Factory Method and Factory Class

Create a project named 08\_FactoryMethodAndFactoryClass and copy the previous project files at our new project and we will work from there.

1. At App.java remove patient2 object and rename patient1 as patient
2. Create factory method at Patient.java class

|  |
| --- |
| **public** **static** Patient **getInstance**(**int** id,String name) {  System.***out***.println("Creating patient using factory method");  **return** **new** Patient(id,name);  } |

1. Go to patient bean and define factory-method=getInstance
2. Run App.java
3. Create Factory class
   1. Create PatientFactory.java class
   2. Create createPatient(int id, String name) method
   3. If we don’t create factory class then the factory method should be static else non static.

|  |
| --- |
| **public** Patient **createPatient**(**int** id, String name) {  System.***out***.println("Using Factory Class to create patient!");  **return** **new** Patient(id,name);  } |

1. Create patient2 bean
   1. Name=Molly
   2. Factory-bean=patientFactory
   3. Factory-method= createPatient
   4. Create patientFactory bean
2. Run App.java

The Factory Method Pattern is also known as **Virtual Constructor.**

#### Advantage of Factory Design Pattern

* Factory Method Pattern allows the sub-classes to choose the type of objects to create.
* It promotes the **loose-coupling** by eliminating the need to bind application-specific classes into the code. That means the code interacts solely with the resultant interface or abstract class, so that it will work with any classes that implement that interface or that extends that abstract class.

#### Usage of Factory Design Pattern

* When a class doesn't know what sub-classes will be required to create
* When a class wants that its sub-classes specify the objects to be created.
* When the parent classes choose the creation of objects to its sub-classes.