**Encapsulation:**

Encapsulation is used to bind together the variables and methods together as a single unit. It keeps the data safe from outside interference and misuse which lead to a concept of data hiding. In Encapsulation the fields are declared private, it cannot be accessed by anyone outside the class. To make this possible, programmers use the access variable private and provide public getter and setter methods for these type variables.

**Inheritance:**

Inheritance is the capability of a class to use the properties and methods of another class while adding its own variables and methods. Using inheritance, the data is made more manageable in a hierarchal order. For example: Cars and bikes have same attributes such as Engine, gearbox, make and model. These attributes can be transferred to a new class named “Vehicle” and this will become a super class. A subclass inherits (gets access to) all the variables and methods from its super classes, including its immediate parent as well as all the ancestors.

**Abstraction:**

It refers to the ability to make a class abstract in object oriented programming and It needs to be extended and its method implemented. It cannot be instantiated. It is used to reduce the complexity and improves the maintainability of the system. 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.

**Polymorphism:**

Polymorphism is the ability of an object to take on many forms. It separates the interface and implementation allowing the programmer to program at the interface in the design of a complex system. The most common use of polymorphism in OOP occurs when a super class reference is used to refer to a sub class object. 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.