Factory Method
Factory Method Intent

Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

(Creational)
The iterator() method in the Java Collections Framework is a Factory Method.

Every compliant collection must implement the iterator() method which defines a common interface for creating iterators.

Consider this code snippet.

```java
class FactoryMethodExample {
    public static void main(String argv[]) {
        List aList = new ArrayList();
        Iterator alIterator = aList.iterator();
    }
}
```

What is the class type of `alIterator`? Do you care?

What do you really care about the object that is created and returned by the call to iterator()?
The structure of the Factory Method separates the Creator and the Product.

A separate client of a Creator often makes the Factory Method call, or it can be an internal call as shown.
Now let’s take a closer look at the Java iterator example.

```java
import java.util.List;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Iterator;

class FactoryMethodExample {
    public static void main(String argv[]) {
        List aList = new ArrayList();
        Iterator alIterator = aList.iterator();
        System.out.println("alIterator is a " + alIterator.getClass().getName());

        List lList = new LinkedList();
        Iterator llIterator = lList.iterator();
        System.out.println("llIterator is a " + llIterator.getClass().getName());
    }
}
```

(26) ...jrv/RIT/Courses/SWEN-262/examples>javac FactoryMethodExample.java
(27) ...jrv/RIT/Courses/SWEN-262/examples>java FactoryMethodExample
alIterator is a java.util.ArrayList$Itr
llIterator is a java.util.LinkedList$ListItr
(28) ...jrv/RIT/Courses/SWEN-262/examples>

What does the $ indicate about the class?
With that information, we can see how this matches the Factory Method structure.

List is an interface so this is not an exact match to GoF pattern.
There are many variations on the classic GoF pattern that developers call Factory Method.

- The classic pattern is a Factory **Method** not a Factory **class**.
  - *If you have a class whose only responsibility is to create objects that match a single interface, there probably is a better class for the method.*
  - *The Abstract Factory pattern contains multiple Factory Methods for a family of related classes.*

- Some "Factory Methods" use a parameter to indicate the object type to create.
  - *Are those different object types really related?*
  - *Should separate factory methods be located in the classes holding the particular data each uses?*
  - *A switch on the parameter is a bad code smell.*