C++

Design Patterns Simplified Version

by admin

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 – This design pattern is used to create an instance from the family of various derived classes.

2) Abstract Factory Method – This design pattern is used to create an instance from the several family of various derived classes.

3) Builder Method – This design pattern is used to separate construction and representation of an object to create different objects via same construction process.

4) Prototype Method – 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 – 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 – This design pattern is used to adapt the interface of a class into another interface which other module expects.

2) Bridge – This design pattern is used to separate out the interface from its implementation logic to provide flexibility.

3) Composite – This design pattern is used to create a tree structure of objects to provide hierarchies in the software.

4) Decorator – This design pattern is used to provide additional functionality to an objects dynamically.

5) Facade – 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 – This design pattern is used to share properties of objects to reduce maintenance cost of objects.

7) Proxy – 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 – This design pattern is used to to handle passing of information from a chain of objects until one of them handles it.
2) Command – This design pattern is used to encapsulate a request as an object.
3) Interpreter – This design pattern is used to provide a way to include grammar of a language to interpret sentences in the language.
4) Iterator – This design pattern is used to iterate through a similar collection of objects.
5) Mediator – This design pattern is used to encapsulate way of interaction between objects to simplify communication between classes.
6) Memento – This design pattern is used to capture an object to expose object's internal state without violating encapsulation.
7) Observer – This design pattern is used to handle notification to multiple objects due to one object modification.
8) State – This design pattern is used to change the functionality of an object whenever its state changes.
9) Visitor – 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.

TAGGED BEHAVIOURAL PATTERNS C++ CREATIONAL PATTERNS DESIGN PATTERNS MODULE PROBLEM SOFTWARE SOLUTION STRUCTURAL PATTERNS

PREVIOUS POST
Creating Menus and Pages in WordPress

NEXT POST
Build Design Pattern Explained With Simple Example: Creational Design Pattern Category

admin

View all posts by admin →

YOU MIGHT ALSO LIKE

Iterator Design Pattern explained with simple example

March 5, 2019

Interface Segregation Principle explained with simple example

February 15, 2019

Open/Closed design principle explained with simple example

February 12, 2019

Leave a Reply

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

COMMENT

NAME *

EMAIL *

WEBSITE

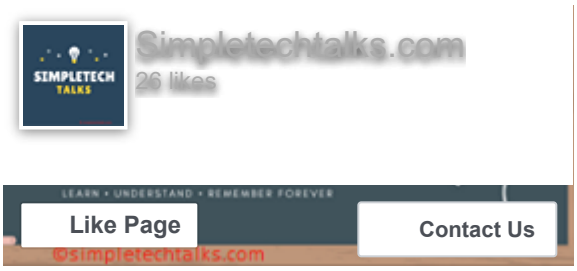
POST COMMENT

SEARCH

Search …

SEARCH

LIKE US



TRENDING

Difference between Copy constructor vs Move constructor

Difference between Copy assignment operator vs Move assignment operator

Program to detect and remove Loop in Linked List

Program to find Nth node from end of Linked List

Diameter Agents Simple Explanation

Program to calculate length of a Linked List

Heap sort explained with simple example

Diameter Protocol Message Structure

Graph and its basic implementation

Diameter Protocol AVP Structure

CATEGORIES

C

C++

C++11

Diameter

Dynamic Programming

Graph

Interview Questions

Linked List

Programming

Programming Questions

Questions & Answers

wordpress