Design Patterns

Table Of Contents

[1. Adapter Design Pattern 3](#_Toc521076685)

[2. Proxy Deign Pattern 3](#_Toc521076686)

[3. Factory Deign Pattern 4](#_Toc521076687)

[4. Builder Pattern 6](#_Toc521076688)

[5. Singleton Design Pattern 7](#_Toc521076689)

*Design Patterns*

# Adapter Design Pattern

The motive of the adapter pattern is to convert the interface of one class to another. In other words, it is used to provide the interface according to client requirement while using the services of a class with a different interface.

*Advantages:*

* It allows two or more previously incompatible objects to interact.
* It allows code re usability.

Example:

1. Target Interface: CustomerServiceInfo

Method : getActiveService()

1. Adoptee class : CustomerProfileInfo

Two Fields : String activeServicename;

String accountNumber;

1. Adapter Class : customerServiceInfoImpl extends CustomerProfileInfo implements CustomerServiceInfo

Signature :

String getActiveService(){

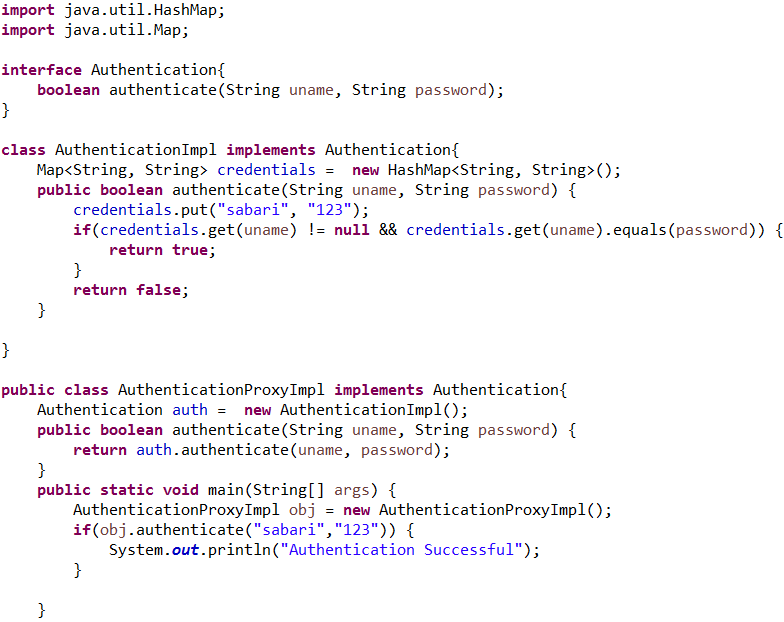
Return activeServicename;

}

Link: <https://www.javatpoint.com/adapter-pattern>

# Proxy Deign Pattern

The main intent of the design pattern is to provide a place holder or surrogate for another object enforcing controlled access to it.



# Factory Deign Pattern

This design pattern is used when we have a super class having multiple sub class and based on the input we are to instantiate any one of the sub class object.

For example, if we have an interface (Image Processor) the provides methods to process images and we have various implementation of the Image Processor to process jpeg format, gif format, bitmap format. Based on the format of the image we must instantiate the corresponding object and then perform the processing.



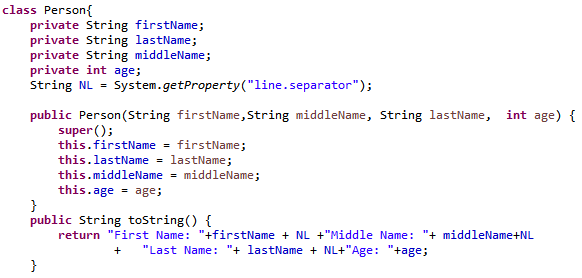


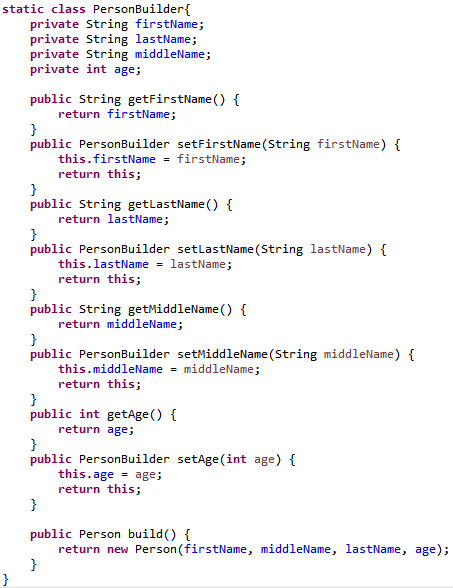
# Builder Pattern

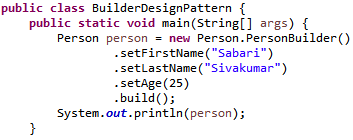
It is a creational design pattern that helps in building objects having more number of fields. The implementation of builder design pattern will have a static nested class having declaration of all the fields contained in the enclosing class along with setter’s method.

The setters contained in the class will have a return statement to return the reference to the current object being constructed (return this).

The only drawback of the builder design patter is that the duplicate code is written for declaring the fields along with the getters and setters.







# Singleton Design Pattern

Considering reflection, Cloneable and serialization the singleton design pattern implementation can be achieved by means of enumeration.

