**Important design patterns you should be familiar with from an interview perspective:**

1. **Singleton Pattern**: Ensures a class has only one instance and provides a global point of access to it.
2. **Factory Pattern**: Defines an interface for creating objects but lets subclasses alter the type of objects that will be created.
3. **Builder Pattern**: Allows for the creation of complex objects step by step. It separates the construction of a complex object from its representation.
4. **Observer Pattern**: Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
5. **Decorator Pattern**: Allows behavior to be added to individual objects, either statically or dynamically, without affecting the behavior of other objects from the same class.
6. **Adapter Pattern**: Allows incompatible interfaces to work together by wrapping an object around an interface to convert its methods into the expected interface.
7. **Strategy Pattern**: Defines a family of algorithms, encapsulates each one, and makes them interchangeable. It lets the algorithm vary independently from clients that use it.
8. **Facade Pattern**: Provides a unified interface to a set of interfaces in a subsystem. It defines a higher-level interface that makes the subsystem easier to use.
9. **Composite Pattern**: Composes objects into tree structures to represent part-whole hierarchies. It lets clients treat individual objects and compositions of objects uniformly.
10. **Template Method Pattern**: Defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure.
11. **Iterator Pattern**: Provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
12. **Chain of Responsibility Pattern**: Allows an object to pass a request along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it along the chain.
13. **Proxy Pattern**: Provides a placeholder for another object to control access to it. It allows you to create a substitute or placeholder for another object.