
- Data and the procedures to manipulate the data are encapsulated - contained within a class.

Encapsulation -Example

- For example, take a String class that can be used to do the following:

✓ create strings

✓ concatenate strings

✓ change a specified character in a string

✓ count characters in a string (determine length)

Example Continued

- The String class would have data members (variables) that represent the characters in a String and function (method) members to manipulate the data members.

- Such methods can be:
  - ✓ a method to create a new string
  - ✓ a method to check whether a string contains a particular character
  - ✓ a method to copy a string