Particularités sur les données

Données constantes, objets immuables ou non modifiables

Données 'globales' / fonctions 'globales'

Faculté des sciences, Université de Montpellier
Module HAI717I - Programmation par objets

Données et opérations globales / constantes

Dans ce cours, nous étudions comment mieux gérer certaines informations dans les classes :

- se placer au bon niveau (description de classe versus description d'instance)
- éviter des redondances de données
- sortir les valeurs littérales du code des méthodes
- définir des constantes et des objets non modifiables

Données et opérations globales à une classe

Certains traitements et données sont relatifs à une classe
plutôt que relatifs à une instance.

Exemple dans une classe `TypeProdAlimentaire`, informations relatives à ...

une instance	la classe
nom	taux TVA
référence	taux de conversion KJoules vers KCal
prix hors taxe	nombre de références de types de produits
nombre de KJoules	
date de création de la référence	

Données et opérations globales à une classe

Les données et opérations globales à une classe sont introduites en Java par le mot-clef static

Exemple dans une classe `TypeProdAlimentaire`, seront static les informations relatives à ...

~~une instance~~	la classe
~~nom~~	taux TVA
~~référence~~	taux de conversion KJoules vers KCal
~~prix hors taxe~~	nombre de références de types de produits
~~nombre de KJoules~~	
~~date de création de la référence~~	

Données et opérations globales à une classe versus propres à une instance

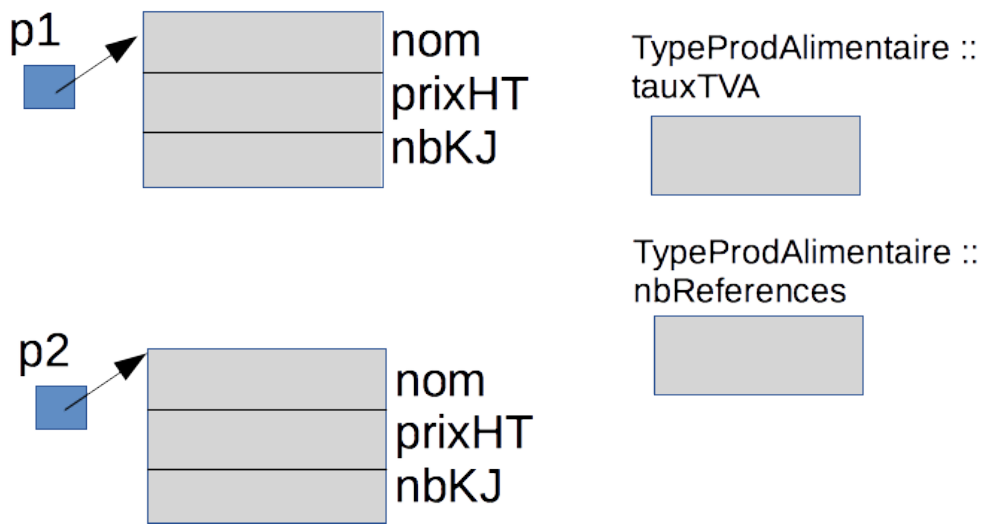
une instance	la classe
nom	taux TVA
référence	*taux de conversion KJoules vers KCal*
prix hors taxe	nombre de références de types de produits
nombre de KJoules	
date de création de la référence	

Commençons par écrire les données relatives à une instance et les données relatives à la classe dont la valeur peut changer ...

```
public class TypeProdAlimentaire{
   private String nom ;
   private double prixHT, nbKJ ;
   private static double tauxTVA ;
   private static int nbReferences ;
}
```

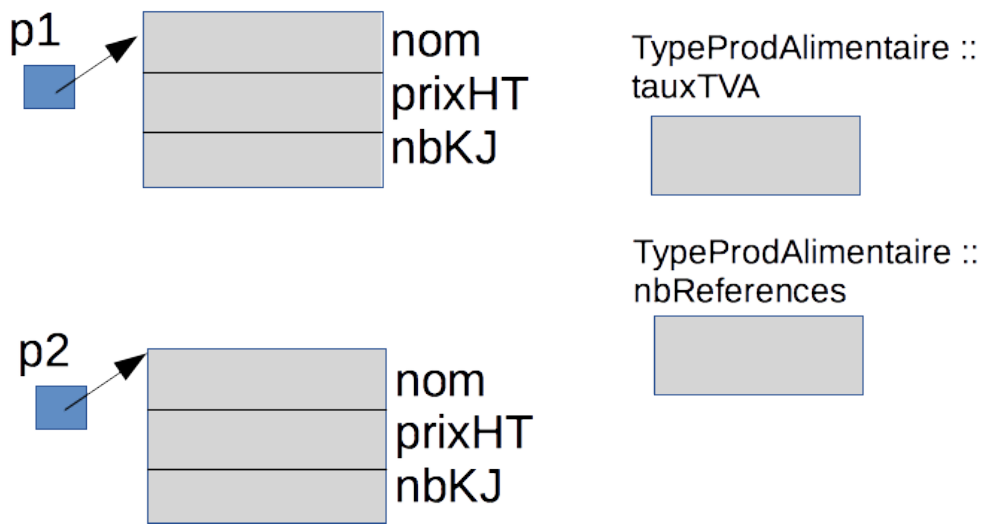
Données et opérations globales à une classe versus propres à une instance

```
public class TypeProdAlimentaire{
   private String nom ;
   private double prixHT, nbKJ ;
   private static double tauxTVA ;
   private static int nbReferences ;
}
... main ...{
TypeProdAlimentaire p1 = new TypeProdAlimentaire() ;
TypeProdAlimentaire p2 = new TypeProdAlimentaire() ;
}
```



Données et opérations globales à une classe versus propres à une instance

- Les données d'instance (ex. prixHT) existent en autant d'exemplaires qu'il y a d'instances
- Les données globales à la classe (ex. tauxTVA) existent en 1 seul exemplaire



Données et opérations globales à une classe

Impact dans les constructeurs

Ces données seront très rarement passées en paramètre à un constructeur, car leur initialisation n'est pas connectée à celle d'un objet

```
public class TypeProdAlimentaire{
   private String nom ;
   private double prixHT, nbKJ ;
   private static double tauxTVA ;
   private static int nbReferences ;

   public TypeProdAlimentaire(String n, double p, double nbKJ
                              ~~double tauxTVA, int nbReferences~~){
      ......
   }
}
```

Données et opérations globales à une classe

Appel

On les appelle avec le préfixe nomClasse.nomAttribut (dans un contexte où ils sont visibles)

```
public class TypeProdAlimentaire{
   private String nom ;
   private double prixHT, nbKJ ;
   private static double tauxTVA ;
   private static int nbReferences ;


...
      ... TypeProdAlimentaire.tauxTVA ...
      ... TypeProdAlimentaire.nbReferences ...
....
}
```

Données et opérations globales à une classe

Manipulation par des méthodes static

```
public class TypeProdAlimentaire{
...
   private static double tauxTVA ;
   private static int nbReferences ;

   public static double getTauxTVA(){return TypeProdAlimentaire.tauxTVA ;}

   public static void setTauxTVA(double tauxTVA) {
      TypeProdAlimentaire.tauxTVA = tauxTVA ;
   }

   public static int getNbReferences() {
      return TypeProdAlimentaire.nbReferences ;
   }

   public static void setNbReferences(int nbReferences) {
      TypeProdAlimentaire.nbReferences = nbReferences ;
   }

}
```

Données et opérations globales à une classe

Appel

On les appelle avec le préfixe nomClasse.nomMéthode

```
public class TypeProdAlimentaire{
   private String nom ;
   private double prixHT, nbKJ ;
   private static double tauxTVA ;
   private static int nbReferences ;
   ...
   public static void setTauxTVA(double tauxTVA) {
      TypeProdAlimentaire.tauxTVA = tauxTVA ;
   }
}

... main ...
      ... TypeProdAlimentaire.setTauxTVA(20) ; ...
      ....
```

Données et opérations globales à une classe

Appel

On les appelle avec le préfixe nomClasse.nomMéthode
Les données et opérations ordinaires sont appelées sur une instance

```
public class TypeProdAlimentaire{
   private String nom ;
   private static double tauxTVA ;
   ...
   public void setNom(String nom){this.nom =nom ;}     ...
   public static void setTauxTVA(double tauxTVA) {
      TypeProdAlimentaire.tauxTVA = tauxTVA ;
   }
}
... main ...
      TypeProdAlimentaire.setTauxTVA(20) ;
      TypeProdAlimentaire p1 = new TypeProdAlimentaire() ;
      p1.setNom("chicorée") ;
```

Données non modifiables

Données non modifiables :

- constantes
- constantes globales (static)
- objets non modifiables

Constantes

En Java, la notion qui existe est *donnée qui ne peut être initialisée qu'une fois*. Elles sont introduites avec final.

final s'applique sur :

- les attributs
- les paramètres
- les variables locales
- Et ... *les classes pour empêcher de les spécialiser par une sous-classe (ex. String est final et ne peut avoir de sous-classe)*

Attribut constant

Un attribut constant est initialisé :

- soit au moment de la déclaration
- soit dans les constructeurs

Il ne dispose pas d'accesseur en modification (de type `set`).

```
public class TypeProdAlimentaire {

   // exemple d'initialisation au moment de la déclaration

   private final Date dateCreation = new Date() ; // donne la date du jour

}
```

Attribut constant

Un attribut constant est initialisé :

- soit au moment de la déclaration
- soit dans les constructeurs

Il ne dispose pas d'accesseur en modification (de type `set`).

```
public class TypeProdAlimentaire {
   private final String reference;

   public TypeProdAlimentaire() {
      // la reference d'un produit est donnee par "ref" suivi du numero courant
      // donne par le nombre de produit deja references
      this.reference = "ref"+TypeProdAlimentaire.nbReferences;
      TypeProdAlimentaire.nbReferences++;
   }
}
```

Constantes globales (static)

Constantes globales (static), quelques exemples :

- le nombre de côtés d'un carré
- le taux de conversion Kjoules vers Kcal
- la valeur de π

  Dans l'API Java : public class Math{public static final double Pi=3.14 ;}

  La classe Math contient également des méthodes static utilitaires comme `ceil, floor, abs`

Constantes globales (static)

Les constantes globales (static) seront souvent `public`
Elles ne risquent rien puisqu'on ne peut en changer la valeur.

```
public class TypeProdAlimentaire {
   public static final double tauxConversionKJ2KCal = 0.239;
}
```

Les objets peuvent-ils être constants ?

Les objets peuvent-ils être constants ?
Premier cas : les objets des classes standards comme celles que nous avons vues.

On introduit tout d'abord une classe `Magasin` toute simple

```
public class Magasin {
   private String nom ;
   public String getNom() {
      return this.nom ;
   }
   public void setNom(String nom) {
      this.nom = nom ;
   }
}
```

Les objets peuvent-ils être constants ?

Les objets peuvent-ils être constants ?
Premier cas : les objets des classes standards comme celles que nous avons vues.
On crée une classe pour représenter des chariots alimentaires attachés à un magasin

```
public class ChariotAlimentaire {
   // on pourra changer les attributs du magasin, mais
   // pas mettre le chariot dans un autre magasin

   private final Magasin magasin = new Magasin();

   public void pourTester() {
      // ne peut fonctionner
      magasin = new Magasin();
      // mais fonctionne
      magasin.setNom("Biocoop");
   }
}
```

Donc l'objet n'est pas constant ! C'est la référence qui est constante.

Objets immuables

Objets immuables en Java

- Quelques classes représentent des objets immuables
  par exemple String, Integer
- Les objets peuvent être transformés en objets immuables ou non modifiables
  - Des techniques de clonage, des classes final, hors du programme du module
  - Des méthodes pour rendre les listes immuables ou non modifiables

Nota : pour avoir une chaîne modifiable

```
StringBuffer s = new StringBuffer("abc");
s.append("def");
System.out.println(s);
```

Listes immuables

Depuis Java 9 : On peut créer des listes immuables avec des méthodes de fabrication statiques.

Par exemple avec List.of

```
// Exemple 1 - création d'une liste de paniers gourmands

List<TypeProdAlimentaire> panierGourmand
      = new ArrayList<TypeProdAlimentaire>();
panierGourmand.add(new TypeProdAlimentaire("chocolat noir Perou", 20.0, 4000));
panierGourmand.add(new TypeProdAlimentaire("nougat Montélimar", 30.0, 3000));
panierGourmand.add(new TypeProdAlimentaire("café Guatemala", 40.0, 10));

// panierGourmandImmV1 est une liste de paniers gourmands
// contenant 1 seul panier

List<List<TypeProdAlimentaire>> listPanierGourmandImmV1
      = List.of(panierGourmand);
```

Listes immuables

List.of

```
(...)
// on peut modifier un élément interne à listPanierGourmandImmV1

panierGourmand.add(new TypeProdAlimentaire("thé blanc groseilles", 40.0, 10));

// mais pas listPanierGourmandImmV1 lui-même
/*
 * Le code ci-dessous provoque une erreur d'exécution
 * java.lang.UnsupportedOperationException
 */

listPanierGourmandImmV1.add(new ArrayList<TypeProdAlimentaire>());
```

Listes immuables

On peut créer directement le panier gourmand immuable avec List.of en donnant les valeurs lors de la création.

```
// Exemple 2 panierGourmandImmV2 est un panier gourmand contenant 2 produits

List<TypeProdAlimentaire> panierGourmandImmV2
    = List.of(new TypeProdAlimentaire("chocolat noir Perou", 10.0, 4000),
new TypeProdAlimentaire("nougat Montélimar", 10.0, 3000));

/*
 * Le code ci-dessous provoque une erreur d'exécution
 * java.lang.UnsupportedOperationException
 */

panierGourmandImmV2.add(new TypeProdAlimentaire("café Guatemala", 10.0, 3000));
```

Pour aller plus loin ; https://docs.oracle.com/javase/9/core/creating-immutable-lists-sets-and-maps.htm#JSCOR-GUID-202D195E-6E18-41F6-90C0-7423B2C9B381

Listes non modifiables

Il existe également des méthodes pour donner une vue non modifiable sur une liste et permettre l'écriture d'accesseurs sans danger

```
public class ChariotAlimentaire {
   private ArrayList<TypeProdAlimentaire> contenu = new ArrayList<>();

   // A éviter car on ne contrôlera rien de ce qui sera mis dedans
   // En effet on donne accès à la référence de la liste :
   public ArrayList<TypeProdAlimentaire> getContenu() {
      return contenu;
   }

   // A préférer car retourne une liste non modifiable :
   public ArrayList<TypeProdAlimentaire> getContenuI() {
      return (ArrayList<TypeProdAlimentaire>)
                  Collections.unmodifiableList(contenu);
   }
}
```

Listes non modifiables

Des méthodes pour donner une vue non modifiable sur une liste et permettre l'écriture d'accesseurs sans danger public class ChariotAlimentaire {

```
  private ArrayList<TypeProdAlimentaire> contenu = new ArrayList<>();
  // A éviter car on donne accès à la référence de la liste
  public ArrayList<TypeProdAlimentaire> getContenu() {
    return contenu;
  }
  // A préférer car retourne une liste non modifiable
  public ArrayList<TypeProdAlimentaire> getContenuI() {
    return (ArrayList<TypeProdAlimentaire>)
              Collections.unmodifiableList(contenu);
  }
}
.... main ....
  ChariotAlimentaire chariot = new ChariotAlimentaire();
  chariot.getContenu().add(new TypeProdAlimentaire());
  // provoque une erreur d'exécution :
  chariot.getContenuI().add(new TypeProdAlimentaire());
```

Synthèse

- Données et opérations globales : static
- Données initialisées une seule fois : final
- Données globales initialisées une seule fois : static final
- Listes immuables : List.of
- Vues non modifiables sur des listes : Collections.unmodifiableList