Travaux pratiques/dirigés de *NFA 032* – Année *2023-2024*

Enseignant *: Pierre Sylvain ILOGA BIYIK*

Le pattern Factory Method

Objectif : Introduire une méthode abstraite de création d'un objet en reportant aux sous-classes concrètes la création effective.

Prérequis : Langage UML, classes abstraites, héritage, encapsulation, exceptions.

Cas pratique 1 : Considérez l'entreprise de vente de véhicules des TPs portant sur les patterns *Abstract Factory* et *Builder*. Après avoir reçu les documents relatifs à la commande de son véhicule, chaque *client* doit effectivement acheter le véhicule choisi en précisant son *mode de paiement*. L'entreprise permet aux clients de payer par *carte bancaire* ou de payer par *PayPal*. Le système de gestion des paiements de l'entreprise accepte ou refuse un mode de paiement en fonction de sa validité. Le paiement par carte bancaire est toujours valide, peu importe le prix officiel (entier) du véhicule. Par contre, le paiement par PayPal est valide uniquement pour les véhicules dont le prix officiel est compris entre 1000€ et 3000€.

Travail à faire : Vous supposerez que le catalogue contient exactement les 13 véhicules contenus dans le Tableau 3, du cas pratique 3, du TP portant sur le pattern *Abstract Factory*. Il vous est demandé d'utiliser le pattern ***Factory Method*** pour écrire un programme Java qui va implémenter ce système gestion des paiements de l'entreprise. Votre programme doit permettre à chaque client de saisir au clavier le prix officiel du véhicule sollicité et son mode de paiement tel que décrit ci-dessous, après quoi le programme va afficher à l'écran un message adéquat en fonction des données fournies par le client :

```
Prix officiel : ___
Mode de paiement - (1) Carte (2) PayPal : ___
```

I- Compréhension de l'énoncé

Afin d'apporter une solution au cas pratique 1, vous devez dans un premier temps vous servir de l'énoncer pour répondre aux questions suivantes :

1- Quel produit générique est manipulé dans cet énoncé ?
2- Quel l'unique attribut de ce produit générique ?
3- Quelles sont les méthodes abstraites de ce produit générique ?
4- Quels sont les produits dérivés de ce produit générique ?
5- Quel acteur générique manipule le produit générique ?
6- Quelles sont les méthodes abstraites de cet acteur générique ?
7- Quelles sont les méthodes concrètes de cet acteur générique ?
8- Quels sont les acteurs dérivés de cet acteur générique ?

II- Programmation en Java

Considérez le diagramme UML général associé au Pattern ***Factory Method*** de la Figure 1.

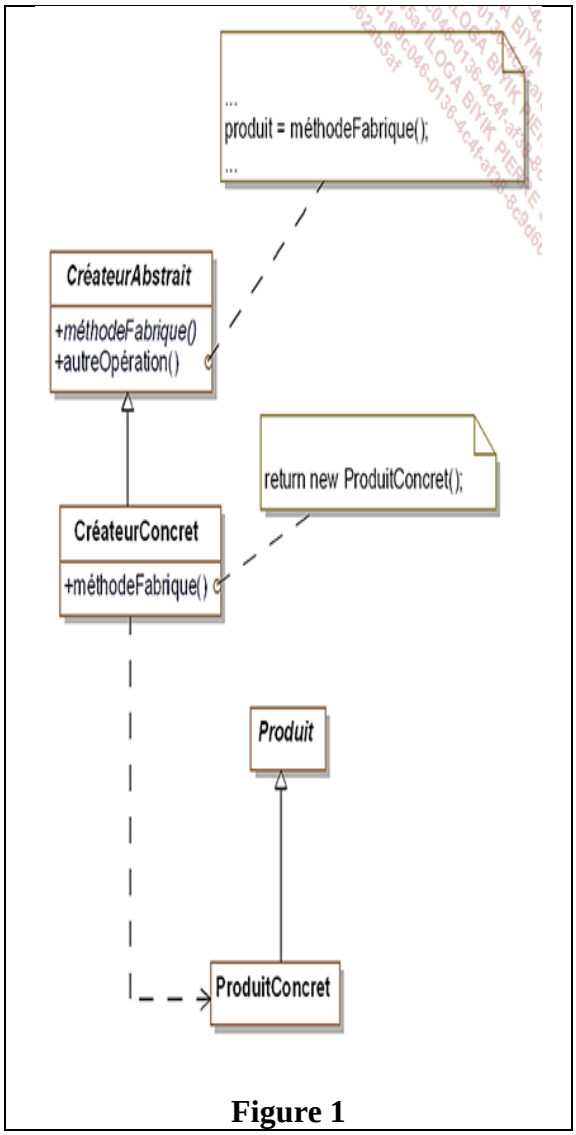

Figure 1

En vous appuyant sur les diagrammes UML des Figures 1 et 2, ainsi que sur les réponses aux questions de la Section I, répondez aux questions suivantes :

1- Proposez le diagramme des classes UML adapté au cas pratique 1 en vous appuyant sur celui de la Figure 1.

2- Créez une classe abstraite correspondant au produit générique identifié à la Question I-1. Cette classe doit disposer des méthodes (abstraites) identifiées à la Question I-3. Le constructeur de cette classe doit initialiser son unique attribut identifié à la Question I-2 avec une valeur prise en paramètre.

3- Pour chacun des produits dérivés identifiés à la Question I-4, vous devez créer une classe fille concrète qui va hériter de la classe abstraite créée à la Question II-2. Chaque classe fille a le même constructeur que sa classe mère, mais elle doit implémenter de manière spécifique chaque méthode abstraite conformément à l'énoncé.

4- Créez une classe abstraite correspondant à l'acteur générique identifié à la Question I-5. Cette classe n'a pas de constructeur, mais elle doit disposer des méthodes identifiées aux Questions I-6 et I-7.
5- Pour chacun des acteurs dérivés possibles identifiés à la Question I-8, créez une classe fille concrète qui va hériter de la classe abstraite créée à la Question II-4. Chaque classe fille concrète ne dispose aussi d'aucun constructeur, mais va effectivement implémenter l'unique méthode abstraite héritée de sa classe mère.
6- Enfin, vous allez créer la classe associée au programme principal qui correspond ici au système de gestion des paiements. Cette classe va se servir de toutes les classes créées depuis le début afin de simuler l'achat d'un véhicule conformément aux consignes données dans l'énoncé.

Cas pratique 2: Suite à des requêtes massives des clients, l'entreprise va évoluer en permettant le paiement par *Paylib* lors de l'achat d'un véhicule, en plus des paiements par carte bancaire et par PayPal. Dans les mêmes conditions que dans le cas pratique 1, et en considérant qu'un paiement par Paylib est valide uniquement pour les véhicules dont le prix officiel est compris entre 500€ et 2500€, il vous est demandé d'utiliser le pattern ***Factory Method*** pour écrire un programme Java qui va prendre en compte cette évolution de l'entreprise pour la gestion de l'achat des véhicules par les clients. Chaque client va saisir au clavier le prix officiel du véhicule sollicité et son mode de paiement tel que décrit ci-dessous, après quoi le programme va afficher à l'écran un message adéquat en fonction des données fournies par le client :

```
Prix officiel : ___
Mode de paiement - (1) Carte (2) PayPal (3) Paylib :
```

Cas pratique 3 : Par la suite, au terme d'une réunion du conseil d'administration de l'entreprise, il a été décidé de donner aux clients la possibilité de louer un véhicule pour une durée fixe de 10 jours, en plus de la possibilité de les acheter. Vous supposerez que le prix officiel de la location de chaque véhicule est disponible dans le catalogue. Afin d'encourager les clients à utiliser cette nouvelle offre, le service marketing l'entreprise a par ailleurs décidé de manière promotionnelle d'accorder à chaque client une remise immédiate de 10% du prix officiel lors de la location d'un véhicule. Il vous est demandé d'utiliser le pattern ***Factory Method*** pour écrire un programme Java qui va prendre en compte cette évolution de l'entreprise pour la gestion de l'achat ou de la location des véhicules par les clients. Chaque client va saisir au clavier la nature de l'opération à réaliser et le prix officiel de cette opération, ainsi que son mode de paiement tel que décrit ci-dessous, après quoi le programme va afficher à l'écran un message adéquat en fonction des données fournies par le client :

```
Opération - (1) Achat (2) Location : ___
Prix officiel : ___
Mode de paiement - (1) Carte (2) PayPal (3) Paylib :
```

Problème : Il vous demandé d'écrire un unique programme Java qui fusionne le problème résolu dans le TP portant sur le pattern **Builder**, avec le cas pratique 3 du présent TP portant sur le pattern **Factory Method**. La fusion consistera simplement à ajouter un nouveau package nommé *paiement* au projet créer pour résoudre le problème résolu dans le TP portant sur le pattern **Builder**. Ainsi, après avoir navigué dans le catalogue, le client doit choisir le véhicule qui l'intéresse ainsi que le format des documents. Enfin il doit décider de l'opération à réaliser (achat ou location) puis du mode paiement. Le client n'a donc plus besoin de saisir au clavier le prix officiel vu qu'il est déjà présent dans le catalogue.