# FAQs Design Principles and Patterns

| | |
|---|---|
| Q. 1 | What are Design Patterns? |
| Ans: | Design Pattern is a solution to a problem in a context.<br>Pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.”<br>Design Patterns are “reusable solutions to recurring problems that we encounter during software development.” |
| Q. 2 | Categories of Design Patterns |
| Ans: | Design Patterns can be broadly classified as:<br>▪ Fundamental patterns<br>▪ Creational patterns<br>▪ Structural patterns<br>▪ Behavioral Patterns |
| Q. 3 | What are the advantages of Design Patterns |
| Ans: | Design patterns allows a designer to be more productive and the resulting design to be more flexible and reusable.<br>Design patterns make communication between designers more efficient |
| Q. 4 | What are the Drawbacks of Design Patterns? |
| Ans: | Listed below are some of the drawbacks of design patterns:<br>▪ Patterns do not allow direct code reuse.<br>▪ Patterns are deceptively simple.<br>▪ Design might result into Pattern overload.<br>▪ Patterns are validated by experience and discussion rather than by automated testing. |
| Q. 5 | What is a Fundamental Design Pattern? |
| Ans: | Fundamental Patterns are fundamental in the sense that they are widely used by other patterns or are frequently used in many programs. |
| Q. 6 | Usage of Interface Pattern |
| Ans: | Interfaces are “more abstract'' than classes since they do not say anything at all about representation or code. All they do is describe public operations.<br>You can keep a class that uses data and services provided by instances of other classes independent of those classes by having it access those instances through an interface. |
| Q. 7 | Usage of Abstract Superclass |
| Ans: | Abstract superclass ensures consistent behavior of conceptually related classes by giving them a common abstract superclass.<br>Forces:<br>▪ You want to ensure that logic common to the related classes is implemented consistently for each class.<br>▪ You want to avoid the runtime and maintenance overhead of redundant code.<br>▪ You want to make it easy to write related classes. |
| Q. 8 | What is a Creational Design Pattern? |

| | |
|---|---|
| Ans: | Creational Patterns provide guidelines on creation, configuration, and initialization for objects. “Decisions typically involve dynamically deciding which class to instantiate or which objects an object will delegate responsibility to.” |
| Q. 9 | What is Factory Method pattern? |
| Ans: | Helps to model an interface for creating an object. Allows subclasses to decide which class to instantiate. Helps to instantiate the appropriate subclass by creating the right object from a group of related classes. Promotes loose coupling. |
| Q. 10 | Pros of Factory Method Pattern |
| Ans: | Loose Coupling<br>• Factory Method eliminates the need to bind application classes into client code.<br>Object Extension<br>• It enables classes to provide an extended version of an object.<br>Appropriate Instantiation<br>• Right object is created from a set of related classes. |
| Q. 11 | Cons of Factory Method |
| Ans: | Two major varieties<br>• The creator superclass may or may not implement the factory method<br>• In any case, the superclass needs to be subclassed to create a concrete product<br>Parameterized factory methods<br>• Multiple kinds of products can be created by passing parameters to factory methods<br>• There is no standardization on this aspect |
| Q. 12 | What is Singleton Pattern? |
| Ans: | The Singleton pattern ensures that only one instance of a class is created. All objects that use an instance of that class use the same instance. |
| Q. 13 | Pros on Singleton Pattern? |
| Ans: | It provides controlled access to unique instance<br>It provides reduced namespace |
| Q. 14 | Cons on Singleton Pattern? |
| Ans: | If the Singleton class has subclasses, then ensuring unique instance is a challenge<br>Ensuring a unique instance always is a challenge<br>It poses difficulty in unit testing as the Singletons introduce global state into the application |
| Q. 15 | What is Builder Pattern? |
| Ans: | The *Builder* pattern is implemented a few times in the .NET framework.<br>A couple of note are the connection string builders.<br>Connection strings can be a picky thing and constructing them dynamically at runtime can sometimes be a pain. Connection String Builder classes demonstrate the builder pattern. |