**NFA019 : Série 4.1- Morane Gruenpeter**

a. La série 4.1 est la première rencontre avec l’héritage en NFA019. On utilise le concept d’héritage pour créer des objets du même type avec des attributs et des méthodes en plus. Sachant que la classe mère est plus générale et la classe fille a tous les attributs de la classe mère en plus de ses propres attributs, elle est donc plus spécifique. Pour la série 4.1 on va créer des Articles en promotion qui ont un float de réduction et un int de quantité minimale en plus des attributs de la classe Article.

b. La seule classe que l’on crée pour cette série est la classe ArticlePromo, or celle-ci demande certaine modification sur les autres classes.

**Les classes qui ont été modifiées :**

* Articles41 : ajout de la méthode prixfacture(int quantite)
* Client41 : ajout de la méthode Tools.calculeEspace(String avant, int largeur) pour les affichages.
* GestionTableDesArticles41 :

1. ajout de la méthode Tools.calculeEspace(String avant, int largeur) pour les affichages.
2. Pour la méthode ajouter() : ajout de la possibilité d’ajouter un article en promotion.
3. Ajout de la méthode editerPromo() qui affiche que les articles en promotion si il y en a.

* GestionTableDesCommandes41 : ajout de la méthode Tools.calculeEspace(String avant, int largeur) pour les affichages.
* GestionDuneCommande41 :

1. ajout de la méthode Tools.calculeEspace(String avant, int largeur) pour les affichages.
2. Changement de la méthode ajouter() qui affiche un message si l’article choisi est en promotion pour savoir la quantité minimale nécessaire pour avoir la promotion.

* UneCommande41 : ajout de la méthode Tools.calculeEspace(Stringavant, int largeur) pour les affichages.
* LigneDeCommande41 :

1. ajout de la méthode Tools.calculeEspace(String avant, int largeur) pour les affichages.
2. Ajout à la méthode facturer() si l’article de la ligne de commande est un ArticlePromo41 on ajoute à la ligne de cette facture le stringFacture() une méthode de la classeArticlePromo41.

* Tools :création de la méthode calculeEspace(String avant, int largeur) pour les affichages.
* iPane.ES : ajout d’unjTextArea pour un affichage plus jolie.

## Class ArticlePromo41

java.lang.Object

Article41

**ArticlePromo41**

public class **ArticlePromo41** extends Article41

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ArticlePromo41**](about:blankArticlePromo41.html#ArticlePromo41())()           constructeur par défaut |  |
| [**ArticlePromo41**](about:blankArticlePromo41.html#ArticlePromo41(int, java.lang.String, float, float, int))(int c, java.lang.String d, float pu, float reduc, int qm)           constructeur avec paramètres, fait appel à la classe mère Article avec super() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| int | [**getQuantiteMin**](about:blankArticlePromo41.html#getQuantiteMin())()           getteur de la quantité minimale nécessaire pour apliquer la réduction |
| float | [**getReduction**](about:blankArticlePromo41.html#getReduction())()           getteur de la réduction de l'article |
| float | [**prixFacture**](about:blankArticlePromo41.html#prixFacture(int))(int quantité)           Calcule le prix totale de l'article actuel avec le paramètre de la quantité désiré |
| void | [**setQuantiteMin**](about:blankArticlePromo41.html#setQuantiteMin(int))(int nouvelle)           setteur de la quantité minimale |
| void | [**setReduction**](about:blankArticlePromo41.html#setReduction(float))(float nouvelle)           setteur de la réduction |
| java.lang.String | [**stringFacture**](about:blankArticlePromo41.html#stringFacture())()           retourne un morceau de String pour afficher sur une facture pour les articles en promotion |
| java.lang.String | [**toString**](about:blankArticlePromo41.html#toString())()           notion de polymorphisme- renvoie un String qui est constitué du toString() de la classe Article, et ajoute les autres attributs à ce String |

c. **Le scénario de la classe Client :**

Le scénario du client est le même que dans la série 3.2, or l’utilisateur peux maintenant ajouter des articles en promotion, afficher le catalogue des articles en promotion ou acheter des articles en promotion.

d**. Les apports du cours :**

Héritage – l’utilisation de l’héritage ce fait avec l’extension

public class fille extends mere{}

la méthode super() qui appelle le constructeur ou la méthode de la classe mère.

Le polymorphisme- qui utilise le même nom d’une méthode pour les objets de la classe fille et la classe mère, et ne renvoie pas le même résultat.

La méthode prixfacture(int quantite) aqui calcule le prix totale d’une ligne de commande pour bien maîtriser les calculs de prix de promotion et hors promotion (cette méthode-là est aussi un exemple de polymorphisme.

J’ai ajouté aussi la méthode stringFacture() qui renvoie la petite partie qui doit figurer sur les factures pour les articles en promotion.

L’utilisation de « instanceof » pour savoir si un objet parvient de la classe fille.

e. **Conclusion :**

Je connaissais déjà les notions d’héritage, ceci dit, je trouve très pratique et naturel d’abordé le sujet de l’héritage avec les articles en promotion, ça permet d’avoir une image plus concrète du polymorphisme et de la syntaxe de l’héritage.

J’ai choisi pendant cette série de voir comment améliorer l’affichage des joptionPanel en ajoutent un JtextArea pour les affichages et une méthode Tools.calculeEspace(String avant, int largeur) qui prend une chaîne de caractères , et calcule le nombre de caractères, si ce nombre est inférieur au paramètre « largeur », il ajoute des espaces et renvoie la nouvelle chaîne de caractère.

**NFA019 : Série 4.2- Morane Gruenpeter**

a. La série 4.2 consiste à créer une classe abstraite et deux interfaces. C’est la première rencontre avec ces concepts en NFA019.

**La classe abstraite-** Une classe abstraite est une classe pour laquelle on ne peut pas créer d'instance, donc la classe abstraite seule n'a pas de sens! Aussi la classe abstraite avec une seule classe fille n'a pas de sens... l’intérêt d’une classe abstraite est de définir un comportement commun pour les classes qui sont dérivées de la classes abstraite. Les méthodes abstraites sont méthodes polymorphes que l’on impose. Elles doivent être implémentées par ses classes filles obligatoirement. Les méthodes dans la classe abstraite peuvent contenir du code, celle-ci est une des différences entre a classe abstraite et l’interface.

**L’interface-** peut contenir seulement des variables constantes finales et des méthodes abstraite sans code, qui doivent être implémenté par les classes qui appartiennent à l’interface. Le principe est similaire aux classes abstraites, or l’interface n’est pas une classe, donc il n’y a pas le concept d’héritage, dans chaque classe qui implémente l’interface on doit réécrire la totalité du code, avec la signature de la méthode qui doit être respecté.

b. Pour cette série on crée une classe abstraite AbstraitArticle, et deux interfaces ; InterfaceGestion et InterfaceStructure. Ci-dessous les classes qui sont modifié par ce changement et la documentation des classes ajoutées.

Les classes qui ont été modifiées :

* Articles42 : contient seulement les méthodes obligatoire : toString() et facturePrix(). Le reste des méthodes et attributs sont dans la classe abstraite.
* Client42 : ajout de new pour aller vers les classes de gestion car maintenant les méthodes ne sont plus définies en static.
* GestionTableDesArticles42 :

1. Standardisation des méthodes en @Override avec la bonne signature qui figure dans l’InterfaceGestion
2. Changement de tous les Articles en AbstraitArticle.

* GestionTableDesCommandes42 :

1. Standardisation des méthodes en @Override avec la bonne signature qui figure dans l’InterfaceGestion
2. Changement de tous les Articles en AbstraitArticle.
3. Ajout d’une méthode ajouter() qui crée une instance UneCommande, renvoie cette instance avec la table des articles et l’objet o vers la gestion d’une commande. Au retour du gestionnaire d’une commande, elle vérifie si la commande est vide, sinon elle l’ajoute à la table des commandes.

* GestionDuneCommande42 :

1. Standardisation des méthodes en @Override avec la bonne signature qui figure dans l’InterfaceGestion
2. Suppression du traitement de l’ajout de la commande dans la table des commandes.

* UneCommande42 :

1. Standardisation des méthodes en @Override avec la bonne signature qui figure dans l’InterfaceStructure

* LigneDeCommande42 :

1. Changement de tous les Articles en AbstraitArticle(dans la méthode facturer()).

* Tools :rien à changer.
* iPane.ES : rien à changer.

**Class AbstraitArticle<TypeCode>**

java.lang.Object

**AbstraitArticle<TypeCode>**

public abstract class **AbstraitArticle<TypeCode>**extends java.lang.Object

|  |  |
| --- | --- |
| **Field Summary** | |
| protected  [TypeCode](about:blankAbstraitArticle.html" \o "type parameter in AbstraitArticle) | [**codeArticle**](about:blankAbstraitArticle.html#codeArticle) |
| protected  java.lang.String | [**designation**](about:blankAbstraitArticle.html#designation) |
| protected  float | [**prixUnitaire**](about:blankAbstraitArticle.html#prixUnitaire) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**AbstraitArticle**](about:blankAbstraitArticle.html#AbstraitArticle())()           Constructeur par defaut |  |
| [**AbstraitArticle**](about:blankAbstraitArticle.html#AbstraitArticle(TypeCode, java.lang.String, float))([TypeCode](about:blankAbstraitArticle.html" \o "type parameter in AbstraitArticle) c,java.lang.String d,float pu)           Constructeur avec param |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| [TypeCode](about:blankAbstraitArticle.html" \o "type parameter in AbstraitArticle) | [**getCode**](about:blankAbstraitArticle.html#getCode())()           getter du code article |
| java.lang.String | [**getDesignation**](about:blankAbstraitArticle.html#getDesignation())()           getter de la désignation de l'article |
| double | [**getPrixUnitaires**](about:blankAbstraitArticle.html#getPrixUnitaires())()           getter du prix unitaires de l'article |
| abstract  float | [**prixFacture**](about:blankAbstraitArticle.html#prixFacture(int))(int quantite)           Méthode abstraite- implémentation obligatoire. |
| void | [**setCode**](about:blankAbstraitArticle.html#setCode(TypeCode))([TypeCode](about:blankAbstraitArticle.html" \o "type parameter in AbstraitArticle) c)           setter du code de l'article |
| void | [**setDesignation**](about:blankAbstraitArticle.html#setDesignation(java.lang.String))(java.lang.String d)           setter de la designation de l'article |
| void | [**setPrixUnitaire**](about:blankAbstraitArticle.html#setPrixUnitaire(float))(float pu)           setter du prix unitaire de l'article |
| abstract  java.lang.String | [**toString**](about:blankAbstraitArticle.html#toString())()           Méthode abstraite- implémentation obligatoire.  Retourne la chaîne de caractère qui désigne l’article |

**Class Article42**

java.lang.Object

AbstraitArticle<java.lang.Integer>

**Article42**

public class **Article42**extends AbstraitArticle<java.lang.Integer>

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class** |
| codeArticle, designation, prixUnitaire |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Article42**](about:blankArticle42.html#Article42())()           Constructeur par defaut |  |
| [**Article42**](about:blankArticle42.html#Article42(java.lang.Integer, java.lang.String, float))(java.lang.Integer c, java.lang.String d, float pu)           constructeur avec paramètres, fait appel à la classe abstraite AbstraitArticle |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| float | [**prixFacture**](about:blankArticle42.html#prixFacture(int))(int quantite)           Méthode abstraite- implementation obligatoire. |
| java.lang.String | [**toString**](about:blankArticle42.html#toString())()           Méthode abstraite- implementation obligatoire. |

|  |
| --- |
| **Methods inherited from class** |
| getCode, getDesignation, getPrixUnitaires, setCode, setDesignation, setPrixUnitaire |

|  |
| --- |
| **Methods inherited from class** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

**Class ArticlePromo42**

java.lang.Object

AbstraitArticle<java.lang.Integer>

**ArticlePromo42**

public class **ArticlePromo42**extends AbstraitArticle<java.lang.Integer>

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class** |
| codeArticle, designation, prixUnitaire |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ArticlePromo42**](about:blankArticlePromo42.html#ArticlePromo42())()           constructeur par défaut |  |
| [**ArticlePromo42**](about:blankArticlePromo42.html#ArticlePromo42(java.lang.Integer, java.lang.String, float, float, int))(java.lang.Integer c, java.lang.String d, float pu, float reduc, int qm)           constructeur avec paramètres, fait appel à la classe abstraite AbstraitArticle avec super() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| int | [**getQuantiteMin**](about:blankArticlePromo42.html#getQuantiteMin())()           getter de la quantité minimale nécessaire pour appliquer la réduction |
| float | [**getReduction**](about:blankArticlePromo42.html#getReduction())()           getter de la réduction de l'article |
| float | [**prixFacture**](about:blankArticlePromo42.html#prixFacture(int))(int quantite)           Méthode abstraite- implementation obligatoire. |
| void | [**setQuantiteMin**](about:blankArticlePromo42.html#setQuantiteMin(int))(int nouvelle)           setter de la quantité minimale |
| void | [**setReduction**](about:blankArticlePromo42.html#setReduction(float))(float nouvelle)           setter de la réduction |
| java.lang.String | [**stringFacture**](about:blankArticlePromo42.html#stringFacture())()           retourne un morceau de String pour la facture d'article en promotion |
| java.lang.String | [**toString**](about:blankArticlePromo42.html#toString())()           Méthode abstraite- implémentation obligatoire. |

|  |
| --- |
| **Methods inherited from class** |
| getCode, getDesignation, getPrixUnitaires, setCode, setDesignation, setPrixUnitaire |

|  |
| --- |
| **Methods inherited from class** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

**Interface InterfaceGestion<TypeObjet1,TypeObjet2,TypeObjet3>**

public interface **InterfaceGestion<TypeObjet1,TypeObjet2,TypeObjet3>**

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**ajouter**](about:blankInterfaceGestion.html#ajouter(TypeObjet1, TypeObjet2, TypeObjet3))([TypeObjet1](about:blankInterfaceGestion.html) tab1, [TypeObjet2](about:blankInterfaceGestion.html) tab2, [TypeObjet3](about:blankInterfaceGestion.html) tab3) |
| void | [**editer**](about:blankInterfaceGestion.html#editer(TypeObjet1, TypeObjet2, TypeObjet3))([TypeObjet1](about:blankInterfaceGestion.html) tab1, [TypeObjet2](about:blankInterfaceGestion.html) tab2, [TypeObjet3](about:blankInterfaceGestion.html) tab3) |
| int | [**menuChoix**](about:blankInterfaceGestion.html#menuChoix(TypeObjet1))([TypeObjet1](about:blankInterfaceGestion.html) tab1)             je suis obligé de mettre une paramètre car pour la gestion d’une commande, j’ajoute le numéro de commande pour laquelle on effectue le choix |
| void | [**menuGeneral**](about:blankInterfaceGestion.html#menuGeneral(TypeObjet1, TypeObjet2, TypeObjet3))([TypeObjet1](about:blankInterfaceGestion.html) tab1, [TypeObjet2](about:blankInterfaceGestion.html) tab2, [TypeObjet3](about:blankInterfaceGestion.html) tab3) |
| void | [**supprimer**](about:blankInterfaceGestion.html#supprimer(TypeObjet1, TypeObjet2, TypeObjet3))([TypeObjet1](about:blankInterfaceGestion.html) tab1, [TypeObjet2](about:blankInterfaceGestion.html) tab2, [TypeObjet3](about:blankInterfaceGestion.html) tab3) |

**Interface InterfaceStructure<TypeCle,TypeObjet>**

**Type Parameters:**

TypeCle -

TypeObjet -

public interface **InterfaceStructure<TypeCle,TypeObjet>**

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**ajouter**](about:blankInterfaceStructure.html#ajouter(TypeObjet))([TypeObjet](about:blankInterfaceStructure.html" \o "type parameter in InterfaceStructure) objet) |
| [TypeObjet](about:blankInterfaceStructure.html" \o "type parameter in InterfaceStructure) | [**retourner**](about:blankInterfaceStructure.html#retourner(TypeCle))([TypeCle](about:blankInterfaceStructure.html" \o "type parameter in InterfaceStructure) cle) |
| void | [**supprimer**](about:blankInterfaceStructure.html#supprimer(TypeCle))([TypeCle](about:blankInterfaceStructure.html" \o "type parameter in InterfaceStructure) cle) |
| int | [**taille**](about:blankInterfaceStructure.html#taille())() |

c. **Le scénario de la classe Client :**

Le scénario du client ne change pas pour cette série.

d**. Les apports du cours :**

**Type code :** la création d’un type générique qui n’est pas définie pour la classe abstraite ou l’interface et est définie pour l’instance directement. Comme pour les hashtable par exemple, où on peut placer différent type d’élément après que l’on désigne ce type en instanciant la table.

Dans notre cas, on a des Articles avec un code qui est la clé de l’article définie en Integer, on pourrait maintenant facilement le changer en String ou même créer un nouvel objet Code.

**Protected :** l’utilisation des attributs en état « protected » est important car il faut que les classes filles aient accès aux attributs de la classe mère.

**Signature d’une méthode dans l’interface :** Il est important de garder une signature générique qui peut avoir une utilisation multiple avec différent objet et différent type.

J’ai remarqué qu’il fallait aussi ajouter throws exception en signature de l’interface, comme j’ai définie pour la méthode menuChoix qui relève une exception si l’utilisateur annule sa demande.

e. **Conclusion :**

Pour la première fois je trouve les notions de classe abstraite et d’interface vraiment claire. Je savais utiliser l’interface mais je ne comprenais pas exactement les avantages de cette utilisation. Après le cours les définitions de chaque option : héritage, classe abstraite ou interface ; sont très explicite et cela donne un grand avantage de maîtrise du programme et de son organisation.

« TypeCode » est la notion la plus importante que j’ai apprise en cours jusqu’à présent. Je ne savais pas que c’était possible et je ne l’avais même pas imaginé comme possibilité. Je suis enchanté de l’avoir découvert.