**Chapter 4 – Abstract Factory Pattern Notes**

1. What is the *Dependency Inversion Principle*?

|  |
| --- |
| *Depend upon abstractions. Do not depend upon concrete classes*. This is stronger than, *Program to an interface, not an implementation*. It suggests that high-level components should not depend on low-level components, they should both depend on abstractions.  See page Ch. 4, p.29-30, 32 |

1. What is the intent of the abstract factory pattern?

|  |
| --- |
| The Abstract Factory Pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes. It helps control the objects that an application creates. It enforces the dependencies between the concrete product classes |

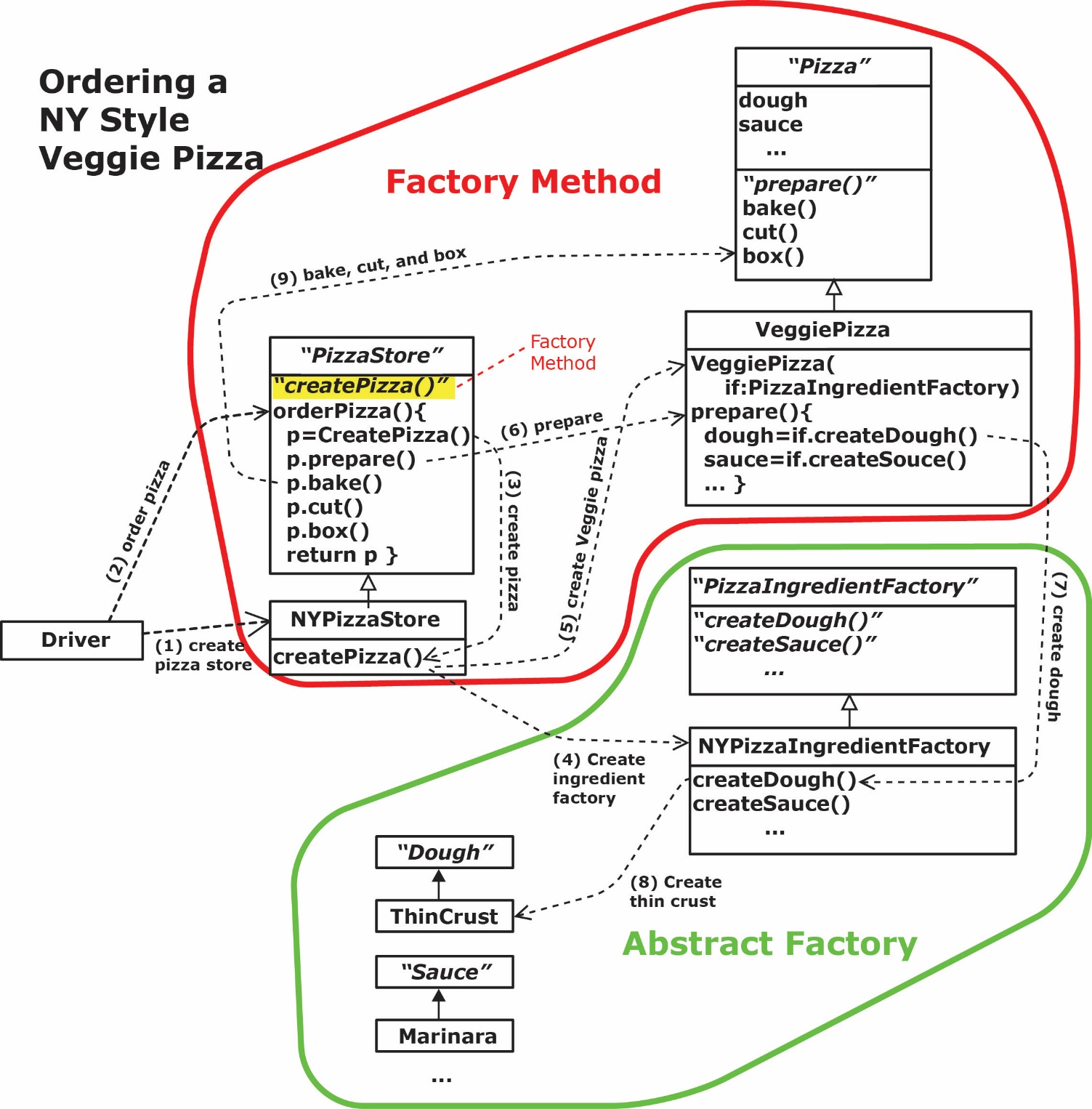
1. Draw a generalized class diagram for the abstract factory pattern.

|  |
| --- |
| E:\Data-Classes\CS 4322 - Software Engineering 2\Notes\04a-Factory Method\pics\aa7.jpg |

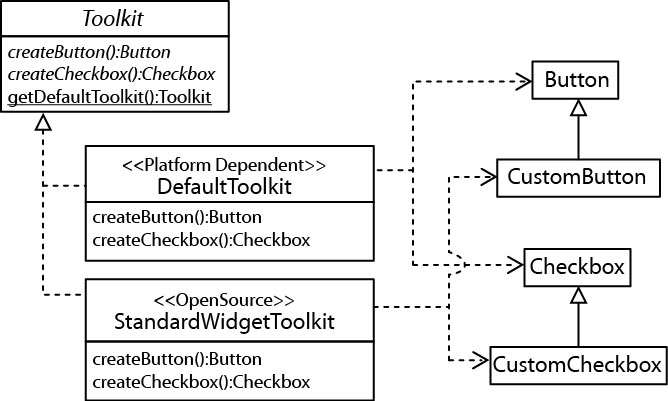
1. How are abstract factory and factory method different?

|  |
| --- |
| The factory method is a single method that is overridden (implemented) in a subclass. With the factory method, usually an object (Creator) is calling its own factory method. The difference is that the intended purpose of the class containing a factory method is not to create objects, that is just a step in fulfilling other responsibilities.  The abstract factory is an object. Its methods are usually implemented with factory methods. In other words, there is a client that wants to make an object. Instead of creating the object itself (e.g. with a factory method), it uses a different object (abstract factory) that it is composed with to create the object.  E:\Data-Classes\CS 4322 - Software Engineering 2\Notes\04a-Factory Method\a4.jpg |

1. Example from text

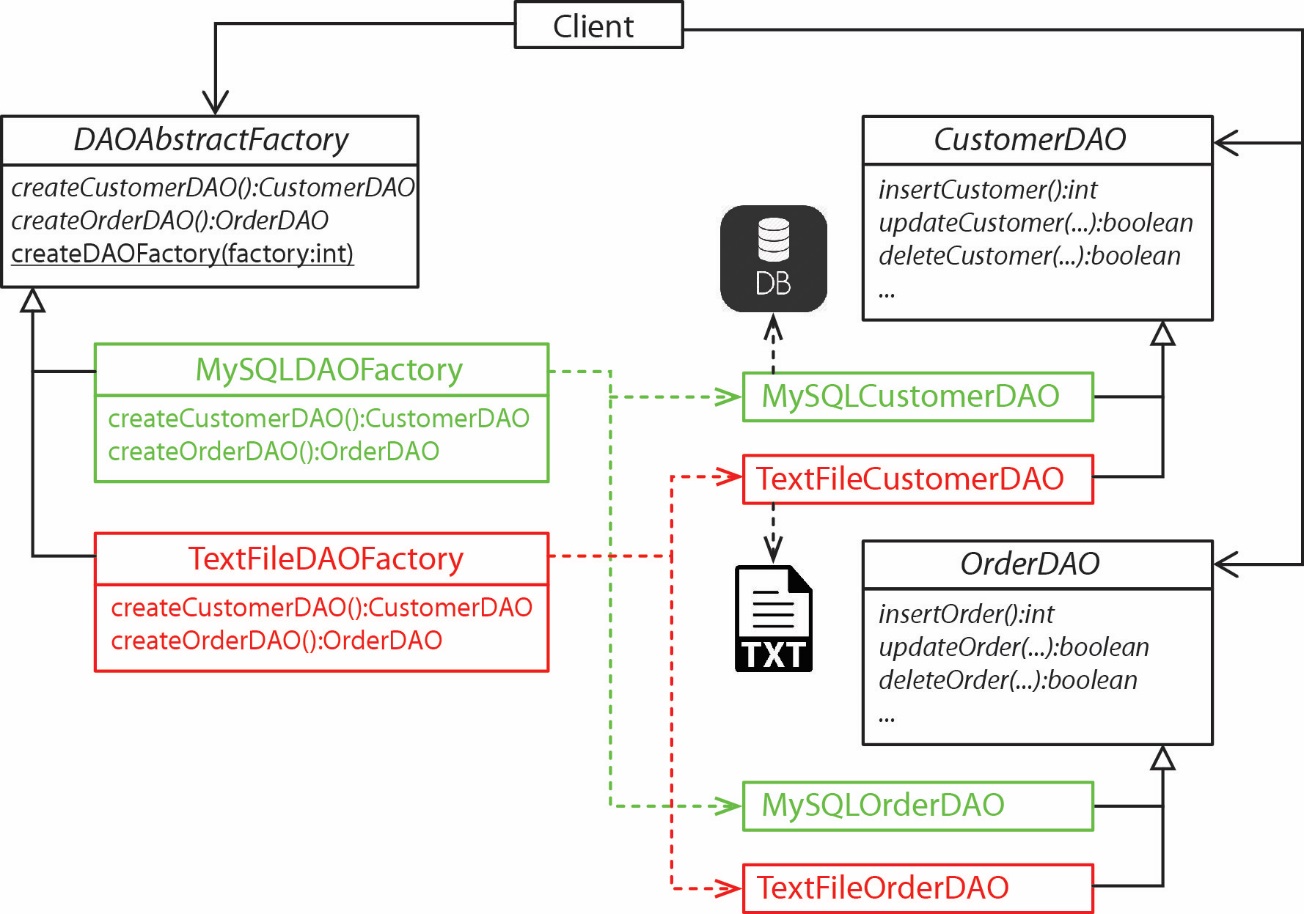


1. Abstract Factory in Java – The java.awt.Toolkit abstract class is an abstract factory. It serves as an abstract layer between the programmer and the native windowing procedures.



* SWT is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented. <https://www.eclipse.org/swt/>
* The *SchemaFactory* abstract class in Java is a base class used for validating XML. It specifies a number of methods for obtaining a Schema object as well as static methods for creating a concrete SchemaFactory.
* The *DocumentBuilderFactory* defines a factory API that enables applications to obtain a parser that produces DOM object trees from XML documents.
* There are many abstract factories in the Java API. Do a search on Factory in the API and you will see more than 140 classes with *factory* in the name. Some are simple factories and some are abstract.

1. Data Access Object (DAO) Factory.



A reasonable reference for DAO is: <https://gerardnico.com/lang/java/dao>

1. Other Examples
2. Flexible Manufacturing – an AbstractCarFactory, with concrete factories: AltimaFactory, MaximaFactory, *etc*. The products are doors, hood, engine, *etc*. A machine could cut different families of parts from sheet metal. Or with robotic assembly, a robot could assemble different families of parts to produce a product.
3. EnvironmentFactory produces walls, doors, *etc* for a game. Different concrete factories can produce these things for different, but related types of games, or for different levels of the same game: beginner, intermediate, advanced.
4. Some real uses for abstract factory: <https://stackoverflow.com/questions/2280170/why-do-we-need-abstract-factory-design-pattern>
5. Homework:

A company must produce reports. All reports have some things in common. There are two types of reports: reports that provide an overview of inventory and reports that provide an overview of sales. In addition, each type of report is different depending on the destination. For example, inventory reports, no matter their destination (local or corporate) have things in common, but differ in the granularity of the presented data. In other words, local management wants to see it one way and corporate management wants to see it another. The same is true for sales reports. are inventory reports and sales reports. Reports are generated for local management and corporate management. For example, an inventory has some things that are the same no matter whether it is for local or corporate management, but then other things are different.

1. Use the abstract factory pattern to model this situation with a class diagram. Use a simple factory to choose the proper concrete factory to create.
2. Write the code for all classes. Note: this truly is a “hello world” application.
3. Write some code to show how this works.