Design Patterns Simplified Version

Posted on August 10, 2015 (http://simpletechtalks.com/design-patterns-simplified-version/) by admin (http://simpletechtalks.com/author/admin/)

Behavioural Patterns (http://simpletechtalks.com/tag/behavioural-patterns/) C++ (http://simpletechtalks.com/tag/c/) Creational Patterns (http://simpletechtalks.com/tag/creational-patterns/) Design Patterns (http://simpletechtalks.com/tag/design-patterns/) module (http://simpletechtalks.com/tag/module/) problem (http://simpletechtalks.com/tag/problem/) software (http://simpletechtalks.com/tag/software/) solution (http://simpletechtalks.com/tag/solution/) Structural Patterns (http://simpletechtalks.com/tag/structural-patterns/)

Design pattern describes solutions to a problem which arises while developing software for a specific condition. Design patterns are basically independent of any programming language. Lot of guys do the mistake of thinking it as an algorithm but its a common way to solve a generic problem which can be simple or complex.

Basically there are three different categories of design patterns depending on the problem they address:
1) Creational Patterns
2) Structural Patterns
3) Behavioural Patterns

Creational Patterns:
These design patterns solves the given problem by means of object creation mechanism. Normal way of object oriented solution to solve a problem sometimes create design problems and increasing complexity of the software. Thus, a set of design patterns are there to handle such kind of issues. Examples of this category of design patterns are as follows:

1) Factory Method (http://simpletechtalks.com/factory-design-pattern/) – This design pattern is used to create an instance from the family of various derived classes.
2) Abstract Factory Method (http://simpletechtalks.com/abstract-factory-design-pattern/) – This design pattern is used to create an instance from the several family of various derived classes.
3) Builder Method (http://simpletechtalks.com/build-design-pattern/) – This design pattern is used to separate construction and representation of an object to create different objects via same construction process.
4) Prototype Method (http://simpletechtalks.com/prototype-design-pattern/) – This design pattern is used clone mechanism to create a new object from a fully initialized object to minimize cost creation of new object.

5) Singleton Method (http://simpletechtalks.com/singleton-design-pattern/) – This design pattern is used to create one and only one object of a class of global context to provide co-ordination between various modules of a software.

Structural Patterns:
These design patterns solves the given problem by means of class and object composition to provide required functionality. Examples of this category of design patterns are as follows:

1) Adapter (http://simpletechtalks.com/adapter-design-pattern/) – This design pattern is used to adapt the interface of a class into another interface which other module expects.
2) Bridge (http://simpletechtalks.com/bridge-design-pattern/) – This design pattern is used to separate out the interface from its implementation logic to provide flexibility.
3) Composite (http://simpletechtalks.com/composite-design-pattern/) – This design pattern is used to create a tree structure of objects to provide hierarchies in the software.
4) Decorator (http://simpletechtalks.com/decorator-design-pattern/) – This design pattern is used to provide additional functionality to an objects dynamically.
5) Facade (http://simpletechtalks.com/facade-design-pattern/) – This design pattern is used to hide the complexities of the system by providing a simple interface to other module of the system.
6) Flyweight (http://simpletechtalks.com/flyweight-design-pattern/) – This design pattern is used to share properties of objects to reduce maintenance cost of objects.
7) Proxy (http://simpletechtalks.com/proxy-design-pattern/) – This design pattern is used to create an object for another object to hide the complexities and reduce maintenance cost of the object.

Behavioural Patterns:
These design patterns solves the given problem by means of communication between different class objects. Examples of this category of design patterns are as follows:

1) Chain of Responsibility (http://simpletechtalks.com/chain-of-responsibility-design-pattern/) – This design pattern is used to to handle passing of information from a chain of objects until one of them handles it.
2) Command (http://simpletechtalks.com/command-design-pattern/) – This design pattern is used to encapsulate a request as an object.
3) Interpreter (http://simpletechtalks.com/interpreter-design-pattern-explained-with-simple-example/) – This design pattern is used to provide a way to include grammar of a language to interpret sentences in the language.
4) Iterator (http://simpletechtalks.com/iterator-design-pattern-explained-with-simple-example/) – This design pattern is used to iterate through a similar collection of objects.
5) Mediator (http://simpletechtalks.com/mediator-design-pattern-explained-with-simple-example/) – This design pattern is used to encapsulate way of interaction between objects to simplify communication between classes.
6) Memento (http://simpletechtalks.com/memento-design-pattern-explained-with-simple-example/) – This design pattern is used to capture an object to expose object's internal state without violating encapsulation.
7) Observer (http://simpletechtalks.com/observer-design-pattern-explained-with-simple-example/) – This design pattern is used to handle notification to multiple objects due to one object modification.
8) State (http://simpletechtalks.com/state-design-pattern-explained-with-simple-example/) – This design pattern is used to change the functionality of an object whenever its state changes.
9) Visitor (http://simpletechtalks.com/visitor-design-pattern-explained-with-simple-example/) – This design pattern is used to provide a way to define a new functionality to a class object without changing its structure.
10) Strategy – This design pattern is used to encapsulate various algorithm inside class to allowing the change of algorithm randomly.

Posted in C++ (http://simpletechtalks.com/category/programming/c/) | Leave a comment (http://simpletechtalks.com/design-patterns-simplified-version/#respond)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment