CS F213
Object Oriented Programming

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Design Patterns

R1: Ch.15

And also refer to Class notes.

Content

1. What is a Design pattern
2. Some important patterns
3. Singleton pattern
4. Example

Introduction

SW analyst or engineer applies potential solutions to development problems, monitor the success or the failure and produces more effective solutions on the next occasion.

It is the nature of the software development that the same problems tend to occur.

Different developers may tend to expend great deal of time and effort on solving these problems from the first principles each time they occur.

The solutions they produce may not be the most appropriate that could be achieved.

This may result in SW systems that are inflexible, inefficient, difficult to maintain.

Design Patterns

- Patterns provide a means for capturing knowledge about problems and successful solutions in software development.
- Experience that has been gained in the past can be reused in similar situations.
- Thus reducing the effort in producing more resilient, more flexible and more efficient.

Frameworks

- Frameworks are partially completed systems which contain unfinished elements which can be completed according to the requirement.
- New elements can be added as per the need.
- Essentially, a framework is a reusable mini-architecture that provides structure and behavior common to all applications of this type.

Patterns Vs. Frameworks

- Patterns are more abstract and general than Frameworks.

  Pattern is a description of the way that a type problem can be solved.

  Pattern is not itself a solution.
- Patterns are more primitive than frameworks.
- A Framework can use several patterns, but a pattern can't employ a framework.
- Patterns normally don't have implementations.

Documenting patterns

Patterns may be documented using templates.

The pattern description should include the following.

- Name: A pattern should be given a meaningful name that reflects the knowledge embodied by the pattern. This may be a single word or a short phrase.
- Problem: This is the description of the problem that a pattern addresses (the intent of the problem). It should identify and describe the objectives to be achieved.
- Context: This represents the circumstances or preconditions under which it can occur.
- Forces: The forces embodied in a pattern are the constraints or issues that must be addressed by the solutions.
- Solution: Description of the static and dynamic relationships among the components of the pattern.

Types of design Patterns

Design patterns are classified according to their scope and purpose.

Three main categories of purpose a pattern can have are :

- Creational Pattern
- Structural pattern
- Behavioral pattern

Creational Patterns

A creational design pattern is concerned with the construction of object instances.

It separates the operation of an application from how its objects are created.

Singleton Pattern

A singleton class has exactly one instance.

All clients need to access a single shared instance of a class.

You want to ensure that no additional instances can be created accidentally.

Define a class with private constructor.

Supply a static method that returns a reference to the single object.

//Sample Java Code

```
public class DBManager
{
     private static DBManager dbm=null;
     public static DBManager getInstance()
     {     if(dbm==null)
           { dbm= new DBManager(); }
           return dbm;
    }
    private DBManager()
    {
    }
}
```

Description of Singleton Pattern

innovate achieve lead

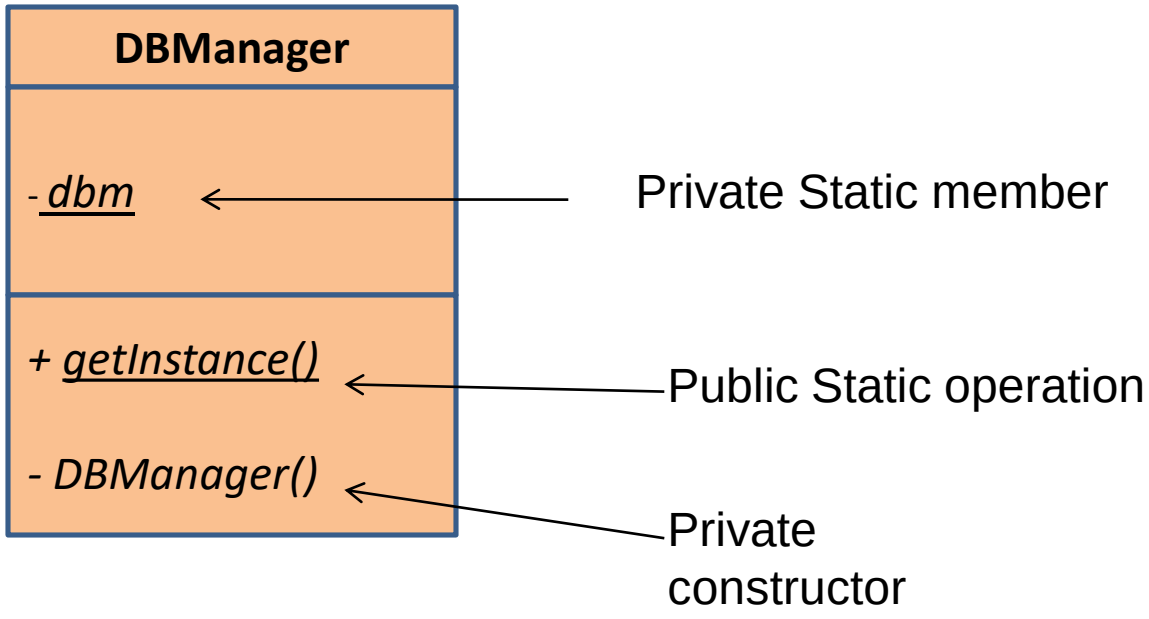
Name: Singleton

Problem: How can a class be defined that should have only one instance and that can be accessed globally within the application.

Context: In some applications it is important that a class have exactly one instance. Ex: A Database manager.

Forces: One approach to making an object globally accessible is to making it global variable, but in general it is not a good design solution as it violates encapsulation.

Solution: Create a class operation *getInstance()*, which when the class is first accessed, creates the relevant object instance and returns the object to the client.


DBManager
- dbm
+ getInstance()
- DBManager()
Private Static member
Public Static operation
Private constructor

Structural Patterns

A structural design pattern is concerned with the way the classes and objects are organized.

Structural patterns offer effective ways of using object-oriented constructs such as inheritance, aggregation and composition to satisfy particular requirements.

Ex: Composite pattern

Behavioral Patterns

Behavioral design patterns address the problems that arise when assigning responsibilities to classes and when designing algorithms. They focus on structural organization and also describe how objects communicate with each other.

Other patterns

There are total 23 Design patterns

- Iterator
- Adapter
- Factory Method
- Proxy
- Decorator
- Prototype etc.
- Observer

Architect Christopher Alexander

- *A Pattern Language* (1977)

"Gang of four"

- Erich Gamma
- Richard Helm
- Ralph Johnson
- John Vlissides

- *Design Patterns: Elements of Reusable Object-Oriented Software* (1995)

Summary

- What is a pattern
- Pattern templates
- Types of patterns with examples
- How and when to use patterns

Other important Object Oriented Programming Languages

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Smalltalk : was the first general purpose object-oriented programming language. It is a pure dynamically-typed object-oriented language. Smalltalk supports a uniform object model.

Everything a programmer deals with is an object including primitive types (such as numbers) and user-defined types.

Smalltalk supports full inheritance.

Smalltalk does not support multiple inheritance.

C++ was developed at Bell Labs by Bjarne Stroustrup (1979). It was designed mainly for systems programming, extending the C programming language. C++ is an object-oriented version of C.

It is a statically-typed object-oriented programming, exception handling,.

C++ is not a pure object oriented languages, because both procedural and objected-oriented development.

It provides multiple inheritance and exception handling.

It does not provide garbage collection

Eiffel was developed in 1985 as a pure object-oriented language.

Multiple inheritance is permitted in Eiffel.

Ada is a statically typed and object-oriented extended from Pascal and other languages.

Ruby is an object-oriented scripting language developed in 1993 by Matsumiko Yukihiro. I

t is similar in purpose to python or Perl.

Ruby has a pure object-oriented support as it does not allow functions.

All methods must belong to some class. Ruby only supports single inheritance.

C#: is an OOP language part of the .NET framework.

C# is not a pure OOPLs since it encompasses functional programming in addition to the object-oriented paradigm.

It has an object-oriented syntax based on C++ and is heavily influenced by Java. In some communities it is thought of as Microsoft's version of Java.

Like Java, it has garbage collection and it is compiled to an intermediate language, which is executed by the runtime environment known as Common Language Runtime (CLR) which is similar to the JVM.

The C# supports conception of class and instances, as well as inheritance and polymorphism.

Python is an object-oriented scripting language developed in 1990 by Guido Van Rossum.

It has become very popular in recent years due to its application in the internet domain.

Python allows both procedural and objected-oriented development.

Python allows multiple inheritance.