Factory Method Design Pattern

Deepan Chakravarthi Padmanabhan

1 Description

Give a brief description. Maybe with a non code metaphor example is good. Be creative.

In software engineering, a design pattern is a generic repeatable solution to a frequently occurring issue in software design. There are three types of design pattern.

- Creational design pattern
- Behavioural design pattern
- Structural design pattern

In general, creational design pattern are concerned with the way of instantiating a class. There are two broader classification of creational design pattern: class-creational and object-creational design pattern. The class creation utilizes inheritance and object creation pattern uses delegation effectively to get the job done. There are 6 types of creational design pattern namely,

- Factory Method design pattern
- Abstract Factory design pattern
- Builder design pattern
- Object Pool design pattern
- Prototype design pattern
- Singleton design pattern

In this report, One of the types of creational design pattern called Factory method design pattern is discussed in detail.

The most used design pattern in Java is **Factory method design pattern**. In this technique, we create objects without exposing the creation logic to the client and refer newly created objects using a common interface

Factory Method is the standard way to create objects but it is unnecessary if the class that is instantiated never changes, or instantiation takes place in an operation that subclasses can easily override (such as an initialization operation).

Factory Method makes a design more customizable and only a little more complicated. In factory method design pattern, define an interface or abstract class for creating an object and the subclasses decide the class to be instantiated.

Impact:

This design pattern solves two problems: How can an object be created so that subclasses can redefine which class to instantiate? and How can a class defer instantiation to subclasses?

Factory method solves the problems by defining a separate operation (factory method) for creating an object and by creating an object by calling a factory method.

Example: In real life circumstances, a particular classification method provides better results for a specific problem. In this example, a ML classification testing system involving numerous methods- Linear SVC, RBF Kernel and Polynomial SVC has been discussed. The tester or the user does not require any knowledge about the implementation of the system. For instance, the object and classifier utilised at any point of the system, in order to provide this abstraction factory method design pattern is necessary. In addition, the factory method design pattern favours easy implementation of numerous classification methods in future without the use of duplicating the code. For example, in future if the system requires a Naive Bayes Classifier creating a corresponding Naive Bayes class and instantiating it by Factory method will help in implementing the S.O.L.I.D principles- open for update and closed for changes. Similarly, various design patterns help in solving numerous generic problems faced in a software development cycle. This case is illustrated with a UML diagram in the next section.

2 UML Diagram

Does the Design pattern have a UML, can you describe

Yes, Factory Method Design pattern has a UML diagram. The UML diagram for the implementation of factory design pattern is given below.

UML Diagram:

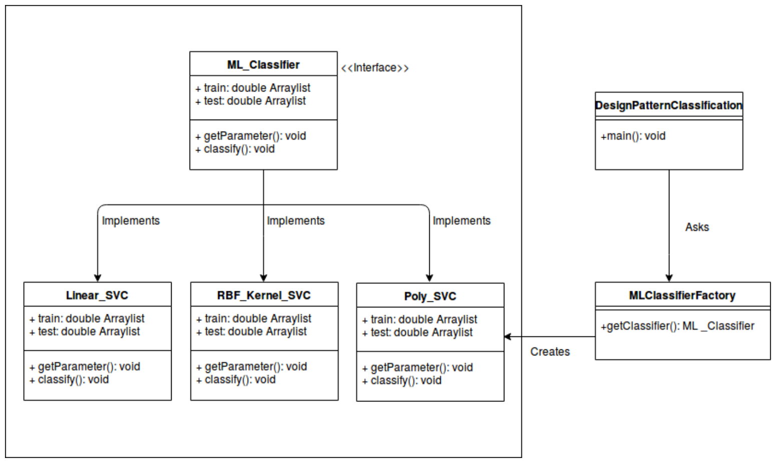

Figure 1: UML-Factory Method Design Pattern

Explanation:

The interface ML_Classifier has the functions declared as getParameter and classify. The classification process examples given are Linear_SVC, RBF_Kernel_SVC and Poly_SVC. All these methods implements the interface ML_Classifier. The factory_method class has a function called Setparameter which receives a string or utilises a methodology to create a particular object for the classification process. This factory method is called from the main program.

Thus objects are created in the factory depending on the requirement of the system without exposing the creation logic to the client. The design pattern will be coded and explained in the upcoming weeks.

3 Example Usage

Explain with an example where you can use the design pattern. Give an example of how the code looks without the design pattern. How the code looks with the design pattern. Does the design pattern adheres to any of the S.O.L.I.D guidelines.

The FactoryDesign/src folder contains the implementation of the above given example. DesignPatternClassification.java file holds the objects for Linear SVC, RBF Kernel SVC and Polynomial SVC created by a Factory method design pattern. The objects are created of type ML classifier and the user is unaware of the classifier object attributes or methods binding to it. This brings about an abstraction and aids S.O.L.I.D principle of "open for extension and closed for modification".

However, the file ConventionalClassification.java file explicitly creates object for the specific classifier object and doesnot support abstraction providing direct access to classifier parameter to the user. This highly inhibits the security and brings about code redundancy in future while implementing other classification type. In former method, a separate implementation of the new classifier can be done and an object could be created for its implementation avoiding redundancy and supporting secure implementation.

The code is provided in the FactoryDesign folder. The details of each file is given below:

- ML_Classifer.java - Interface
- Linear_SVC.java - Class for specific classification task. This holds a structural implementation (not functionally precise provided for an overview) for Linear SVC classification method.
- RBF_Kernel_SVC.java - Class for specific classification task. This holds a structural implementation (not functionally precise provided for an overview) for RBF Kernel SVC classification method.
- Poly_SVC.java - Class for specific classification task. This holds a structural implementation (not functionally precise provided for an overview) for Polynomial SVC classification method.
- MLClassifierFactory.java - Factory method for creating objects for each classification method. The objects returned are of the type ML_Classifier and does not provide explicit details of the classification task objects.
- DesignPatternClassification.java - Creates objects for each task using the Factory method design pattern as explained in this document.

- ConventionalClassification.java - This part of the program does not implement the Factory method design pattern and it is provided to illustrate the difference.

References:

1. SOLID principle, Available on: https://en.wikipedia.org/wiki/SOLID , Viewed on: 20.12.2018
2. Factory Method Design Pattern - Tutorials point, Available on: https://www.tutorialspoint.com/design_pattern/factory_pattern.htm , Viewed on: 20.11.2018
3. Factory Method Design Pattern - Refractoring GUru, Available on: https://refactoring.guru/design-patterns/factory-method , Viewed on: 20.11.2018