TD R1.01.P2 – Semaine 50

Objectifs du TD

- Comprendre le type *Duree* en le testant
- Utiliser simultanément *ArrayList* et *Duree*

Ce TD dure 1 X 1h30

Exercice 1. Test de la classe *Duree*

Etant donné la *javaDoc* de la classe *Duree* en annexe, écrire pour chacune des méthodes de cette classe *Duree* la méthode de test associée dans les 3 cas habituels : cas normaux, cas limites et cas d'erreurs.

Exemple : la méthode *void ajouter (Duree autreDuree)* sera testée par les méthodes *void testAjouterDuree()* et *void testCasAjouterDuree (...)*.

Le principe de base est qu'une *Duree* doit toujours être positive (ou nulle). Une méthode de la classe *Duree* se mettra en erreur si la *Duree* courante devient strictement négative ou si le paramètre passé à cette méthode est une *Duree* ou un temps strictement négatif.

On n'oubliera pas qu'un objet *Duree* ne peut exister que s'il a été explicitement créé avec un des 2 constructeurs de la classe *Duree* : *Duree (int millisec)* ou *Duree (int heures, int minutes, int secondes)*.

Les tests se feront dans l'ordre suivant :

- `void testConstructeursEtGetLeTemps()`
- `void testAjouter()`
- `void testSoustraire()`
- `void testCompareA()`
- `void testEnTexte()`

Exercice 2. Les types *ArrayList* et *Duree*

En page 3, on trouve un extrait de la *javaDoc* du type *ArrayList* de l'API Java.

On se rappelle que pour créer une *ArrayList* (collection), il faut :

- appeler le constructeur de la classe pour instancier un objet,
- préciser le type des données qui seront mémorisées dans la collection

Exemple de création d'une collection d'entiers (taille égale à zéro après la création) :

```
ArrayList<Integer> tab = new ArrayList<Integer>() ;
```

Dans cet exercice on va manipuler une collection (*ArrayList*) de *Duree*.

1. Ecrire une méthode *void remplirListe (ArrayList<Duree> theL)* dont le rôle est de remplir *theL* (supposée déjà créée) avec 5 objets *Duree* (1h33min10sec, 0h47min43sec, 3h28min16sec, 0h0min59sec, 2h0min14sec).

2. Ecrire la méthode *String toString (ArrayList<Duree> theL)* qui renvoie toutes les *Duree* contenues dans la liste *theL* (passée en paramètre) sous forme d'une chaîne de caractères au format :
   - duree 1 : HHH :MM :SS
   - duree 2 : HHH :MM :SS
   - …
   - duree n : HHH :MM :SS

3. Ecrire la méthode *Duree somme (ArrayList<Duree> theL)* qui renvoie la somme sous forme d'une *Duree* de toutes les *Duree* contenues dans la liste *theL* passée en paramètre.

4. Ecrire la méthode *Duree plusPetite (ArrayList<Duree> theL)* qui renvoie la plus petite *Duree* de la liste.

5. Ecrire la méthode *int nbreInterval (ArrayList<Duree> theL, Duree t1, Duree t2)* qui renvoie le nombre de *Duree* de la liste *theL* comprises entre *t1* et *t2* (*t1* <= *Duree* <= *t2*).

6. Ecrire la méthode *Duree plusGrdEcart (ArrayList<Duree> theL)* qui renvoie la *Duree* correspondant au + grand écart de temps entre 2 *Duree* contenues dans la liste.

javaDoc du type *ArrayList* de l'API Java

boolean	**add**(**E** e) Appends the specified element to the end of this list.
void	**add**(int index, **E** element) Inserts the specified element at the specified position in this list.
void	**clear**() Removes all of the elements from this list.
boolean	**contains**(**Object** o) Returns `true` if this list contains the specified element.
E	**get**(int index) Returns the element at the specified position in this list.
int	**indexOf**(**Object** o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	**isEmpty**() Returns `true` if this list contains no elements.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(**Object** o) Removes the first occurrence of the specified element from this list, if it is present.
int	**size**() Returns the number of elements in this list.
String	**toString**() Returns a string representation of this collection.

ANNEXE : JavaDoc de la classe Duree

Class Duree

java.lang.Object
    Duree

---

public class **Duree**
extends java.lang.Object

Cette classe définit une durée temporelle. Elle permet la manipulation d'intervalles de temps. Une durée s'exprime en millisecondes.

Author:

Kamp J-F.

Constructor Summary

Constructors

Constructor and Description
Duree(int millisec) Constructeur avec initialisation en millisecondes.
Duree(int heures, int minutes, int secondes) Constructeur à partir des données heures, minutes, secondes.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
void	**ajouter**(**Duree** autreDuree) Modificateur qui ajoute une durée à la durée courante.
int	**compareA**(**Duree** autreDuree) Accesseur qui effectue une comparaison entre la durée courante et une autre durée.
java.lang.String	**enTexte**(char mode) Accesseur qui renvoie sous la forme d'une chaîne de caractères la durée courante.
int	**getLeTemps**() Accesseur qui retourne la valeur de la durée courante en millisecondes.
void	**soustraire**(**Duree** autreDuree) Modificateur qui soustrait une durée à la durée courante.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Duree

public Duree(int millisec)

Constructeur avec initialisation en millisecondes.

Parameters:

millisec - la durée exprimée en millisecondes

Duree

public Duree(int heures,
             int minutes,

int secondes)

Constructeur à partir des données heures, minutes, secondes.

Parameters:

heures - nbre d'heures

minutes - nbre de minutes

secondes - nbre de secondes

Method Detail

getLeTemps

```
public int getLeTemps()
```

Accesseur qui retourne la valeur de la durée courante en millisecondes.

Returns:

la durée en millisecondes

compareA

```
public int compareA(Duree autreDuree)
```

Accesseur qui effectue une comparaison entre la durée courante et une autre durée.

Parameters:

autreDuree - durée à comparer à la durée courante

Returns:

un entier qui prend les valeurs suivantes :

- -1 : si la durée courante est + petite que autreDuree
- 0 : si la durée courante est égale à autreDuree
- 1 : si la durée courante est + grande que autreDuree
- -2 : si autreDuree est null (cas d'erreur)

enTexte

```
public java.lang.String enTexte(char mode)
```

Accesseur qui renvoie sous la forme d'une chaîne de caractères la durée courante.

Parameters:

mode - décide de la forme donnée à la chaîne de caractères
La forme de la chaîne de caractères dépend du "mode" (caractère passé en paramètre) choisi :

- si mode == 'J' => chaîne de caractères de la forme "JJJ jours HH h"
- si mode == 'H' => chaîne de caractères de la forme "HHH:MM:SS"
- si mode == 'S' => chaîne de caractères de la forme "SSS.MMM sec"
- si mode == 'M' => chaîne de caractères de la forme "MMMMMM millisec"

Returns:

la durée sous la forme d'une chaîne de caractères

ajouter

```
public void ajouter(Duree autreDuree)
```

Modificateur qui ajoute une durée à la durée courante.

Parameters:

autreDuree - durée à rajouter

soustraire

```
public void soustraire(Duree autreDuree)
```

Modificateur qui soustrait une durée à la durée courante.

Parameters:

autreDuree - durée à soustraire