

Contents

Obsah

1 Design Patterns	1
2 Distributed Object Computing	3

1 Design Patterns

Design Patterns

Design pattern is a general repeatable solution to a comonly occuring problem in software design.

- It's not a finished design. It is a description for how to solve a problem.
- It is like algorithm (that solves computational problem) – but solves *design problem*.

Design Patterns Classification

Design Patterns are divided into many types:

- Creational Patterns – deal with object creation mechanisms.
- Structural Patterns – ease the design by identifying a simple way to realize relations between entities.
- Behavioral Patterns – identify common communication patterns between objects.

Factory Method Pattern

Factory Method Pattern is a Creational Pattern that deals with the problem of creating object without specifying the exact class of object that will be created.

Factory method is a static method that returns new implemented object.

```
class Complex {
    public static Complex fromCartesian(double re, double im){
        return new Complex(re, im);
    }
    public static Complex fromPolar(double mod, double ang) {
        return new Complex(mod*cos(ang), mod*sin(ang));
    }
    private Complex(double re, double im) {
        ...
    }
}
```

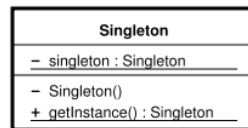
```

    Complex c=Complex.fromPolar(1, pi);
}

```

Singleton Pattern

Singleton Pattern is a Creational Pattern that is used to restrict instantiation of a class to *one* object.



```

class Singleton {
    private Singleton() {} //private constructor!!!
    private final static Singleton INSTANCE;
    public static Singleton getInstance() {
        if (INSTANCE==null) {
            INSTANCE=new Singleton();
        }
        return INSTANCE;
    }
}

```

Adapter Pattern

Adapter Pattern is a Structural Pattern that 'adapts' one interface for a class into one that client expects.

```

class DList<T> { //Double-Linked List
    public void insertHead(T o) { ... }
    public void insertTail(T o) { ... }
    public T removeHead() { ... }
    public T removeTail() { ... }
    public int getNumItems() { ... }
    ... constructors and other methods ...
}

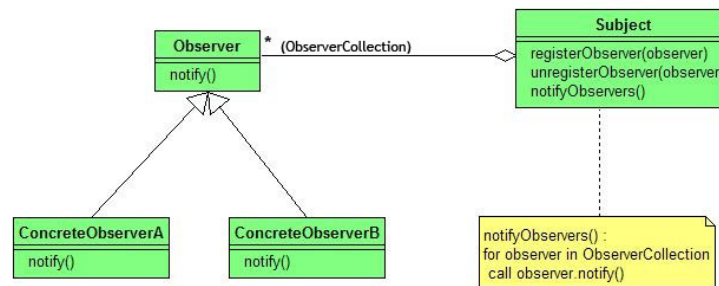
interface Stack<T> {
    void push(T o);
    T pop();
    bool isEmpty();
}

```

```
//Double-Linked List implemented by Stack
class DListImpStack<T> implements Stack<T> {
    private DList<T> list=new DList<T>();
    public void push(T o) {
        list.insertTail(o);
    }
    public T pop() {
        return list.removeTail();
    }
    public bool isEmpty() {
        return list.getNumItems()==0;
    }
}
```

Observer Pattern

Observer Pattern is a Behavioral Pattern that is used to observe the state of an object.



Further Reading on Design Patterns

- [http://en.wikipedia.org/wiki/Design_pattern_\(computer_science\)](http://en.wikipedia.org/wiki/Design_pattern_(computer_science))
- <http://objekty.vse.cz/Objekty/Vzory>
- Fowler, Martin. *Patterns of Enterprise Application Architecture*.
- Gamma, Erich. *Design Patterns: Elements of Reusable Object-Oriented Software*.

2 Distributed Object Computing

Distributed Object Computing

- Programs can be written as distributed. Some offer services to others.

- Services can be distributed over network.
- Services and programs can be written in different programming languages.

Common things

- IDL – Interface Definition Language
 - Definition of classes that will be exported by service provider.
 - It's language independent.
- IDL file is compiled into target language skeleton file.
- Service connects to IIOP (Internet Inter-ORB Protocol) server or something similar (ORPC, JRMP).
- Client connects to server with request and server forwards this request to service provider.
- Response and exceptions goes through server as well.

Further Reading on Distributed Object Computing

- <http://my.execpc.com/~gopalan/misc/compare.html>
- <http://en.wikipedia.org/wiki/CORBA>
- http://en.wikipedia.org/wiki/Distributed_Component_Object_Model
- http://en.wikipedia.org/wiki/Java_remote_method_invocation