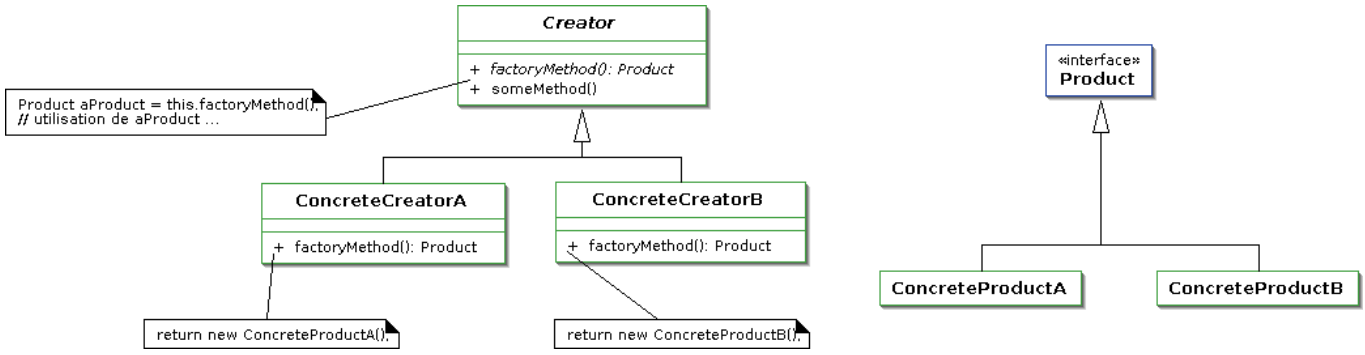
Design Pattern : factory method

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.*

Ce design pattern est également appelé *Virtual Constructor*

Structure



Eléments caractéristiques

- une méthode dont la fonction est de créer un objet, cet objet est renvoyé comme résultat.

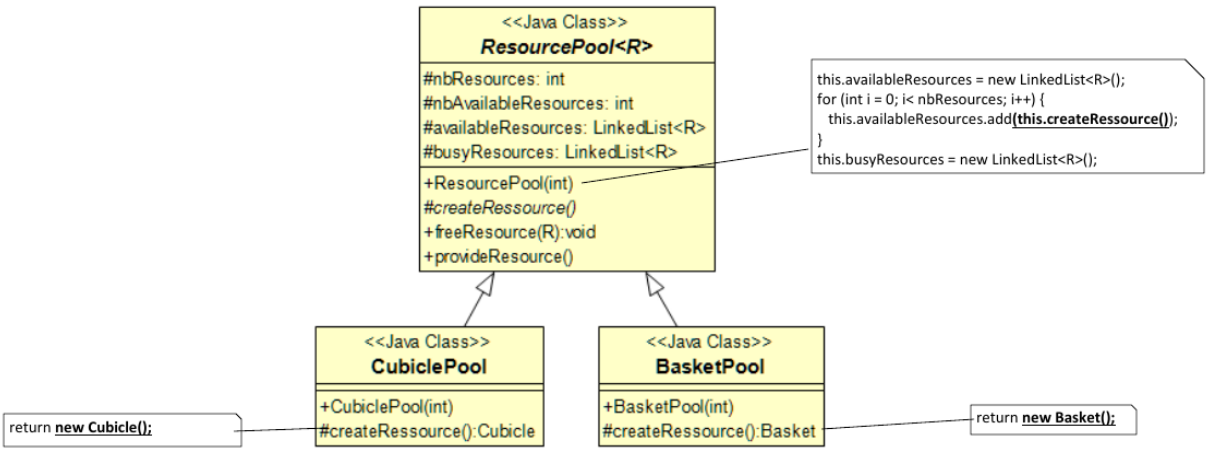
  Cette méthode est surchargée dans les sous-classes pour modifier la classe de l'objet qu'elle crée.

Exemples rencontrés en TD

Dans l'api java

- la méthode `iterator()` de `java.util.Collection` crée et fournit un itérateur dont l'implémentation diffère selon la classe concrète de la collection considérée.

TD – Piscine : initialisation du gestionnaires de ressources.
Initialisation du pool de ressources de chaque type de gestionnaire de ressources avec les ressources adaptées : la responsbilité de la création des ressources est déléguée aux sous-classes de `ResourcePool` grâce à la *factory method* `createResource()`.
NB : le constructeur `ResourcePool` reprend le principe d'une *template method* en s'appuyant sur la *factory method* abstraite.



TD – Casse-briques : création des « briques graphiques ».
Ici la *factory method* `createBlockView()` délègue le travail de création à un objet de type `IBlockViewFactory`. Cela permet en créant plusieurs implémentations de `IBlockViewFactory` (telles que `BlockViewFactory`) de générer des vues différents pour les briques (plusieurs « looks »).