**Exercise 1- Devashish Bishnoi**

**Encapsulation:**

It is a way to ensure that users do not have access to the data and method inside a class. The only way in which they may be accessed will be through public methods. In order to make this possible, programmers use the access variable private and provide public getter and setter methods for these type variables. This creates a program with tight encapsulation and allow complete control on how data is to be read and how data is to be changed. The benefit of this is information hiding where the users are not aware of how data is stored internally.

**Inheritance:**

Classes are organised in hierarchy in order to avoid duplication and reduce redundancy. The classes in the lower down in the hierarchy inherit all the variables and methods from the higher hierarchies. A subclass is a class found in the lower hierarchy, it is also known as the child class. However, a superclass class is a class which is found in the upper hierarchy, also known as a parent class. Superclass will contain all the variables and methods common to all subclasses and subclasses will contain specialised variables and methods. Thus, redundancyis greatly reduced or eliminated as these common variables and methods will not need to be repeated in all the subclasses.

So, a subclass inherits (gets access to) all the variables and methods from its superclasses, including its immediate parent as well as all the ancestors. It is important to note that a subclass is not a "subset" of a superclass. Subclass is a "superset" of a superclass and not a “subset” as a subclass inherits all the variables and methods of the superclass. In addition, it can have its own more specific variables and methods.

**Polymorphism:**

Polymorphism separates the interface and implementationallowing the programmer to program at the interface in the design of a complex system.

This is can be done by allowing the superclass to provide a common interface to all its subclasses, which will carry out the actual implementation. We do not want user to instantiate instance of a superclass. So, a subclass instance will process all the attributes operations of its superclass meaning that whenever a superclass instance is expected, it can be substituted by a subclass instance. This creates substitutability as a reference to a class may hold an instance of that class or an instance of one of its subclasses.

For example, if a subclass instance is assigned to a superclass reference, this means only the methods defined in the superclass can be invoked; methods defined in the subclass can’t be.

**Abstraction:**

Abstraction is where a class contains one or more abstract methods. This method will only have name, list of arguments and the return type. It will not contain any implementation i.e., the method’s body. The keyword abstract is used to declare an abstract method and an abstract class. Implementation of these methods is not possible in the current (superclass) class. The implementation of these abstract methods must be provided later once it has a subclass. Thus, a class containing abstract methods cannot be instantiated, as its definition is not complete.

To use an abstract class, a subclass is needed to be derived from the abstract class. Then in the derived subclass the abstract methods need to be overridden thus providing implementation to all the abstract methods. So, now a subclass can be instantiated.