INF1256 Informatique pour les sciences de la gestion

–Introduction aux notions de classe et objet–

Johnny TSHEKE, Ing. Jr.

Université du Québec à Montréal
Département d'Informatique
tsheke_shele.johnny@uqam.ca

Séance 10

1 Introduction

2 Définir et construire ses propres classes

3 Notion d'instanciation d'objets

4 Distinction entre une classe et un objet

5 Accès aux données (propriétés, méthodes)

1 Introduction

2 Définir et construire ses propres classes

3 Notion d'instanciation d'objets

4 Distinction entre une classe et un objet

5 Accès aux données (propriétés, méthodes)

Quelques notions sur l'OO

OO : Orienté Objet (Object-Oriented) est un style de programme manipulant les classes et les objets

classe (class) : un prototype d'objet défini par le programmeur. Tout objet de la classe est caractérisé par ses attributs

attributs : des variables et/ou des méthodes

instance : un objet précis d'une classe

méthode : une fonction définie dans une classe

variable de classe : variable partagée par toutes les instances de la classe. Dans la classe avec static mais hors méthode

variable d'instance : Chaque instance a sa propre copie de la variable. Dans la classe sans le mot ~~static~~ mais hors méthode

héritage : transfert des caractéristiques d'une classe (`super classe`) à ses dérivées (`sous classes`). Toutes les classes descendent de `Object`

⇒ Pour plus d'info :

`https://docs.oracle.com/javase/tutorial/java/javaOO/`

1 Introduction

2 Définir et construire ses propres classes

3 Notion d'instanciation d'objets

4 Distinction entre une classe et un objet

5 Accès aux données (propriétés, méthodes)

Définition d'une classe simple

```
class NomDeLaClasse{

  <déclarations des champs et constructeurs>
  <déclarations méthodes>
}
```

- Première lettre du nom de la classe en majuscule
- Les variables de classe et les méthodes sont optionnelles mais c'est mieux d'en avoir au moins un attribut (variable de classe ou méthode), sinon pas trop de sens

Définition d'une classe – forme plus générale

```
modifier class NomClasse extends NomSuprClasse  implements NomsInterfacesSeparesParDesVirgules{

   //contenu de la classe
  }
```

- modifier est optionnelle et peut être `public`, `private`, etc.
- extends NomSuprClasse optionnelle si héritage
- implements Interfaces optionnelle si implémente interface(s)

Contrôle d'accès aux éléments d'une classes

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

- Niveau classe : `public` si classe accessible de partout. Si pas de protection explicite alors accessible seulement dans le package.
- Niveau attribut de classe : `public` , `ptotected`, `private`, défaut (pas de protection explicite) Telle que illustré dans le tableau

⇒ voir `https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html`

mot clé `this` dans une méthode ou un constructeur

Le mot clé this

- permet de faire référence à l'objet lui-même. ex :

```
this.nomVariable;
this.nomMethode();
```

- peut-être utilisé dans une méthode ou un constructeur
- Ne peut pas être utilisé dans une méthode (static) de classe

`static` :variables et méthodes de classe

- `static` dans la déclaration d'une variable globale
- `static` dans la déclaration d'une méthode
- `NomClasse.nomVariableClasse;` ou `NomObject.nomVariableClasse;` (accès variable de classe)
- `NomClasse.nomMethodeClasse(args);` ou `NomObject.nomMethodeClasse(args);` (accès méthode de classe)
- Une méthode de classe (static) ne peut pas accéder directement à une variable de classe ou une méthode de classe. Il faut créer une instance (un objet)
- Une méthode de classe (static) ne peut pas utiliser le mot clé this

Exemple de déclaration d'une classe

Exemple de la classe de citoyen canadien : Canadien.java

```
package inf1256s10;
// Johnny Tsheke @UQAM -- INF1256

public class Canadien {
        public static String  nomPays = "Canada";//variable de classe
        private String nom ="";//nom de la persoonne
        String getNom(){
                return(nom);
        }

        String getNomPays(){
                return(nomPays);
        }
}
```

Méthode `main`

En Java,

- chaque application doit avoir une méthode `main` (dans une des classes)

```
public static void main(String[] args){
//contenur e la mthode
}
```

- `args` est un tableau des arguments éventuel à passer au lancement du programme (application)
- C'est la méthode qui s'exécute par défaut au lancement du programme

constructeur

- constructeur→ méthode spéciale appelée à la création d'une nouvelle instance de la classe. Elle permet, notamment, de faire des initialisations
- Porte le même nom que la classe mais n'a pas de type de retour. même pas void
- les paramètres d'un constructeur permettent de passer des arguments nécessaires à la création d'une nouvelle instance de la classe
- une classe peut avoir plusieurs constructeus. Le constructeur a exécuté sera déterminer par les paramètre au moment de l'instanciation
- Les classe sont présumées avoir un constructeur par défaut qui ne reçoit aucun argument
- instanciation → création d'une nouvelle instance (un nouvel objet) d'une classe

Exemple classe avec constructeur, méthodes et main

Exemple de la classe Canadien2 avec constructeur : Canadien2.java

```
package inf1256s10;
// Johnny Tsheke @UQAM -- INF1256

public class Canadien2 {
        protected static String  nomPays = "Canada";//variable de classe
        private String nom ="";//nom de la persoonne
        private String prenom ="";
        public Canadien2(String firstName, String givenName){//constructeur
                prenom =firstName;
                nom = givenName;
        }
        public String getNom(){//méthode --getter
                return(nom);
        }
        public String getPrenom(){//méthode --getter
                return(nom);
        }

        public static String getNomPays(){//méthode
                return(nomPays);
        }
        public static void main(String[] args){
                System.out.println("Le nom du pays est: "+ Canadien2.getNomPays());
        }
}
```

1 Introduction

2 Définir et construire ses propres classes

3 Notion d'instanciation d'objets

4 Distinction entre une classe et un objet

5 Accès aux données (propriétés, méthodes)

Instanciation d'une classe en Java

- instanciation→ création d'un objet en faisant appel à la classe comme si c'était une méthode et en faisant précédé du mot `new`
- Au besoin, on passe les arguments définis dans le constructeur.
- Concrètement, l'instanciation se fait en affectant à une variable, l'appel de la classe comme

```
/* instanciation de la classe Canadien  */
Canadien can = new Canadien()
```

⇒ l'instanciation se fait à l'endroit où on veut créer l'objet. cela peut-être dans la même classe ou ailleurs après importation de la classe

Exemple d'instanciation d'une classe sans constructeur

Exemple de la classe Canadien sans constructeur : Canadien.java

```
package inf1256s10;
// Johnny Tsheke @UQAM -- INF1256

public class Canadien {
        public static String  nomPays = "Canada";//variable de classe
        private String nom ="";//nom de la persoonne
        String getNom(){
                return(nom);
        }

        String getNomPays(){
                return(nomPays);
        }
}
```

Exemple d'instanciantion de la classe Canadien : canadien3.java

```
package inf1256s10;

import inf1256s10.Canadien;//importation
public class Canadien3 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
        Canadien can = new Canadien();//instanciation
        System.out.println("Le nom du pays est: "+ can.nomPays);
        }
}
```

Exemple d'instanciation d'une classe avec constructeur

Exemple d'instanciantion de la classe Canadien2 : canadien4.java

```
package inf1256s10;

import inf1256s10.Canadien2;//importation
public class Canadien4 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
        Canadien2 can = new Canadien2("Charles","Gagnon");//instanciation
        System.out.println("Le nom de la personne est: "+ can.getNom());
        }
}
```

1 Introduction

2 Définir et construire ses propres classes

3 Notion d'instanciation d'objets

4 Distinction entre une classe et un objet

5 Accès aux données (propriétés, méthodes)

classe Vs Objet

classe : prototype, définition

objet : instance de la classe. Quand on instancie une classe, on obtient un objet.

1 Introduction

2 Définir et construire ses propres classes

3 Notion d'instanciation d'objets

4 Distinction entre une classe et un objet

5 Accès aux données (propriétés, méthodes)

Accès au données d'une classe ou d'un objet

- pour accéder au données d'une classe ou d'un objet, on utilise l'opérateur . (point).
- variable de classe : NomClasse.nomVariableDeClasse
- pour un objet : On utilise le nom de la variable qui pointe sur l'objet :
  nomVariableObjet.nomVariableAttribut
  nomVariable.nomMethode(arguments)
- Pour affecter une variable (si autorisé), on peut utiliser l'opérateur point :
  NomClasse.nomVariableDeClasse = valeur
  nomVariableObjet.nomVariableAttribut= valeur
  ou définir une méthode qui reçoit la valeur en argument et fait l'affectation dans la méthode (`un setter`)

Exemple d'instanciation et accès aux données

Exemple de plusieurs instanciations et accès aux données

Exemple de plusieurs instanciations et modification de données

Exemple de modification d'une variable de classe avec le nom de la classe