# Design Patterns

## *Interview Questions*

# Q1: What are the main categories of Design Patterns?

## Answer

The Design patterns can be classified into three main categories:
- Creational Patterns
- Behavioral Patterns
- Functional Patterns

# Q2: What is Singleton pattern?

## Answer

Singleton pattern comes under creational patterns category and introduces a single class which is responsible to create an object while making sure that only single object gets created.


## Q3: What is a pattern?

### Answer

Patterns in programming are like recipes in cooking. They are not ready dishes, but instructions for slicing and dicing products, cooking them, serving them and so forth.
Pattern content As a rule, a pattern description consists of the following:

- a problem that the pattern solves;
- motivation for solving the the problem using the method suggested by the pattern;
- structures of classes comprising the solution;
- a description of the nuances of pattern implementation in various contexts; relations with other patterns.

## Q4: What is Design Patterns and why anyone should use them?

### Answer

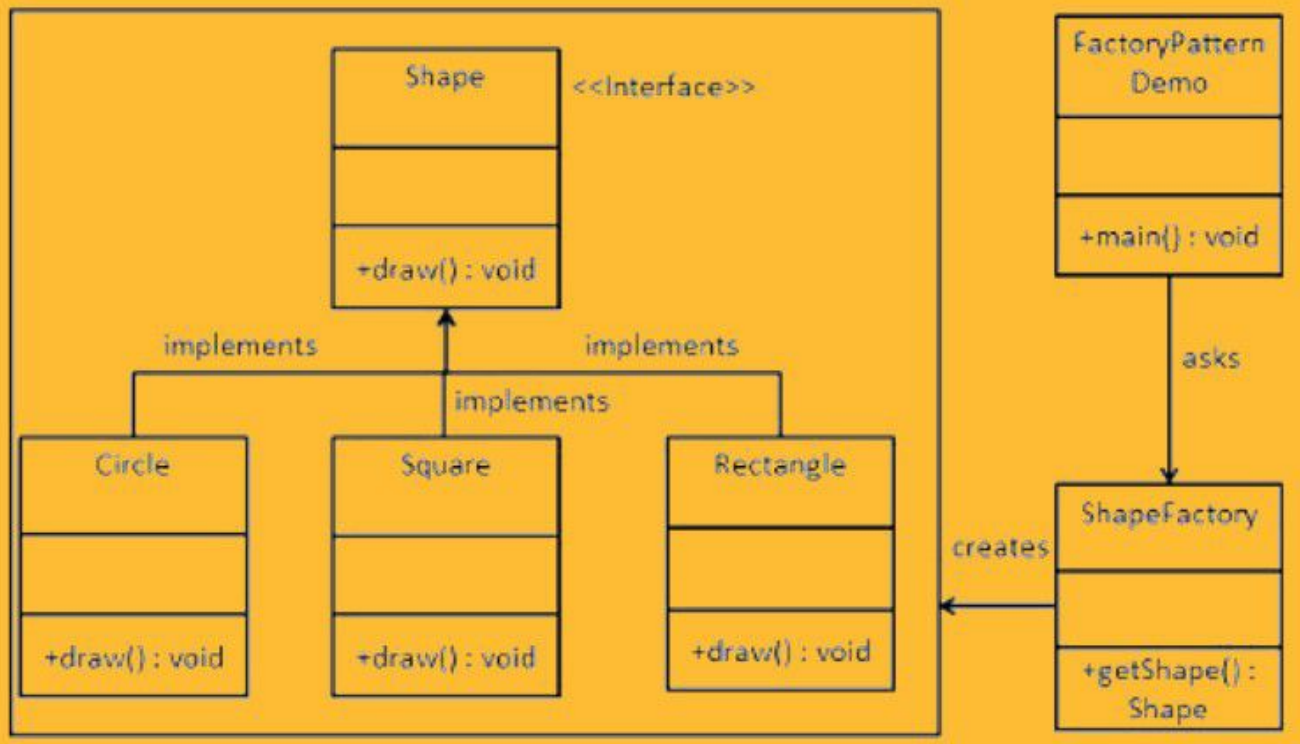
Design patterns are a well-described solution to the most commonly encountered problems which occur during software development.
Design pattern represents the best practices evolved over a period of time by experienced software developers. They promote reusability which leads to a more robust and maintainable code.

SWIPE

# Q5: What is Factory pattern?

## Answer

Factory pattern is one of most used design pattern and comes under creational patterns category. In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.




SWIPE

# Q5: What is Factory pattern?

Pro's:

Allows you to hide implementation of an application seam (the core interfaces that make up your application)
Allows you to easily test the seam of an application (that is to mock/stub) certain parts of your application so you can build and test the other parts
Allows you to change the design of your application more readily, this is known as loose coupling

Con's:

Makes code more difficult to read as all of your code is behind an abstraction that may in turn hide abstractions.
Can be classed as an anti-pattern when it is incorrectly used, for example some people use it to wire up a whole application when using an IOC container, instead use Dependency Injection.

SWIPE

# Q6: What is Filter pattern?

## Answer

Filter pattern or Criteria pattern is a design pattern that enables developers to filter a set of objects using different criteria and chaining them in a decoupled way through logical operations. This type of design pattern comes under structural pattern as this pattern combines multiple criteria to obtain single criteria.

Filter design pattern is useful where you want to add filters dynamically or you are implementing multiple functionalities and most of them require different filter criteria to filter something. In that case instead of hard coding the filters inside the functionalities, you can create filter criteria and re-use it wherever required.

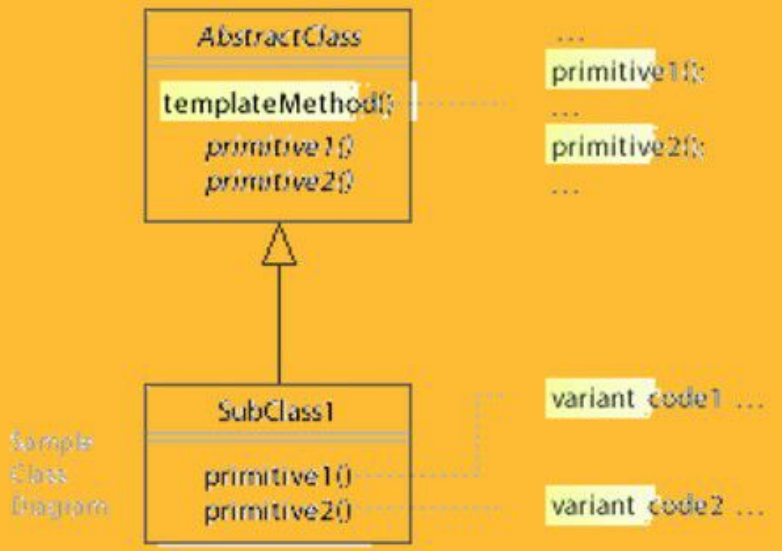
```
List<Laptop> laptops = LaptopFactory.manufactureInBulk();
AndCriteria searchCriteria = new AndCriteria(
  new HardDisk250GBFilter(),
  new MacintoshFilter(),
  new I5ProcessorFilter());
List<Laptop> filteredLaptops = searchCriteria.meets(laptops);
```

SWIPE

# Q7: What is Template pattern?

## Answer

In Template pattern, an abstract class exposes defined way(s)/template(s) to execute its methods. Its subclasses can override the method implementation as per need but the invocation is to be in the same way as defined by an abstract class. This pattern comes under behavior pattern category.




SWIPE

# Q8: What is Inversion of Control?

## Answer

Inversion of control is a broad term but for a software developer it's most commonly described as a pattern used for decoupling components and layers in the system.
For example, say your application has a text editor component and you want to provide spell checking. Your standard code would look something like this:

```
public class TextEditor {

    private SpellChecker checker;

    public TextEditor() {
        this.checker = new SpellChecker();
    }
}
```

What we've done here creates a dependency between the TextEditor and the SpellChecker. In an IoC scenario we would instead do something like this:

```
public class TextEditor {

    private SpellChecker checker;

    public TextEditor() {
        this.checker = new SpellChecker();
    }
}
```

You have inverted control by handing the responsibility of instantiating the spell checker from the TextEditor class to the caller.

```
SpellChecker sc = new SpellChecker; // dependency
TextEditor textEditor = new TextEditor(sc);
```

# Q 9 : Name types of Design Patterns?

## Answer

Design patterns can be classified in three categories: Creational, Structural and Behavioral patterns.

- Creational Patterns - These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new opreator. This gives program more flexibility in deciding which objects need to be created for a given use case.

- Structural Patterns - These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.

-Behavioral Patterns - These design patterns are specifically concerned with communication between objects.

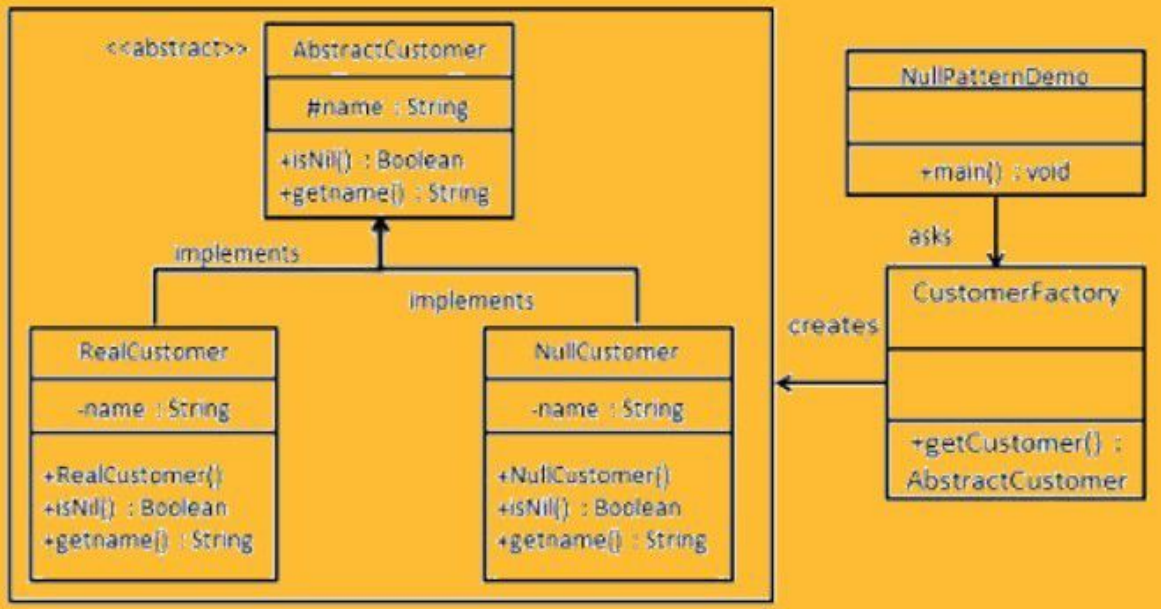
SWIPE

# Q10: What is Null Object pattern?

## Answer

In Null Object pattern, a null object replaces check of NULL object instance. Instead of putting if check for a null value, Null Object reflects a do nothing relationship. Such Null object can also be used to provide default behaviour in case data is not available.

In Null Object pattern, we create an abstract class specifying various operations to be done, concrete classes extending this class and a null object class providing do nothing implementation of this class and will be used seamlessly where we need to check null value.

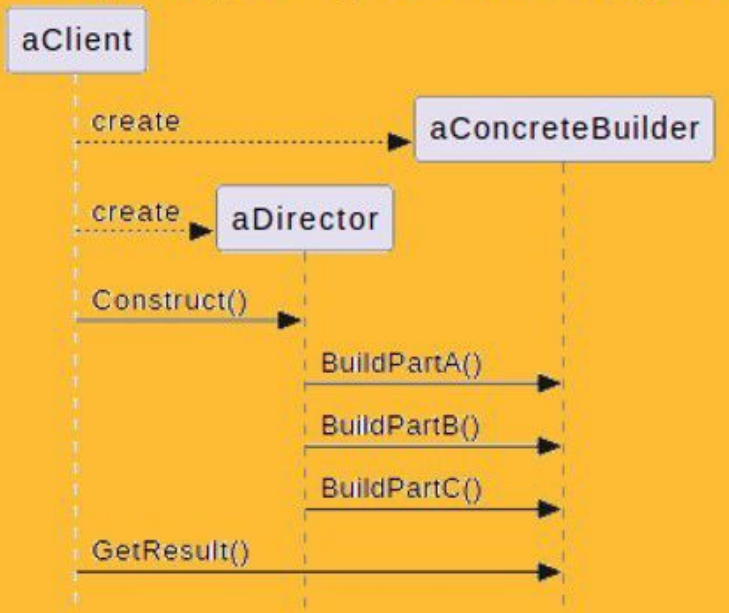



SWIPE

# Q11: What is Builder pattern?

## Answer

Builder pattern builds a complex object using simple objects and using a step by step approach. This builder is independent of other objects.




The Director class is optional and is used to make sure that the building steps are executed in the right order with the right data by the right builder. It's about validation and delegation.

Builder/Director pattern's steps invocations could be semantically presented by method chaining or so called Fluent Interface syntax.

## Q12: Can we create a clone of a singleton object?

### Answer

Yes, we can but the purpose of Singleton Object creation is to have single instance serving all requests.

## Q13: What is Dependency Injection?

### Answer

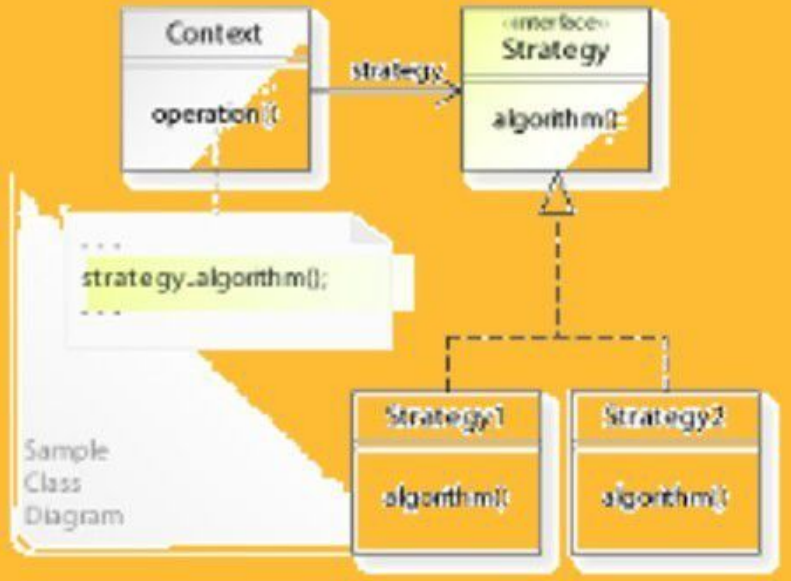
Dependency injection makes it easy to create loosely coupled components, which typically means that components consume functionality defined by interfaces without having any first-hand knowledge of which implementation classes are being used.
Dependency injection makes it easier to change the behavior of an application by changing the components that implement the interfaces that define application features. It also results in components that are easier to isolate for unit testing.

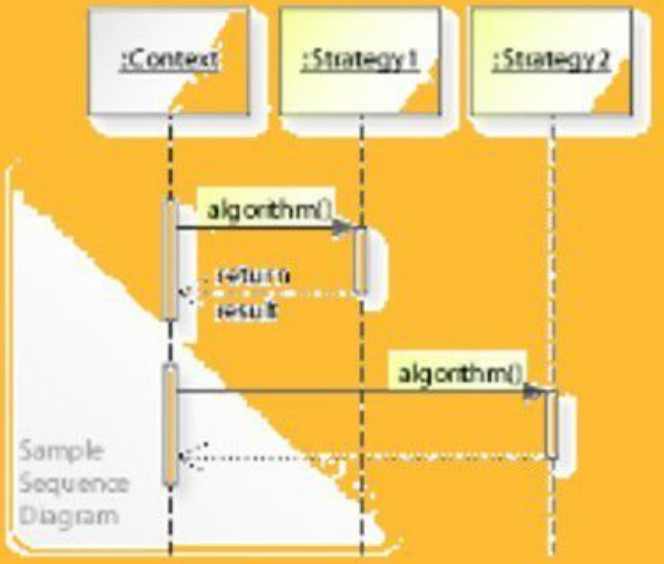
SWIPE

# Q14: What is Strategy pattern?

## Answer

In Strategy pattern, a class behavior or its algorithm can be changed at run time. This type of design pattern comes under behavior pattern.
In Strategy pattern, we create objects which represent various strategies and a context object whose behavior varies as per its strategy object. The strategy object changes the executing algorithm of the context object.

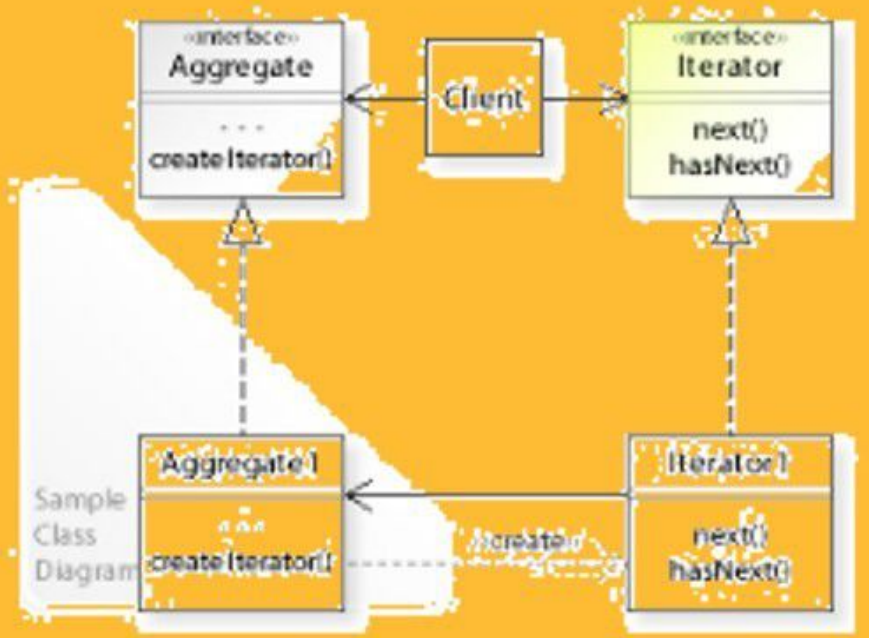


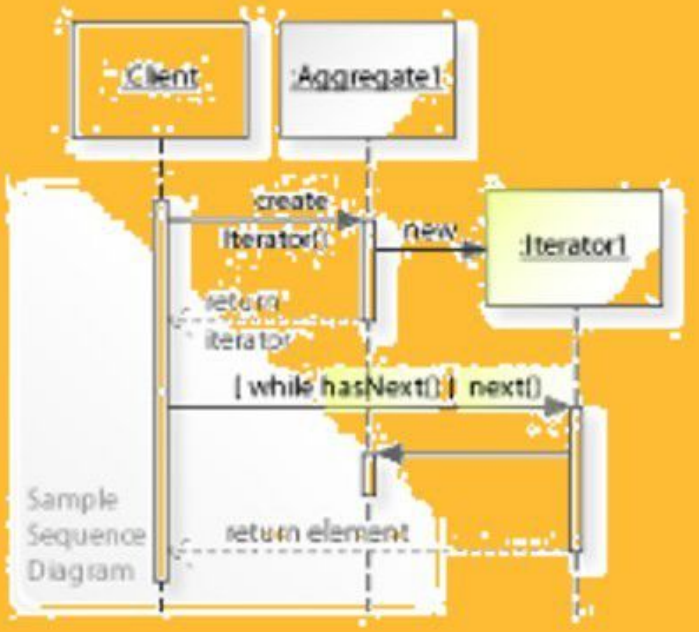




# Q15: What is Iterator pattern?

## Answer

Iterator pattern is very commonly used design pattern in Java and .Net programming environment. This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation. Iterator pattern falls under behavioral pattern category.




SWIPE

# Q16: What is Proxy pattern?

## Answer

In proxy pattern, a class represents functionality of another class. This type of design pattern comes under structural pattern.
In proxy pattern, we create object having original object to interface its functionality to outer world.