**Class:** A class is like a blueprint or a template for creating objects. Just like how a blueprint defines the structure and characteristics of a building, a class defines the properties and behaviours of objects.

**Object:** An object is an instance of a class, similar to how a house is an instance of a blueprint. Each object has its own unique identity, state, and behaviour, but follows the structure defined by its class.

**Encapsulation:** Encapsulation is like a capsule or a container that wraps together data (attributes) and methods (behaviours) that operate on the data. This encapsulation ensures that the internal workings of an object are hidden from the outside world, and only the necessary interfaces are exposed for interaction.

**Inheritance:** Inheritance is akin to the concept of inheritance in real life, where children inherit certain traits or characteristics from their parents. Similarly, in OOP, a class can inherit attributes and methods from another class, allowing for code reuse and creating a hierarchy of classes.

**Polymorphism:** Polymorphism is like a person having multiple roles. For example, a person can be a teacher, a parent, and a musician at different times. Similarly, in OOP, polymorphism allows objects of different classes to be treated as objects of a common superclass, enabling flexibility and extensibility in the code.

**Abstraction:** Abstraction is like using a TV remote control without needing to understand the internal mechanisms of the TV. Similarly, in OOP, abstraction allows you to focus on the essential features of an object while hiding the complex implementation details.