2014

Design Patterns: Participant Guide

JP Morgan Chase & Co.
Technology Training

Contents

Lab 1. Abstract Factory Design Pattern

Time Needed: (15min)

Skills Learned

At the end of this exercise, you will be able to:

- Identify the situations in which the Abstract Factory design pattern can be used to develop an application.

Problem Statement

Assume that there is an application that stores Devise objects. These objects may be stored in a text file or in a relational database. The code used to access these object remains ignorant of how the objects are being stored. As a developer, you have to design an interface that allows an application to access stored objects of the type Device, irrespective of the storing mechanism being employed.

Application Design

To develop the interface, you have to use the Abstract Factory design pattern by using the following guidelines:

- The interface application should have one abstract factory class
- The application should have two concrete implementations of the factory interface defined, one for each kind of data store: text file and a database.

Solution Instructions

1.	There are no restrictions on what the field names and method names should be as long as they are named logically.

Lab 2. Adapter Design Pattern

Time Needed: (15min)

Skills Learned

At the end of this exercise, you will be able to:

- Know where and when Adapter Design Pattern is implemented.

Problem Statement

A real life example, we can think of an adapter that fits between a wall socket that produces 240V and a plug that expects 120V.

Test Design

You need to implement Adapter Design Pattern using the following guidelines:

- Create an adapter class that converts 240V to 120V.
- Implement the Adapter pattern using any of the 2 approaches, (Class Adapter, Object Adapter) however both these approaches produce same result.

Solution Instructions

1.	There are no restrictions on what the field names and method names should be as long as they are named logically.

Lab 3. Command Design Pattern

Time Needed: (15min)

Skills Learned

At the end of this exercise, you will be able to:

- Know where and when Command Design Pattern is implemented.

Problem Statement

If you've ever seen some old Java GUI code, or seen the GUI code written by a new developer, you might run across an actionPerformed() method in a Swing application that looks like this:

```
public void actionPerformed(ActionEvent e)
{
  Object o = e.getSource();
  if (o = fileNewMenuItem)
    doFileNewAction();
  else if (o = fileOpenMenuItem)
    doFileOpenAction();
  else if (o = fileOpenRecentMenuItem)
    doFileOpenRecentAction();
  else if (o = fileSaveMenuItem)
    doFileSaveAction();
  // and more ...
}
```

This code becomes very hard to maintain, and you'll have as many "if" statements as menu items. This is very tightly coupled, we have to separate sender from receiver. Use Command Design pattern to solve this problem.

Test Design

You need to implement Adapter Design Pattern using the following guidelines:

- Create a simple Java Command interface.
- Modify our JMenuItem to implement our Command interface.
- After these changes, write new actionPerformed() method in ActionListener class.

Solution Instructions

1.	There are no restrictions on what the field names and method names should be as long as they are named logically.

Lab 4. Template Method Design Pattern

Time Needed: (15min)

Skills Learned

At the end of this exercise, you will be able to:

- Identify the situations where the Template Method Design Pattern can be used to develop Java-based applications.

Problem Statement

Develop an application for processing flat files for customer investments in:

- Stocks or
- Derivatives

Requirements

Details of customers who invest in Stock or Derivatives are held in a database, which is accessed by an application used by financial advisors. In this application, investment flat files, Stock files are in CSV format and Derivative files are in text format. In a Stock file, data is separated by the delimiter ",". While, the Derivative file data in a flat file is separated using the '|' character. The data for either file format is persisted within the database being used. As a developer, you have to design an application that processes financial system data files. This application should be able to differentiate between a Stock flat file and a Derivative flat file while reading data from the database. Optionally, you can configure system login support in the application, to decide access control to the data files.

Application Design

To develop the application, you have to implement Template Method design pattern by using the following guidelines:

- DataHandler is the base abstract class that provides the template method algorithm for processing financial system data files.
- Algorithm: Template Method – execute().
- The Standard operations that are to be performed by the base class are:
    1. Validate file type
    2. Transform Data
    3. Process Data
- Subclasses should have methods to implement actions based on the investment file type (Stock or Derivative).

- The Hook Method for this application is the isDebugMode() method, for which the default configuration is set in the DataHandler class.

Solution Instructions

1.	There are no restrictions on what the field names and method names should be as long as they are named logically.