# Design Pattern Exercises

# 

Factory Method Pattern How does this pattern promote loosely coupled code?

Does the following code fragment implement the Factory Method design pattern?

public class XMLReaderFactory {

// This method returns an instance of a class

// that implements the XMLReader interface.

// The specific class it creates and returns is

// based on a system property.

public static XMLReader createXMLReader();

}

public interface XMLReader {

public void setContentHandler(ContentHandler handler):

public void parse(InputStream is);

}

Describe the working of the Abstract Factory in your own words.

What pattern(s) is(are) often used together with the Abstract Factory pattern?

Suppose you plan to manage address and telephone information as part of a personal information manager (PIM) application. The PIM will act as a combination address book, personal planner, and appointment and contact manager, and will use the address and phone number data extensively.

You can initially produce classes to represent your address and telephone number data. Code these classes so that they store the relevant information and enforce business rules about their format. For example, all phone numbers in North America are limited to ten digits and the postal code must be in a particular format.

Shortly after coding your classes, you realize that you have to manage address and phone information for another country, such as the Netherlands. The Netherlands has different rules governing what constitutes a valid phone number and address, so you modify your logic in the Address and PhoneNumber classes to take the new country into account.

Now, as your personal network expands, you need to manage information from another foreign country... and another... and another. With each additional set of business rules, the base Address and PhoneNumber classes become even more bloated with code and even more difficult to manage. What's more, this code is brittle—with every new country added, you need to modify and recompile the classes to manage contact information.

What design pattern can be used to solve the problem?

# Project

Design TMAPaint application (see the picture) with these functionalities:

1. Supports 2 look-and-feels (2D, 3D)
2. Allows user to add, remove shape objects. Shape object can be either Rectangle or Circle
3. Has a panel that list out current objects added. This panel will be updated once an object is added/removed. User can select the objects in this panel to change its parameters. The change will be immediately reflected on the main panel.
4. Supports undo/redo
5. Supports printing function which implements this predefined interface:

interface Printer() {

// start up the physical device

// load paper

// ...

// print the object

// printedObject.print();

// stop the physical device

public void print(PrintObject printedObject);

}

class PrintObject{

public void print(Graphics g) {…}

}

Circle (2)

🔾3D

⚫2D

Print

Redo

Undo

Remove

Add

Position

Width

100

50

50

Radius

Height

100

50

100

(10, 20)

(30, 50)

(10, 20)

(30, 50)

Position

Rectangle (3) (2)

100

100

(30, 50)