**Workout 2020-09-09 – Factory Patterns Name: Russell Springer**

**Submission Instructions:**

For the questions in parts 1-3 and 5 below, answer the questions in this document and save the document to the program directory you create for Question 4. Zip up the workout directory and submit it to ASULearn. No package declarations please.

* Simple Factory
* Define the “Simple Factory”?

It's not actually a design pattern, more of a "programming idiom". A client is related to the simple factory that 5takes charge of creating new instances of the class, which are further specified by the individual classes

* In one or two sentences, describe how the Simple Factory is different from the Factory Method Pattern?

in the case of the factory method pattern, abstract classes, creator and product outline what concrete components, concretreContainer and ConcreteProduct do, where the concrete creator takes charge of instantiating new objects (products)

* Factory Method Pattern
* Draw the UML diagram (using one of our UML applications) for a classic/typical Factory Method Pattern design. Save it as a png image and include it here.



* Define the term “Product Classes”.

the concrete products

* Define the term “Creator Classes”.

factory methoids that create product classes

* Write the definition of the Factory Method Pattern.

the factory method patter defines an uinterface for creating an object, but lets the subclasses decide which class to instantiate./ Fcotry Method lets a class defer instantiation to subclasses

* Dependency Inversion Principle
* Write the definition of the DIP.

depend upon abstractions do not depend on concrete classes

* What is meant by “Class A is dependent on Class B”?

That class A uses class B directly and so if class b changes, class A might have to change toi operate the same, also class A might instantiate an instance of class B

* Write the three guidelines the textbook gives for following DIP.

no var should hold a ref toi a concrete class,

no class should derive from a concrete class,

no method should override an implemented method of any of its base classes.

* Describe how this principle is followed in the Factory Method Pattern.

beside the fact that all three of those guide lines are followed, the factory patter uses abstractions to create lower level components and seperate them from higher ones

* Programming – During the next couple of workouts, we’ll create a Factory for Game Boards. We’ll start with chess and will write only the game pieces today with a Simple Factory.

You will need the following classes (in addition to the Client to test the other classes.)

* ChessPieceFactory is the Simple Factory. It uses a default constructor and has one method:  
   Piece createPiece(String color, String type).  
  createPiece returns a chess piece of the requested type and color.
* An abstract Piece class with two protected fields:  
   String color, type  
  color will have values “Black” or “White”. type will contain one of the following: “Rook”, “Bishop”, “knight”, “Queen”, “King”, or “Pawn”.  
  Piece has one method:  
   toString() the returns a 2-character String containing the first character of color and the first character of type. Such as “BK” for the Black King or “Wk” for the White knight.
* Concrete classes for Rook, Bishop, Knight, Queen, King, and Pawn that extends Piece. The constructor is passed a String (the color) and assigns the inherited fields color and type.
* A Chess class containing a method named play() that does nothing more than create one of each piece and displays the piece. The Chess class should have a field for its ChessPieceFactory, and the Chess constructor should be passed the factory instance.
* A Client class that contains main, instantiates the ChessPieceFactory, instantiates the Chess class, and calls play()
* What advantages exist for creating the Simple Factory as opposed to the client directly calling constructors for the chess pieces? What Design Principle is the motivation behind this “almost pattern”?

the client doesnt care what the constructors do, or even if they actually exist, all that responsibility is thrown over to the chess class and again to the chess piece factory. It seperastes the cliemnt from the imnner workin gs of the program.