La Programmation Orientée Objet (POO) avec Java

Module 3 – La création de classes

Objectifs

- Créer des classes java
- Ajouter des attributs d'instance et de classe
- Écrire des constructeurs
- Définir des méthodes d'instance et des méthodes de classe

Déclaration d'une classe

- Une classe est définie dans un fichier portant le même nom que cette classe
- Le nom d'une classe commence par une majuscule

- Une classe est définie dans un package
- Le nommage du package est défini en se basant sur le nom de domaine de la société et le nom du projet
  - Exemple : **`fr.eni.ecole.jeuDeDes`**
- Le répertoire de la classe correspond au nom du package
  - Exemple : **src/fr/eni/ecole/jeuDeDes/**

La création de classes

Déclaration d'une classe

```
package fr.eni.ecole.jeuDeDes;

/**
 * Classe modélisant un dé à jouer
 */
public class De {

}
```

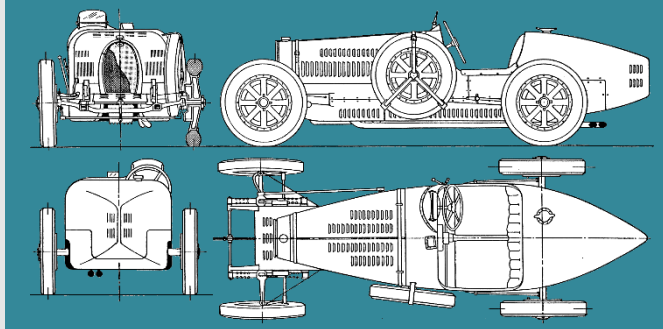
Déclaration du package

Déclaration de la classe

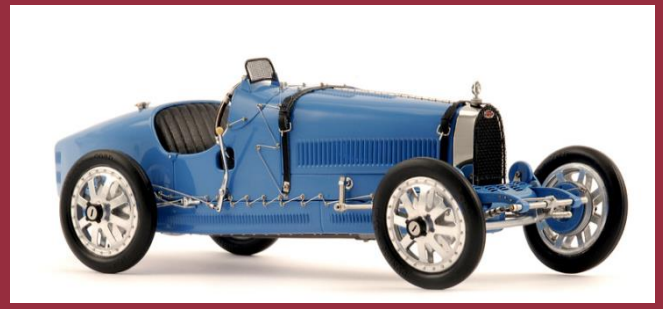
La création de classes

Les attributs d'instance

- Ensemble de valeurs caractérisant une instance d'une classe



Bugatti35
-couleur:String -roueSecours:boolean



Instance1 : Bugatti35	
couleur	"Bleu"
roueSecours	false

Instance2 : Bugatti35	
couleur	"Bordeau"
roueSecours	true

classe

Instances

La création de classes

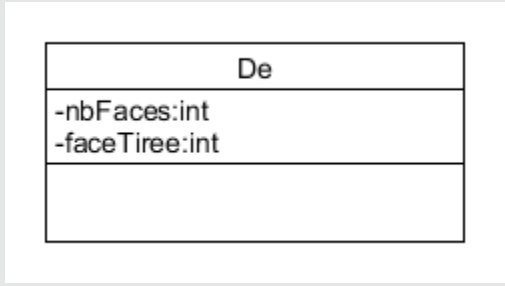
Les attributs d'instance

```
package fr.eni.ecole.jeuDeDes;

/**
 * Classe modélisant un dé à jouer
 */
public class De {

  private int nbFaces;
  private int faceTiree;

}
```



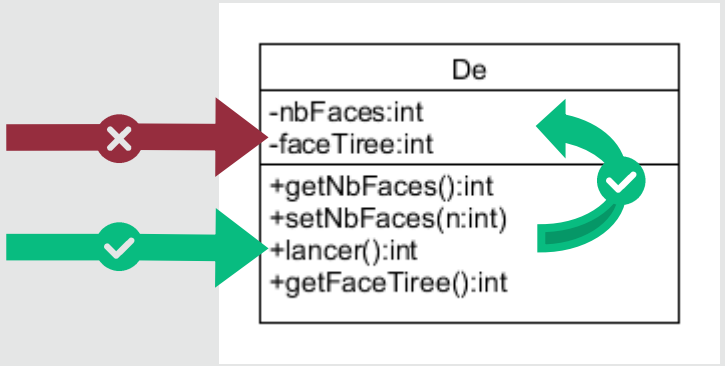
La création de classes

Les visibilités au sein de la classe

Visibilité	Mot clef	Signification
Privée	`private`	Accessible uniquement au sein de la classe
Publique	`public`	Accessible de n'importe où
Package	Ø	Accessible uniquement au sein de ce package
Protégé	`protected`	Accessible au sein de la classe et de ses héritières

Le principe d'encapsulation

- Une classe est responsable de ses données
  - De l'extérieur de la classe, il est impossible de manipuler directement les attributs
  - Par contre, il est possible d'appeler l'une des méthodes qui elle a accès aux attributs



La création de classes

Les méthodes d'instance

```
private int nbFaces;
private int faceTiree;
private static Random rand = new Random();
public int getNbFaces() {
  return this.nbFaces;
}
public void setNbFaces(int nbFaces) {
  this.nbFaces = nbFaces;
}
public int lancer() {
  return this.faceTiree = De.rand.nextInt(this.nbFaces) + 1;
}
public int getFaceTiree() {
  return this.faceTiree;
}
```

Au sein de la classe, le mot clef **this.** permet d'accéder aux éléments d'instance

La création de classes

Les méthodes d'instance

```
private int nbFaces;
private int faceTiree;
private static Random rand = new Random();
public int getNbFaces() {
  return this.nbFaces;
}
public void setNbFaces(int nbFaces) {
  this.nbFaces = nbFaces;
}
public int lancer() {
  return this.faceTiree = De.rand.nextInt(this.nbFaces) + 1;
}
public int getFaceTiree() {
  return this.faceTiree;
}
```

Getter pour l'attribut nbFaces

Setter pour l'attribut nbFaces

Getter pour l'attribut faceTiree

Les méthodes permettant l'accès en lecture sont appelées getter et l'accès en écriture setter

La création de classes

Les méthodes d'instance

```
package fr.eni.ecole.jeuDeDes;

public class TestDe {

  public static void main(String[] args) {
    De monDe = new De();
    monDe.setNbFaces(6);
    do {
      System.out.println("Le dé a fait un " + monDe.lancer());
    } while (monDe.getFaceTiree() != 6);
  }

}
```

```
Le dé a fait un 4
Le dé a fait un 1
Le dé a fait un 2
Le dé a fait un 6
```

La création de classes

Les méthodes d'instance

```
package fr.eni.ecole.jeuDeDes;

public class TestDe {

  public static void main(String[] args) {
    De monDe = new De();
    monDe.nbFaces = 6;
    do {
      System.out.println("Le dé a fait un " + monDe.lancer());
    } while (monDe.getFaceTiree() != 6);
  }
}
```

Interdit en raison de sa visibilité privée

Le constructeur par défaut

- Est présent lorsqu'il n'y a aucun constructeur de défini dans une classe
- Ne prend aucun argument
- Est public

La création de classes

Le constructeur par défaut

```
package fr.eni.ecole.jeuDeDes;

public class TestDe {

  public static void main(String[] args) {
    De monDe = new De();
    monDe.setNbFaces(6);
    do {
      System.out.println("Le dé a fait un " + monDe.lancer());
    } while (monDe.getFaceTiree() != 6);
  }

}
```

Appel au constructeur par défaut car aucun constructeur n'est défini dans la classe De

```
Le dé a fait un 4
Le dé a fait un 1
Le dé a fait un 2
Le dé a fait un 6
```

La création de classes

Le constructeur

- Porte le même nom que la classe
- N'a pas de type de retour

```
public class De {

  private int nbFaces;
  private int faceTiree;
  private static Random rand = new Random();

  public De(int nbFaces) {
    this.setNbFaces(nbFaces);
    this.lancer();
 }
    …
}
```

La création de classes

Le constructeur

```
package fr.eni.ecole.jeuDeDes;

public class TestDe {

  public static void main(String[] args) {
    De monDe = new De();
    monDe.setNbFaces(6);
    do {
      System.out.println("Le dé a fait un " + monDe.lancer());
    } while (monDe.getFaceTiree() != 6);
  }

}
```

Dès qu'un constructeur est défini,
il n'y a plus de constructeur par défaut

La création de classes

Le constructeur

```
package fr.eni.ecole.jeuDeDes;

public class TestDe {

  public static void main(String[] args) {
    De monDe = new De(6);
    do {
      System.out.println("Le dé a fait un " + monDe.lancer());
    } while (monDe.getFaceTiree() != 6);
  }

}
```

```
Le dé a fait un 3
Le dé a fait un 3
Le dé a fait un 3
Le dé a fait un 2
Le dé a fait un 4
Le dé a fait un 6
```

La création de classes

La surcharge de constructeurs

```
/**
 * Constructeur pour créer un dé avec le nombre de faces passé en paramètre
 * @param nbFaces nombre de faces pour le dé à créer
 */
public De(int nbFaces) {
  this.setNbFaces(nbFaces);
  this.lancer();
}

/**
 * Constructeur pour créer un dé à 6 faces
 */
public De() {
  this.setNbFaces(6);
  this.lancer();
}
```

Plusieurs constructeurs peuvent être définis au sein d'une classe

La création de classes

La surcharge de constructeurs

```java
/**
 * Constructeur pour créer un dé avec le nombre de faces passé en paramètre
 * @param nbFaces nombre de faces pour le dé à créer
 */
public De(int nbFaces) {
  this.setNbFaces(nbFaces);
  this.lancer();
}

/**
 * Constructeur pour créer un dé à 6 faces
 */
public De() {
  // appel à l'autre constructeur
  //(obligatoirement la première instruction)
  this(6);
}
```

Plusieurs constructeurs peuvent être définis au sein d'une classe

Un constructeur peut faire appel à un autre constructeur de la classe en utilisant **this()** en première instruction

La création de classes

Les attributs de classe

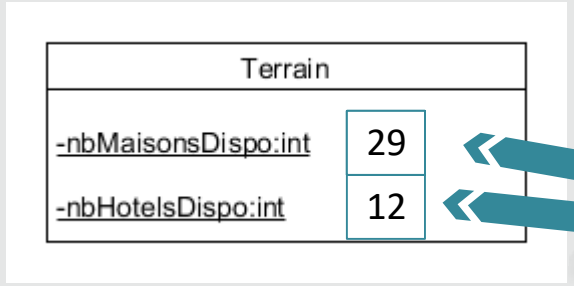
- Valeur commune ou collective
- Défini avec le mot clef **static**

```
public class Terrain {
    …
  private int coutConstruction;
  private int niveauConstruction;

  private static int nbMaisonsDispo = 32;
  private static int nbHotelsDispo = 12;
    …
}
```

La création de classes

Les attributs de classe



Attributs de classe

belleville : Terrain	
coutConstruction	50
niveauConstruction	2

bourse : Terrain	
coutConstruction	150
niveauConstruction	0

Attributs d'instance

lecourbe : Terrain	
coutConstruction	50
niveauConstruction	0

vaugirard : Terrain	
coutConstruction	50
niveauConstruction	0

paix : Terrain	
coutConstruction	200
niveauConstruction	1

Les méthodes de classe

- Méthode dont l'exécution ne dépend pas d'une instance particulière
- Définie avec le mot clef **`static`**
- Une méthode de classe ne peut accéder qu'aux attributs de classe (pas aux attributs d'instance)

```
private static void verifNbFaces(int nbFaces) throws Exception {
  if (nbFaces <= 1)
    throw new Exception("Un dé doit avoir au moins deux faces");
}

public void setNbFaces(int nbFaces) throws Exception {
  De.verifNbFaces(nbFaces);
  this.nbFaces = nbFaces;
}
```

Méthode de classe

Méthode d'instance

La création de classes

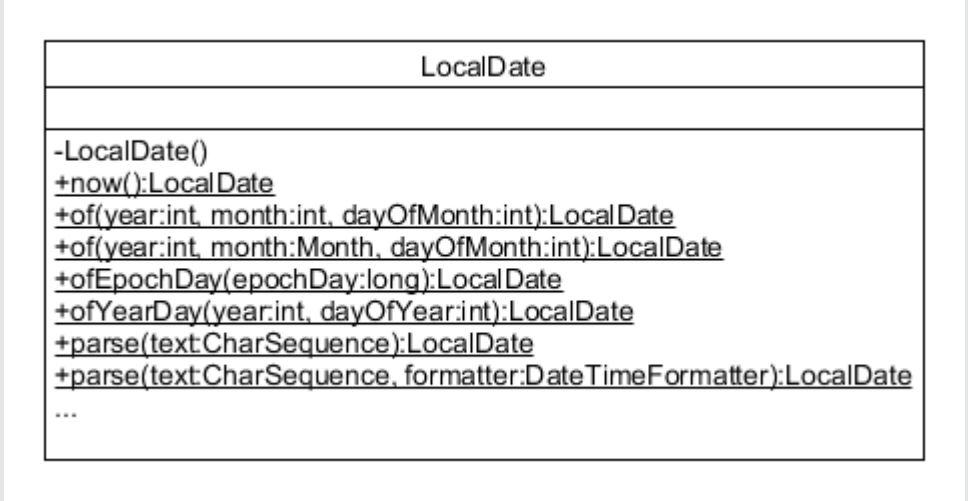
Les méthodes d'instance et méthodes de classe

	Méthode d'instance	Méthode de classe
Mot clef	∅	`static`
Accès aux attributs de classe	✓	✓
Accès aux attributs d'instance	✓	✗
Appel depuis la classe	`this.nomMethodeInstance()` Exemple : `this.lancer()`	`nomClasse.nomMethodeClasse()` Exemple : `De.verifNbFaces(6)`
Appel hors de la classe	`nomInstance.nomMethodeInstance()` Exemple : `de1.lancer()`	

La création de classes

Fabrique par méthode de classe

- Constructeur privé
- Méthodes de classe permettant de récupérer une instance
  - Nommage explicite
  - Constructeur appelé que si nécessaire
  - Retour possible d'un sous type



Les types valeur et les types référence

	type Valeur	type Référence
Quoi ?	types de base (`int`, `float`, `boolean`, `char`…)	tableaux et instances
Passage en paramètre	copie de la valeur Donc modification d'une copie	copie de l'adresse Donc modification de l'original
Retour d'une fonction	retourne la valeur	retourne l'adresse

Démonstration

TP