Inheritance:  
Inheritance is used to derive a new class from an existing class, enabling code reuse. The properties and behaviours of the parent class are inherited by the subclass, allowing for hierarchical relationships and specialized functionality.

Polymorphism:  
Polymorphism is applied to allow objects of different types to be treated as instances of a common class. Method overriding and overloading are supported to provide multiple implementations of a method, depending on the object at runtime or compile-time.

Abstraction:  
Abstraction is the process of hiding implementation details and expose only the essential features of an object. Complex systems are simplified by focusing on high-level operations, which helps reduce complexity and improve code maintainability.

Encapsulation:  
Encapsulation is used to bundle data and methods that operate on the data within a class. Access to the internal state is restricted through public interfaces, ensuring controlled modification and protecting the object’s integrity from external interference.

Static Binding:  
Static binding is determined at compile time, where method calls and variable references are resolved early. It is applied to methods that are final, private, or static, ensuring faster execution and preventing any changes during runtime.

Dynamic Binding:  
Dynamic binding is resolved at runtime, where the method to be invoked is determined based on the actual object type. It is utilized in polymorphism, allowing flexible behavior and enabling method overriding in subclasses.