

# TD/TP A43 : Conception et programmation orientées objet (avec *corrections*)

|   |    |
|---|----|
| TD/TP A43 : Conception et programmation orientées objet (avec corrections)..... | 1  |
| Sequence 1 : Utilisation d'objets .....   | 3  |
| Exercice 1.....   | 3  |
| Exercice 2.....   | 3  |
| Exercice 3.....   | 3  |
| Exercice 4.....   | 4  |
| Séquence 2 : Expression en Java.....  | 5  |
| Exercice 1.....   | 5  |
| Exercice 2.....   | 5  |
| Exercice 3.....   | 5  |
| Séquence 3 : Attributs et méthodes d'un objet.....                              | 7  |
| Exercice 1.....   | 7  |
| Exercice 2.....   | 7  |
| Exercice 3.....   | 7  |
| Exercice 4.....   | 8  |
| Séquence 4 : Héritage.....  | 12 |
| Exercice 1.....   | 12 |
| Exercice 2.....   | 13 |
| Exercice 3.....   | 14 |
| Séquence 5 : Interactions entre objets.....                                     | 16 |
| Exercice 1.....   | 16 |
| Exercice 2.....   | 16 |
| Exercice 3.....   | 17 |
| Exercice 4.....   | 17 |
| Exercice 5.....   | 18 |
| Sequence 6 : Relations entre objets.....  | 20 |
| Exercice 1.....   | 20 |
| Exercice 2.....   | 21 |
| Exercice 3.....   | 21 |
| Sequence 7 : Architecture applicative.....                                      | 22 |
| Exercice 1.....   | 22 |
| Exercice 2.....   | 23 |
| Sequence 8 : Flux d'exécution - Thread.....                                     | 25 |
| Exercice 1.....   | 25 |
| Séquence 9 : Généricité et flux.....  | 26 |
| Exercice 1.....   | 26 |
| Exercice 2.....   | 26 |
| Exercice 3.....   | 26 |
| Exercice 4.....   | 27 |
| Exercice 5.....   | 28 |
| Séquence 10 : Polymorphisme et architecture ouverte.....                        | 30 |
| Exercice 1.....   | 31 |
| Exercice 2.....   | 32 |
| Exercice 3.....   | 32 |
| Exercice 4.....   | 34 |
| Exercice 5.....   | 34 |
| Exercice 6.....   | 34 |
| Séquence 11 : Design-pattern.....   | 36 |
| Exercice 1.....   | 36 |
| Exercice 2.....   | 36 |
| Exercice 3.....   | 37 |
| Sequence 12 : Réingénierie.....   | 39 |
| Exercice 1.....   | 39 |
| Exercice 2.....   | 39 |
| Exercice 3.....   | 40 |

|   |    |
|---|----|
| Exercice 4.....                         | 41 |
| Exercice 5.....                         | 41 |
| TOP.....                                | 44 |
| Exercice 1 : Dérivation formelle.....   | 44 |
| Exercice 2 : Opérateurs vectoriels..... | 44 |
| Annexes.....                            | 48 |
| Aide en ligne sur Random.....           | 48 |
| Aide en ligne sur Timer.....            | 49 |
| Aide en ligne sur TimerTask.....        | 50 |
| Aide en ligne sur Scanner.....          | 50 |
| Aide en ligne sur System.....           | 54 |

## Sequence 1 : Utilisation d'objets

Objectifs : Première appropriation du modèle objet. Distinguer action et propriété.

### Exercice 1

Modéliser sur un mode objet (actions (verbe), propriétés (*getXXX*)) quelques objets techniques de la vie quotidienne, par exemple :

- une calculette (chaque bouton sera représenté par une action distincte)
- un chronomètre (chaque bouton sera représenté par une action distincte)
- un fichier
- une fenêtre d'environnement graphique, *Windows*, *X-Window*... (chaque bouton sera représenté par une action distincte)

#### Correction

- calculette: voir exercice suivant
- chronometre : voir exercice suivant et séquence 3
- fichier : ouvrir, lire, écrire, fermer. Voir chapitre sur les flux dans le cours
- fenêtre: iconiser, désiconiser, ouvrir plein écran, déplacer, retailler, passer en avant plan, passer en arrière plan, fermer, etc...

### Exercice 2

On considère la classe définie en colonne de gauche. Traduire en une suite d'instructions d'un pseudo langage objet la série des actions de la colonne de droite:

| Voiture  |
|--|
| <code>+demarrer(): void</code>                     |
| <code>+mettreCarburant(volume:double): void</code> |
| <code>+choisirRapport(rapport:int=0): void</code>  |
| <code>+mettreClignotant(sens:boolean): void</code> |
| <code>+freiner(): void</code>                      |
| <code>+getVolumeCarburant(): double</code>         |

- Je démarre le moteur de la voiture et passe la première.
- Je passe la seconde et met le clignotant à gauche.
- Je consulte la jauge à essence.
- Je freine, met le clignotant à droite, et coupe le moteur.
- Dans une station service je met 52.5 l de carburant

#### Correction

```
voiture.demarrer()
voiture.choisirRapport(1)
voiture.choisirRapport(2)
voiture.mettreClignotant(false)
print voiture.getVolume()
voiture.freiner()
voiture.mettreClignotant(true)
voiture.mettreCarburant(52,5)
```

### Exercice 3

Même question

| Calculette                       |
|----------------------------------|
| <code>+zero(): void</code>       |
| <code>+un(): void</code>         |
| <code>+deux(): void</code>       |
| <code>+trois(): void</code>      |
| <code>+quatre(): void</code>     |
| <code>+cinq(): void</code>       |
| <code>+six(): void</code>        |
| <code>+sept(): void</code>       |
| <code>+huit(): void</code>       |
| <code>+neuf(): void</code>       |
| <code>+mul(): void</code>        |
| <code>+moins(): void</code>      |
| <code>+plus(): void</code>       |
| <code>+div(): void</code>        |
| <code>+raz(): void</code>        |
| <code>+getValue(): double</code> |
| <code>+dot(): void</code>        |
| <code>+egal(): void</code>       |

- Je réalise avec la calculette ci-dessus l'opération suivante :  $564 / 2 = \dots$
- Je consulte le résultat
- Je remet la calculette à zéro.

#### Correction

```
calculette.cinq()
calculette.six()
calculette.quatre()
calculette.div()
calculette.deux()
calculette.egal()
print calculette.getValue()
calculette.raz()
```

## Exercice 4

Même question

| <b>Chronometre</b>   |
|--|
| <pre>+start(): void +stop(): void +setLapTime(): void +getTime(): int +getLapTime(index:int): int +raz(): void</pre> |

### **Correction**

```
chrono.start()
chrono.setLapTime()
chrono.setLapTime()
chrono.setLapTime()
print chrono.getTime()
print chrono.getLapTime(0)
print chrono.getLapTime(1)
print chrono.getLapTime(2)
```

- *Je démarre le chronometre.*
- *J'enregistre le temps intermédiaire au premier tour.*
- *J'enregistre le temps intermédiaire au second tour.*
- *J'enregistre le temps intermédiaire au troisième tour.*
- *Je stoppe le chronomètre.*
- *Je consulte le temps final.*
- *Je consulte le premier temps intermédiaire.*
- *Je consulte le second temps intermédiaire.*
- *Je consulte le troisième temps intermédiaire.*

## Séquence 2 : Expression en Java

Objectif : commencer à faire la liaison entre le concept général d'objet et le langage Java.

### Exercice 1

On considère la classe Compteur où sont explicités le constructeur (*Compteur*) et l'attribut interne de l'objet (*value*). Traduire les actions ci-dessous en instructions appliquées à ce compteur. On utilisera si possible Java à la place du pseudo langage utilisé en séquence 1. Le signe – placé devant *value* signifie que cet attribut est privé et ne peut en conséquence pas être accédé de l'extérieur de l'objet. Les signes + signifient que les entités désignées sont utilisables sans restriction.

| Compteur         |
|------------------|
| -value: int      |
| +raz(): void     |
| +up(): void      |
| +getValue(): int |
| +Compteur()      |

- On crée un nouveau compteur;
- On effectue une remise à zéro sur ce compteur.
- On incrémente trois fois ce compteur
- On consulte sa valeur.
- On incrémente encore une fois
- On consulte la valeur.
- On effectue une remise à zéro sur ce compteur.
- On consulte la valeur.

### Correction

```
Compteur compteur=new Compteur();
compteur.raz();
compteur.up();
compteur.up();
compteur.up();
print compteur.getValue();
compteur.raz();
print compteur.getValue();
```

### Exercice 2

En vous aidant du cours implémenter en Java un tel *Compteur*. Le tester en appliquant la série d'instructions évoquées dans la question précédente.

Remarque : la phrase « on consulte sa valeur » peut par exemple être implémentée en provoquant un affichage de la valeur en question à l'écran (par *System.out.println(...)* en java)

### Correction

Fichier *Compteur.java*

```
public class Compteur {
    private int value;
    public Compteur() { raz(); }
    public void up() { value++; }
    public void raz() { value = 0; }
    public int getValue() { return value; }
}
```

Fichier *Launcher.java*

```
public class Launcher {
    public static void main(String[] args) {
        Compteur compteur=new Compteur();
        compteur.raz();
        compteur.up();
        compteur.up();
        compteur.up();
        System.out.println(compteur.getValue());
        compteur.raz();
        System.out.println(compteur.getValue());
    }
}
```

### Exercice 3

Reformuler la classe *Compteur* en explicitant le plus possible l'emploi du mot réservé *this*.

## Correction

*Fichier Compteur.java*

```
public class Compteur {  
    private int value;  
    public Compteur() { this.raz(); }  
    public void up() { this.value++; }  
    public void raz() { this.value = 0; }  
    public int getValue() { return this.value; }  
}
```