Object Oriented Programming

It is a programming methodology that divides implementation of large application software into independently programmable units called objects each with its own attributes (data) and its own behavior (code).

Basic Elements of OOP

1. **Specification:** Every object has a class that defines a set of fields (variables) and a set of methods (functions), which respectively describe the attributes and the behavior of that object.
2. **Instantiation:** An object is activated (created) from a class by first allocating a memory-block for holding values of the fields defined by the class and then initializing these values by calling the constructor method defined by the class.
3. **Identification:** Every object has a unique identity and when a method defined by a class is called on its object the identity of this object is passed to the implementation of that method.
4. **Containment:** An object that holds another object in its field exhibits one of the following types of *'has a'* relationship with that object
   - **(a) Composition:** It is a type of containment in which the lifetime of the inner object is controlled by its outer object.
   - **(b) Aggregation:** It is a type of containment in which the lifetime of the inner object is independent of the outer object.
5. **Subtyping:** A derived class[1] extends an existing base class[2] to define additional fields and methods or to override (provide new) implementations for its existing methods.
6. **Polymorphism:** An object of a derived class can be treated as an object of its base class and when a method defined by the base class is called on an object of its derived class which has overridden its implementation then this implementation is invoked at runtime[3].

[1] Subclass

[2] Superclass

[3] Dynamic binding

7. **Interface:** An abstract class that does not support instantiation defines a set of pure (unimplemented) methods, which can be implemented by its non-abstract (instantiation capable) derived classes.
8. **Inheritance:** An object of a derived class exhibits one of the following types of *'is a'* relationships with its base class.
   (a) Realization: It is a type of inheritance in which the base class is abstract but the derived class is non-abstract.
   (b) Specialization: It is a type of inheritance in which the base class and the derived class are both non-abstract (or abstract).

Fundamental (SOLID) Principles of OOP

1. **Single Responsibility Principle:** An object should handle exactly one requirement of its application. [*A class should only define members which are required by all of its objects*]
2. **Open Closed Principle[1]:** The set of features exposed by an object should be open for extension but closed for modification. [*A class should only publish its methods and not its fields[2]*]
3. **Liskov's Substitution Principle:** An object of a derived class should be used as a replacement of an object of its base class. [*Any usage of a derived class object should be implemented to work with its base class*]
4. **Interface Segregation Principle:** Each functionality supported by an object should be exposed through a separate interface. [*A non-abstract class should only implement pure methods of its abstract base classes*]
5. **Dependency Inversion Principle[3]:** An object should be used through its interface and not through its implementation. [*A non-abstract class should only be used for instantiation and any usage of its object should be implemented to work with its abstract base classes*]

[1] Abstraction

[2] Encapsulation

[3] Loose coupling