**Factory Method**

* **Introduction:**
  + We want to move the object creation logic from our code to a separate class.
  + We use this pattern when we don’t know in advance which class we may need to instantiate beforehand & also to allow new classes to be added to system and handle their creation without affecting client code.
  + We let the subclasses decide which object to instantiate by overriding the factory method.
* **UML:**

**A screenshot of a cell phone

Description automatically generated**

Role – Product:

Base class or interface of products created by factory method.

Role – Concrete Product:

Implements the product interface or class.

Role – Creator:

Declares the abstract factory method.

Additionally uses the factory method to create product.

Role – Concrete Creator:

Implements factory method and returns one of concrete product instance.

* **Implementation Steps:**
  + We start by creating a class for our creator.
    - Creator itself can be concrete if it can provide a default object or it can be abstract.
    - Implementations will override the method and return an object.
* **Example – UML:**

**A screenshot of a cell phone

Description automatically generated**

* **Implementation & Design Consideration:**
  + The creator can be a concrete class & provide default implementation for the factory method. In such cases you’ll create some default object in base creator.
  + You can also use the simple factory way of accepting additional arguments to choose between different object types. Subclasses can then override factory method to selectively create different objects for some criteria.

**Design Consideration:**

* + Creator hierarchy in factory method pattern reflects the product hierarchy. We typically end up with a concrete creator per object type.
  + Template method design pattern often makes use of factory methods.
  + Another creational design pattern called “abstract factory” makes use of factory method pattern.
* **Example:**

The java.util.Collection (or java.util.AbstractCollection) has an abstract method called iterator(). This method is an example of factory method.

Remember the most defining characteristic of factory method pattern is “subclasses providing the actual instance”.

So static methods returning object instances are technically GoF factory methods.

* **Pitfalls:**
  + More complex to implement. More classes involved and need unit testing.
  + You have to start with Factory Method design pattern from the beginning. It is not easy to refactor existing code into factory method pattern.
  + Sometimes this pattern forces you to subclass just to create appropriate instance.