**Low Level Design (LLD) or Design Patterns**

**Design Pattern**

* Design Patterns are template – based reusable solutions that solve common problems of software development.
* These problems include repetitive code, redundant functions & logic etc.
* Design Patterns help the developers to save considerable effort & development time.
* Design Patterns are commonly used in object – oriented software products by incorporating best practices & promoting reusability for developing robust code.
* **Gang of Four (GOF):** In 1994, four authors Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides published a book titled “Design patterns – Elements of Reusable Object – Oriented Software” which initiated the concept of Design Pattern in Software Development.

**Advantages of Java Design patterns**

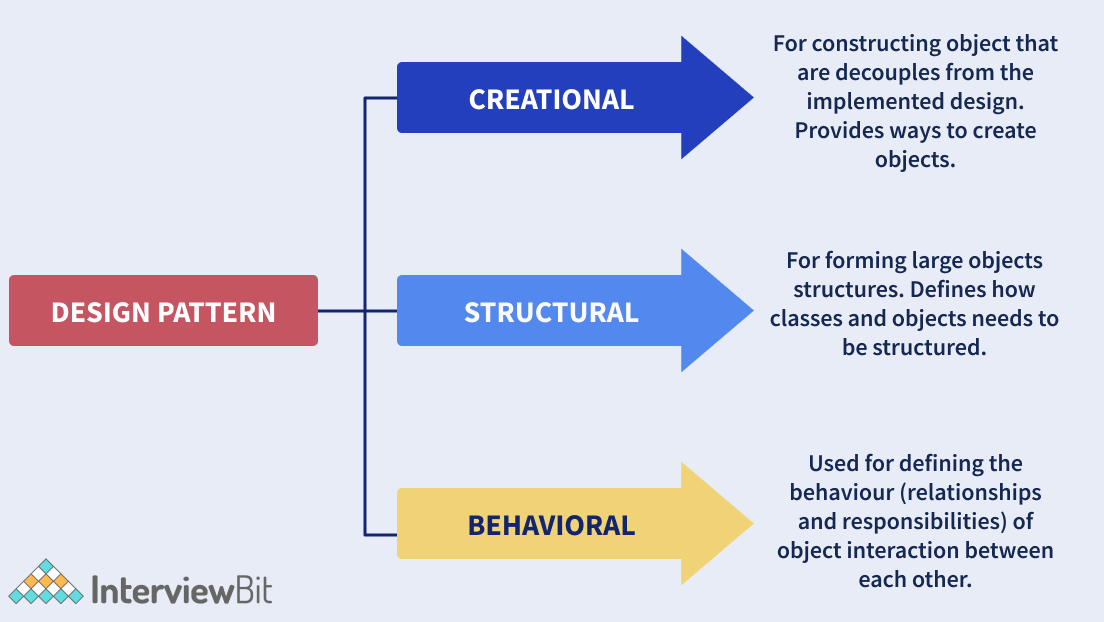
* They are reusable & can be used in multiple projects.
* They provide template solutions for defining system architecture.
* They provide transparency to software design.
* They are well – tested & proven means of developing robust solutions effortlessly.

**Question: How can a design pattern be described?**

Ans: To describe a design pattern, we follow below things

* Define a pattern name & what classification of design pattern that pattern would fall to.
* Define a problem & what’s the corresponding solution
* What are the variations & language – dependent alternatives for the problem that needs to be addressed?
* What are the real – time use cases & the efficiency of the software that uses these patterns?

**3 Types of Design patterns**



1. **Creational Patterns**

* These patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator.
* This gives program more flexibility in deciding which objects need to be created for a given use case.
* There are 6 types of creational design patterns
  1. **Factory method pattern**
  2. **Abstract factory pattern**
  3. **Singleton pattern**
  4. **Prototype pattern**
  5. **Builder pattern**
  6. Object Pool pattern

1. **Structural Patterns**

* These patterns help in defining how the structures of classes & objects should be like for defining the composition between classes, interfaces & objects.
* This pattern simplifies the structure by identifying the relationships.
* There are 7 types of Structural design patterns
  1. **Adapter pattern**
  2. Bridge pattern
  3. Composite pattern
  4. **Decorator pattern**
  5. **Facade pattern**
  6. Proxy pattern
  7. Flyweight pattern

1. **Behavioral Patterns**

* These patterns help to define how the objects should communicate & interact with one another and still should be loosely coupled.
* There are 12 types of behavioral design patterns
  1. Chain of Responsibility pattern
  2. **Command Pattern**
  3. Interpreter pattern
  4. **Iterator pattern**
  5. Mediator pattern
  6. Memento Pattern
  7. **Observer Pattern**
  8. State Pattern
  9. **Strategy Pattern**

3.10 Template Pattern

3.11 Visitor Pattern

3.12 Null Object

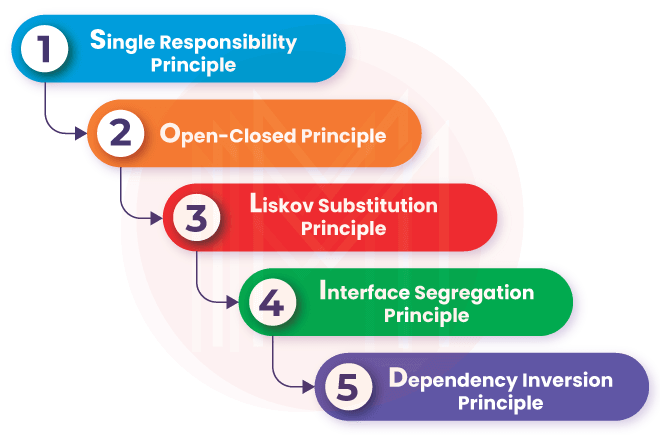
**Difference between Design Patterns & Design Principles**

|  |  |
| --- | --- |
| **Design Patterns** | **Design Principles** |
| Design patterns are the reusable template solutions used for common issues that can be customized as per the requirement.  These can be implemented efficiently, are safe to use & can be tested properly.  e.g., Creational Pattern, Structural Pattern & Behavioral Pattern | Design Principles are followed while designing software systems for any sort of platform by using a programming language.  e.g., **SOLID** principles are the design principles that are followed as guidelines to develop extensive, scalable & robust software systems. |

**Difference between Design Patterns & Algorithm**

**Ans**: Design Patterns & Algorithms are used to describe typical solutions to a problem. However the primary difference is that a design pattern offers a high – level description of solutions, while Algorithm outline a set of actions to achieve a goal.

**SOLID Principles**



Solid principles were the Object – Oriented principles introduced by Robert C. Martin in 2000. **SOLID** acronym goes:

* **S stands for SRP or Single Responsibility Principle**
* The single responsibility principle states that every class or module should be accountable & responsible for only one functionality. There should be one & only reason for changing any class.
* **O stands for OCP or Open – Closed Principle**
* The open closed principle states that every class is open for extension but closed for modification.
* Here we are allowed to extend the entities behavior by not modifying anything in the existing source code.

* **L stands for LSP or Liskov Substitution Principle**
* The Liskov substitution principle states that the objects can be replaced by the subtype instances without affecting the correctness of the program.
* **I stands for ISP or Interface Segregation Principleshivam**
* The interface segregation principle states that we can use as many interfaces specific to the client’s requirements instead of creating only one general interface.
* Clients should not be forced to implement the functionalities that they don’t require
* **D stands for DIP or Dependency Inversion Principle**
* The dependency inversion principle states that the high – level modules should not be dependent on the concrete implementations or the lower-level modules. Rather, they should stay dependent on the abstractions.

**What are some of the design patterns used in Java’s JDK library?**

Ans: Following design patterns used in JDK are

1. **Singleton pattern** is used in classes like Calender & Runtime
2. **Factory pattern** is used for methods like Integer.valueOf method in wrapper classes
3. **Decorator pattern** are used by the Wrapper classes
4. **Observer pattern** is used for handling event frameworks like awt, swing etc.

**What is Inversion of Control?**

* Inversion of control is a pattern used to decouple the dependencies between layers & components in the system.
* The Dependency – Injection (DI) pattern is an example of an IoC pattern that helps in removing dependencies in the code.

**Creational Design Pattern**

1. Factory design pattern / Factory Method design pattern

* Factory pattern belongs to creational design patterns.
* In factory method design patterh