Simple Factory Pattern:

- Appropriate when you want to create objects without exposing their instantiation logic to clients.
- Useful when you have multiple classes that share a common interface or base class and need to create instances of these classes based on certain conditions.
- Has a central factory class with a static method for object creation.
- The client code interacts with this factory class to obtain instances of objects without direct knowledge of their concrete types.
- Key Components:
    - Creator (Factory) Class: Contains a static factory method for creating objects.
    - Concrete Products: Classes that implement a common interface or extend a common base class.
- Problem Solved:
    - Centralizes object creation logic, reducing code duplication.
    - Encapsulates object creation details, making the system more maintainable and flexible.
    - Provides a single entry point for creating objects.

Factory Method Pattern:

- Appropriate when you want to delegate the responsibility of object creation to subclasses.
- Useful when you have a family of related classes, and clients should work with these classes through a common interface.
- Has an abstract creator class or interface declaring a factory method.
- Subclasses of the creator class implement the factory method to create specific objects.
- Key Components:
    - Creator (Factory) Class: Declares the factory method, which returns an object of a common interface or base class.
    - Concrete Creators: Subclasses that implement the factory method to create specific products (objects).
    - Products: Classes that implement a common interface or extend a common base class.
- Problem Solved:
    - Supports extensibility by enabling the addition of new products (objects) without modifying existing code.
    - Promotes loose coupling by allowing clients to work with the creator's interface, abstracting concrete class details.

Abstract Factory Design Pattern:

- Appropriate when you need to create families of related or dependent objects without specifying their concrete classes.
- Useful for building complex systems with multiple interrelated objects.
- Has multiple abstract factory classes (one for each family of objects).
- Concrete factory classes implement these abstract factories to create families of related objects.
- Key Components:
    - Abstract Factory: Declares factory methods for creating objects belonging to a family.
    - Concrete Factory: Implements the abstract factory, creating a family of related objects.
    - Products: Classes that implement a common interface or extend a common base class within a family.
- Problem Solved:
    - Addresses the challenge of creating complex systems with multiple interrelated objects.
    - Ensures that objects created by a factory are consistent and compatible with each other.
    - Promotes scalability by allowing the addition of new families of objects.