Travaux pratiques/dirigés de *NFA 032* – Année *2023-2024*
Enseignant *: Pierre Sylvain ILOGA BIYIK*

Le pattern Abstract Factory

Objectif : Création d'objets regroupés en *familles*, sans devoir connaître les classes concrètes destinées à la création de ces objets.

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

Cas pratique 1 : Une entreprise de vente de véhicules gère des véhicules fonctionnant à l'*essence* et des véhicules fonctionnant à l'*électricité*. Cette entreprise met à la disposition de ses clients un catalogue électronique en ligne dans lequel ils peuvent consulter les caractéristiques des véhicules disponibles. Dès sa création, l'entreprise dispose de deux types de véhicules, il s'agit des *automobiles* et des *scooters*. Chacun de ces véhicules possède les caractéristiques suivantes dans le catalogue :

- Son *modèle* : chaîne de caractères.
- Sa *couleur* : chaîne de caractères.
- Sa *puissance* (en nombre de chevaux) : entier.
- Sa *masse* (en Kg et uniquement pour les automobiles) : réel.

Travail à faire : Il vous est demandé d'utiliser le pattern ***Abstract Factory*** pour écrire un programme *Java* qui va créer ce catalogue avec le contenu décrit dans le Tableau 1. Dans votre programme, chaque véhicule disposer, d'une seule méthode nommée *AfficheVehicule* lui permettant d'afficher à l'écran un message adéquat incluant ses caractéristiques, son type et sa famille. Après avoir créé le catalogue, votre programme va d'abord afficher les informations sur le premier véhicule, puis permettre à chaque client de consulter le catalogue en saisissant des caractères au clavier ainsi qu'il suit :

```
'a' : Afficher tout le catalogue
'p': Afficher véhicule précédent
's' : Afficher véhicule suivant
'f' : Fermer le catalogue
Votre choix : ___
```

Tableau 1 : Contenu du catalogue à la création de l'entreprise

Type	Num	Modèle	Couleur	Puissance	Masse	Famille
Automobiles	1	van	jaune	600	3200	essence
	2	suv	rouge	140	1600	électricité
	3	coupé	rouge	300	1000	électricité
	4	suv	noir	200	1200	essence
Scooters	5	2 roues	noir	20	-	électricité
	6	3 roues	jaune	30	-	électricité
	7	2 roues	bleu	15	-	essence

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 est le produit générique mis en vente par l'entreprise ?
2- Quels sont les produits dérivés de ce produit ?
3- Quels sont les attributs de chaque produit dérivé ?
4- Quelles sont les méthodes de chaque produit dérivé ?
5- A quelles familles chaque produit dérivé peut-il appartenir ?

II- Programmation en Java

Considérez le diagramme UML général associé au Pattern ***Abstract Factory*** présenté à la Figure 1.

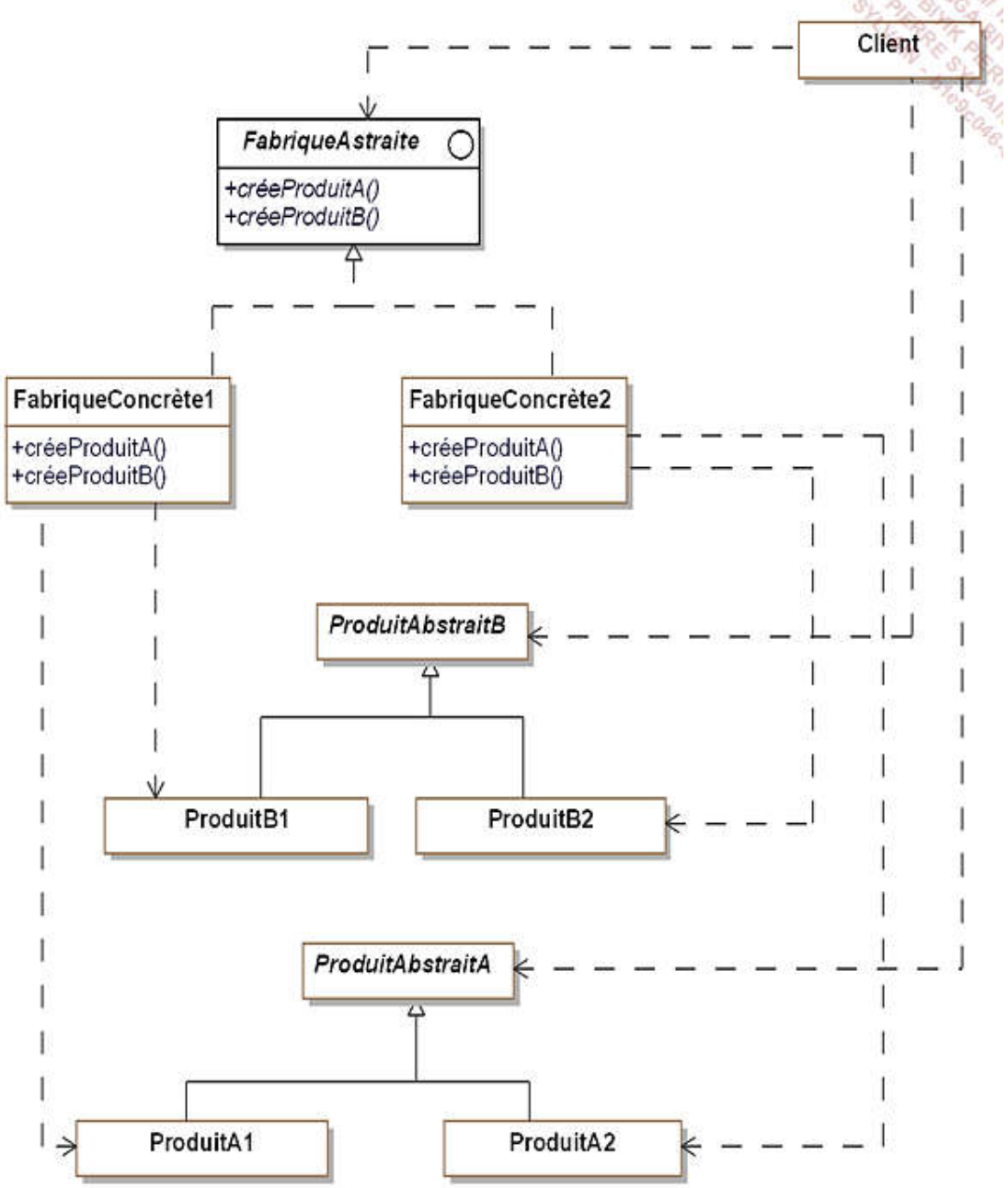

Figure 1

En vous appuyant sur le diagramme UML de la Figure 1 et sur les réponses aux questions de la Section I, répondez aux questions suivantes :

1- Construisez le diagramme UML spécifique à l'énoncé en renommant correctement les classes et les méthodes.
2- Pour chaque produit dérivé, créez une classe abstraite qui dispose d'un constructeur adéquat pour initialiser les attributs du produit dérivé avec des valeurs fournies en paramètres. Cette classe abstraite doit aussi disposer des méthodes (abstraites) identifiées à la Question I-4.
3- Pour chacune des familles possibles du produit dérivé, 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 interface qui va représenter le produit générique mis en vente par cette entreprise. Cette interface n'a besoin d'aucun constructeur et d'aucun attribut. Mais elle doit disposer d'autant de méthodes abstraites que de produits dérivés identifiés à la Question I-2. Chaque méthode abstraite a pour rôle de prendre en paramètres les attributs d'un produit dérivé et de créer (retourner) ce produit dérivé tel que défini à la Question II-2.
5- Pour chacune des familles possibles du produit dérivé, créez une classe fille concrète qui va implémenter l'interface créée à la Question II-4 en créant effectivement des produits dérivés de chaque famille dans les méthodes héritées de la classe mère. Ces méthodes vont donc retourner des produits dérivés concrets tels que définis à la Question II-3.
6- Enfin, vous allez créer la classe associée au programme principal qui correspond ici au *Client* dans le diagramme UML (et au catalogue dans l'énoncé). Cette classe va se servir de toutes les classes créées depuis le début afin de remplir le catalogue conformément au contenu du Tableau 1 et pour simuler la manipulation du catalogue par le client conformément aux consignes données dans l'énoncé.

Cas pratique 2 : Deux ans après sa création, l'entreprise a évolué en mettant aussi en vente des véhicules *hybrides*, fonctionnant à la fois à l'essence et à l'électricité. Il vous est demandé d'utiliser le pattern ***Abstract Factory*** et de suivre la méthodologie utilisée au cas pratique 1 pour écrire un programme *Java* qui va prendre en compte cette évolution en supposant que le catalogue contient les véhicules décrits dans le Tableau 2.

Tableau 2 : Contenu du catalogue deux ans après la création de l'entreprise

Type	Num	Modèle	Couleur	Puissance	Masse	Famille
Automobiles	1	van	jaune	600	3200	essence
	2	suv	rouge	140	1600	électricité
	3	coupé	rouge	300	1000	électricité
	4	suv	noir	200	1200	essence
	5	coupé	vert	500	2500	hybride
Scooters	6	2 roues	noir	20	-	électricité
	7	3 roues	jaune	30	-	électricité
	8	2 roues	bleu	15	-	essence
	9	3 roues	rouge	35	-	hybride
	10	2 roues	vert	20	-	hybride

Cas pratique 3 : Cinq ans après sa création, l'entreprise a une nouvelle fois évolué en mettant en vente des *jets ski*, en plus des automobiles et des scooters. Chaque *jet ski* possède toutes les caractéristiques prévues pour un véhicule dans le système (y compris sa masse). Il vous est demandé d'utiliser le pattern ***Abstract Factory*** et de suivre la méthodologie utilisée au cas pratique 1 pour écrire un programme *Java* qui va prendre en compte cette évolution en supposant que le catalogue contient les véhicules décrits dans le Tableau 3.

Tableau 3 : Contenu du catalogue cinq ans après la création de l'entreprise

Type	Num	Modèle	Couleur	Puissance	Masse	Famille
Automobiles	1	van	jaune	600	3200	essence
	2	suv	rouge	140	1600	électricité
	3	coupé	rouge	300	1000	électricité
	4	suv	noir	200	1200	essence
	5	coupé	vert	500	2500	hybride
Scooters	6	2 roues	noir	20	-	électricité
	7	3 roues	jaune	30	-	électricité
	8	2 roues	bleu	15	-	essence
	9	3 roues	rouge	35	-	hybride
	10	2 roues	vert	20	-	Hybride
Jets ski	11	recreation	blanc	800	700	essence
	12	tow sport	jaune	1000	900	électricité
	13	tow sport	blanc	950	600	hybride