Design Pattern, Elements & IT's Usage in Java

Designed by Rita Ganatra

What are Design Patterns?

- A design patterns are **well-proven solution** for solving the specific problem/task.
- Design patterns are reusable solution to commonly occurring problem within a given context in Software design. It's not a finished design that can be transformed directly into app. Rather, it is a **template** to solve a recurring design problem in code.
- Design patterns are **programming language independent strategies for solving the common object-oriented design problems.** That means, a design pattern represents an idea, not a particular implementation.
- By using the design patterns you can make your code more flexible, reusable and maintainable. Design patterns can speed up the development process by providing tested, proven development paradigms.

Now, a question will be arising in your mind what kind of specific problem? Let me explain by taking an example.

Problem **Given:**
Suppose you want to create a class for which only a single instance (or object) should be created and that single object can be used by all other classes.

Every design pattern has **some specification or set of rules** for solving the problems. What are those specifications, you will see later in the types of design patterns.

Design Pattern represent the best practices used by experienced object-oriented software developers.

Design Pattern is the most important part because java internally follows design patterns.

To become a professional software developer, you must know at least some popular solutions (i.e. design patterns) to the coding problems.
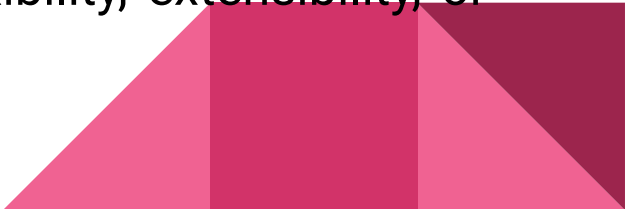
Elements of Design Pattern

A pattern has 4 essential elements:

- Pattern name - The **pattern name** is a handle we can use to describe a **design problem,** its solutions, and consequences in a word or two. Naming a pattern immediately increases our design vocabulary.

- Problem. - The **problem** describes **when to apply the pattern**. It explains the problem and its context. It might describe class or object structures.

- Solution. - The **solution** describes the **elements that make up the design**, **their relationships, responsibilities, and collaborations.** Instead, the pattern provides an abstract description of a design problem and how a general arrangement of elements (classes and objects).
- Consequences. - The **consequences** are the **results and trade-offs** of **applying the pattern.** The consequences for software often concern space and time trade-offs(understanding the **constraints** and **scope**, balancing your teammates decisions, and creating the "best" solution). They may address language and implementation issues as well. Since reuse is often a factor in object-oriented design, the consequences of a pattern include its **impact** on a system's flexibility, extensibility, or portability.

Advantage of design pattern:

1. They are **reusable** in multiple projects.
2. They provide the **solutions** that help to define the system architecture.
3. They capture the **software engineering experiences.**
4. They provide **transparency** to the design of an application.
5. They are **well-proved and testified solutions** since they have been built upon the knowledge and experience of expert software developers.
6. Design patterns don't guarantee an absolute solution to a problem. They provide **clarity to the system architecture and the possibility of building a better system.**

Usage of Design Pattern

Design Patterns have two main usages in software development.

- Common platform for developers

Design patterns provide a **standard terminology** and are specific to **particular scenario**. For example, a singleton design pattern signifies use of single object so all developers familiar with single design pattern will make use of single object and they can tell each other that program is following a singleton pattern.

- Best Practices

Design patterns have been evolved over a long period of time and they provide best solutions to certain problems faced during software development. Learning these patterns helps inexperienced developers to learn software design in an easy and faster way.

We must use the design patterns **during the analysis and requirement phase of SDLC**(Software Development Life Cycle).

Design patterns ease the analysis and requirement phase of SDLC by providing information based on prior hands-on experiences.

Thank You