Les constructeurs

Constructeur: rappel

- Pour instancier une classe, c'est-à-dire créer un objet, on utilise l'opérateur **`new`** qui fait appel à une méthode spéciale de la classe appelée constructeur.
- **Le rôle principal d'un constructeur est d'initialiser les attributs lors de la création d'un objet**.
- **Un constructeur est une méthode spéciale qui porte le même nom que la classe** dans laquelle elle est définie, elle n'a pas de type de retour ni de mot clé `void`.
- Exemple:

```
public class Rectangle {
   private int longueur;
   private int largeur;

   public Rectangle(int longueur, int largeur) {
      this.longueur = longueur;
      this.largeur = largeur;
   }
//..
}
```

Le nom du constructeur est le même que la classe (commence par une majuscule).
Pas de type retour ni `void`.

- Utilisation: si on veut construire un rectangle de longueur 7 et de largeur 4, on utilise l'instruction:

```
Rectangle rect = new Rectangle (7,4);
```

Plusieurs constructeurs

- On peut définir **plusieurs constructeurs dans une classe**. Ces constructeurs **se différencient par le nombre de leurs paramètres ou les types de leurs paramètres**. On dit que le **constructeur est surchargé**.
- Pour déterminer quel constructeur doit être utilisé, l'interpréteur Java regarde, lors de son appel, la liste des paramètres définis dans chaque constructeur.
- Un constructeur sans paramètres est appelé **constructeur par défaut**.

```
public class Rectangle {
   private int longueur;
   private int largeur;

public Rectangle(int longueur, int largeur) {
     this.longueur = longueur;
     this.largeur = largeur;
   }
//Construire un rectangle de type carré
// longueur = largeur
   public Rectangle(int cote) {
      longueur = cote;
      largeur = cote;
   }
// constructeur sans paramètres ou par défaut
   public Rectangle() {
      longueur = 1;
      largeur = 1;
   }
   //...
}
```

```
public class GestionRectangle {

   public static void main(String[] args) {
      //Construire un rectangle de longueur 1 et de largeur 1.
      //Appel du troisième constructeur.

       Rectangle rect1 = new Rectangle();

      //Construire un rectangle de longueur 8 et de largeur 4.
      //Appel du premier constructeur.

      Rectangle rect2 = new Rectangle(8,4);

      //Construire un rectangle de TYPE CARRÉ
      //longueur = largeur = 5.
      //Appel du deuxième constructeur.

       Rectangle rect3 = new Rectangle(5);

   }

}
```

Constructeur par défaut

- Si une classe ne contient aucun constructeur écrit par le programmeur, alors le compilateur Java lui fournit un constructeur (implicite) sans arguments qui consiste à initialiser tous les attributs avec leur valeur par défaut :
  - byte, short, int, long ← 0
  - float, double ← 0.0
  - boolean ← false
  - char ← '\u0000'
  - objet quelconque (incluant String) ← null

On suppose que les attributs ne sont pas initialisés avec des valeurs explicites

- C'est un constructeur par défaut, par défaut.
- **Attention! Dès que nous définissons un constructeur pour une classe, le constructeur par défaut n'existe plus.**
- Exemple: la classe Point2D n'a pas de constructeur. Java lui fournit un constructeur **implicite (par défaut)**.

```
class Point2D {
   private int x;
   private int y;

   public String afficher() {
      return "(" + x + "," + y + ")";
   }
}
```

Équivaut à

```
class Point2D {
   private int x;
   private int y;

   public Point2D() {
      x = 0;
      y = 0;
   }
   public String afficher() {
      return "(" + x + "," + y + ")";
   }
}
```

- On peut alors écrire dans une méthode `main` d'une autre classe:

```
Point2D point = new Point2D();
System.out.println(point.afficher()); //(0,0) s'affiche
```

Collaboration entre constructeurs (1/2)

- Un constructeur peut appeler un autre constructeur de la même classe. `this()` **est utilisé pour faire appel à un autre constructeur de classe actuelle et doit être la première instruction.**
- Exemple: Ces deux codes sont équivalents.

```
public class Rectangle {
   private int longueur;
   private int largeur;

public Rectangle(int longueur, int largeur) {
     this.longueur = longueur;
     this.largeur = largeur;
   }
//Construire un rectangle de type carré
   public Rectangle(int cote) {
      longueur = cote;
      largeur = cote;
   }
// constructeur sans paramètres, ou par défaut
   public Rectangle() {
      longueur = 1;
      largeur = 1;
   }
   //...
}
```

Avec `this()`

```
public class Rectangle {
   private int longueur;
   private int largeur;

   public Rectangle(int longueur, int largeur) {
      this.longueur = longueur;
      this.largeur = largeur;
   }

   public Rectangle(int cote) {
      this(cote, cote);
      //Appel du constructeur avec 2 paramètres
   }

   public Rectangle() {
      this(1,1);
   //Appel du constructeur avec 2 paramètres

   }
   //...
}
```

Collaboration entre constructeurs (2/2)

- Supposons que l'on rajoute un attribut de type `boolean` qui vaut `true` si le rectangle est plein et `false` s'il est vide.

```
public class Rectangle {
   private int longueur;
   private int largeur;
   private boolean plein; // false par défaut

   public Rectangle(int longueur, int largeur) {
      this.longueur = longueur;
      this.largeur = largeur;
   }

   public Rectangle(int longueur, int largeur,
       boolean plein) {

      this(longueur, largeur);
      this.plein = plein;
   }

   //…

}
```

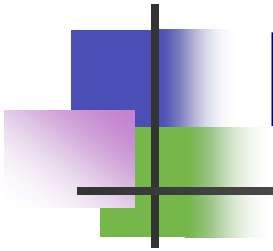
```
public class GestionRectangle {
public static void main(String[] args) {

/* Construire un rectangle de longueur 8 et de
largeur 4. la variable plein prend la valeur par
défaut : false */
Rectangle rect0 = new Rectangle(8, 4);

/* Construire un rectangle plein de longueur 10
et de  largeur 7  */
Rectangle rect1 = new Rectangle(10, 7, true);

}
}
```

L'appel du constructeur, `this(…)`, doit être la première instruction



Exercice

Téléchargez le fichier Cours.java sur Moodle puis complétez la classe `Cours` comme demandé ici-bas.

1. Ajoutez un constructeur qui permet de construire un cours à partir du code et du titre (constructeur à 2 paramètres).
2. Ajoutez un constructeur qui permet de construire un cours à partir du code, du titre et du nombre d'heures (constructeur à 3 paramètres). Utilisez le constructeur de la question 1 (constructeur à 2 paramètres).
3. Modifiez le constructeur initial à 4 paramètres pour utiliser le constructeur de la question2 (constructeur à 3 paramètres).
4. Ajoutez un constructeur par défaut (sans paramètre) qui initialise les attributs comme suit:
   - Code «Inconnu»
   - Titre: chaine vide
   - Nombre d'heures: 0
   - Session: -1

- Testez chaque constructeur.

Exercice: La classe Cours

```
public class Cours {
	private String code;
	private String titre;
	private int session;
	private int nbHeures;

	// mutateurs
	public void setCode(String code) {
		if (code.length() == 10) {
			this.code = code.toUpperCase();
		} else {
			this.code = "INCONNU";
		}
	}
	public void setTitre(String titre) {
		this.titre = titre;
	}
	public void setNbHeures(int nbHeures) {
		this.nbHeures = nbHeures;
	}
	public void setSession(int session) {
		if (session <= 6 && session >= 1) {
			this.session = session;
		}

	}
	// Accesseurs
	public String getCode() {
		return code;
	}
	public int getSession() {
		return session;
	}
	public void afficher() {
		System.out.println("Informations du cours: " + code);
		System.out.println("*******************************");
		System.out.println("Titre: " + titre);
		System.out.println("Session: " + session);
		System.out.println("Nombre d'heures par semaine: " + nbHeures);
	}
	public int heuresSession() {
		return nbHeures * 15;
	}
}
```

Exercice: corrigé

```
 // quesion 1
// constructeur à 2 paramètres
public Cours(String code, String titre) {
	setCode(code);
	this.titre = titre;
	// session et nbHeures valent 0

}
// question 2
// constructeur à 3 paramètres
public Cours(String code, String titre, int nbHeures) {
	this(code, titre);
	this.nbHeures = nbHeures;
	// session vaut 0

}

// question 3
public Cours(String code, String titre, int session, int nbHeures) {
	this(code, titre, nbHeures);
	setSession(session);

}

// question 4
public Cours() {
	this("Inconnu", "", 0, -1);

}
```

Exercice: corrigé

```
public static void main(String[] args) {
	// 1- Test du constructeur à 2 paramètres
	Cours c1 = new Cours("420-ZF5-MO", "Programmation structurée");
	c1.afficher();

	// 2- constructeur à 3 paramètres
	Cours c2 = new Cours("420-ZC6-MO", "Algorithmie et programmation", 6);
	c2.afficher();

	// 3- constructeur à 4 paramètres
	Cours c3 = new Cours("420-ZH5-MO", "Bases de données", 3, 5);
	c3.afficher();

	// 4- constructeur par défaut
	Cours c4 = new Cours();
	c4.afficher();

}
```

Sources

- Apprendre Java et la programmation orientée objet:
  https://www.ukonline.be/cours/java/apprendre-java
- Tasso, A. (s.d.). Le livre de JAVA premier langage avec 109 exercices corrigés. Éditions EYROLLES.
- Les images:
  https://webcourses.ucf.edu/courses/1249560/pages/what-is-object-oriented-programming