**Simple Factory**

* **Introduction:**

Multiple types can be instantiated and the choice is based on some simple criteria



Here we simply move the instantiation logic to a separate class and most commonly to a static method of this class.

Some do not consider simple factory to be a “design pattern”, as its simply a method that encapsulates object instantiation. Nothing complex goes on in that method.

“We are studying simple factory as it is often confused with ‘factory method’ pattern.

Typically we wanted to do this if we have more than one option when instantiating object and a simple logic is used to choose correct class.

* **UML:**

**A screenshot of a cell phone

Description automatically generated**

Role – Product:

Objects of this class & it’s sub-classes are needed.

Role – Simple Factory:

Provides a static method to get instance of product subclass.

* **Implementation Steps:**
  + We start by creating a separate class for our simple factory.
    - Add a method which returns desired object instance.
      * This method is typically static and will accept some argument to decide which class to instantiate.
      * You can provide additional arguments which will be used to instantiate objects.
* **Implementation considerations:**
  + Simple factory can be just a method in existing class. Adding a separate class however allows other parts of your code to use simple factory.
  + Simple factory itself doesn’t need any state tracking so it’s best to keep this as a static method.
  + Design consideration:
    - Simple factory will in turn may use other design pattern like builder to construct objects.
    - In case you want to specialize your simple factory in sub-classes, you need factory method design pattern instead.
* **Example:**

The java.text.NumberFormat class has getInstance() method, which is an example of simple factory.

* **Comparison with Factory Method**

|  |  |
| --- | --- |
| Simple Factory | Factory Method |
| We simply move our instantiation logic away from client code. Typically in a static method. | Factory method is more useful when you want to delegate object creation to sub-classes. |
| Simple factory knows about all classes whose object it can create. | In factory method we don’t know in advance about all product sub-classes. |

* **Pitfalls:**

The criteria used by simple factory to decide which object to instantiate can get more convoluted / complex over time. If you find yourself in such situation then use factory method design pattern.