Reference: http://dos.iitm.ac.in/OOSD\_Material/Basic%20Concepts/Basic%20Concepts%20Of%20OO/s1-M1-P1.htm

**Basic Concepts of Object-Orientation**

**Abstraction**

* Essence of abstraction is to deal with the problem at a higher level ignoring low level and possibly not so important details concerning the problem at hand.
* A good example of abstraction is to consider people of the globe. Dealing with all the few billion people in solving the world problems such as global warming, is to look at the problem without creating proper abstractions. The abstraction that was created after World War II is the concept of United Nations. At the level of United Nations, individual nations form the abstraction. At a country level, individual states form the abstractions. At the state level, towns and cities form the abstraction. At the level of United Nations, dealing with towns and cities enormously complicates the problem.
* Hence abstraction is a way of managing problem complexity by focusing on the essential details and ignoring lower level details.

**Forms of Abstractions in Programming World:**

* Procedural Abstraction
* Data Abstraction

**Procedural Abstraction:**

Procedural abstraction provides mechanisms for abstracting well defined procedures or operations as entities. The implementation of the procedure requires a number of steps to be performed. A simple example is a *debit* operation which performs various steps to debit certain amount from the bank account. Hence at the banking level, *credit* and *debit* become well defined procedural abstractions.

Procedural abstractions are used extensively by requirements analysts, as well as designers and programmers.

Procedural abstractions are normally characterized in a programming language as "function/sub-function" or "procedure" abstraction.

**Data Abstraction:**

This principle is at the core of Object Orientation. In this form of abstraction, instead of just focusing on operations, we focus on data first and then the operations that manipulate the data. A simple example is queue data and the associated operations add() and delete(). Both add() and delete() operations manipulates queue data. In a simple procedural abstraction, there would be only add and delete operations separately but their association with the queue data will not be captured. Similarly, in the bank example discussed above, credit and debit operations get associated with the bank account data.

The advantage of data abstraction over procedural abstraction is that the data and the associated operations get specified together and hence it is easy to modify the code when data changes. For example when the queue data structure changes from array structure to linked list, it is easy to figure out the associated operations that need to be changed. It is also easy to add code associated with requirements change. For example if the queue needs to be reset at regular intervals, a simple reset() can be added.

Quiz: (1) Identify simple data abstractions in your kitchen and associated operations. Justify these abstractions by showing that when requirements change, these abstractions will help to manage the change.

(2) Identify operations that can be associated with the *counter* abstraction. Is it possible for you to associate irrelevant operations to *counter* abstraction? (Example - Associating *paint* operation to door, when the required operation is *open* ).