Laboratoire 3 – Programmation orientée objet
ITI 1521. Introduction à l'informatique II
8-12 Février 2021
Dû en ligne après une semaine de votre Lab
/10

Objectifs

- Manipuler des tableaux et des références
- Utiliser des tests `JUnit` prédéfinis
- Implémenter des classes
- Approfondir la notion d'encapsulation de données

I. Première partie - Validation des programmes à l'aide de `JUnit`

Question 1 : (3 POINTS)

Implémenter une méthode nommée *replace* dans une classe nommée `Use` *:*
static String[] ***replace****(String[] tabIn, String[] tab, String[] tabOut) ;*

Cette méthode retourne une copie du tableau désigné par la référence *tabIn* où tous les mots se trouvant aussi dans le tableau désigné par *tab* ont été remplacés par le mot se trouvant à la même position dans le tableau *tabOut*. Le tableau désigné par *tabIn* demeure le même par un appel à la méthode. Par exemple, si on appelle la méthode *replace* avec les suivants :
String [] tabIn = new String[]{"Book","off","I"};
String [] tab = new String[]{"Java","C++","off"};
String [] tabOut = new String[]{"Id","Name","my"};
ell doit renvoyer le tableau suivant :
{"Book","my","I"};

Il s'agit d'un formalisme pour le traitement d'erreurs. On traitera les exceptions plus tard. La méthode *replace* retourne la valeur null lorsque l'une des conditions suivantes n'est pas satisfaites :

- La valeur des paramètres formels ne peut être null.
- La valeur des éléments des tableaux ne peut être null.
- Les tableaux désignés par *tab* et *tabOut* doivent être de même longueur.

Utiliser des tests `JUnit` prédéfinis. **`JUnit`** (http://junit.org) est un outil Java qui facilite ce travail.
Votre Assistant d'Enseignement présentera une brève introduction à `JUnit`. Il mènera une discussion au tour de chacun des cas du programme `TestReplace.java` : ce que chaque test valide, ses forces et faiblesses, quels sont les cas manquants (Voir ci-contre).

```
/***************** TestReplace.java *************/
import org.junit.*;
public class TestReplace.java {

@Test
 public void testInIsNull() {
String[] same = { "I" };
String[] displace = { "You" };
Assert.assertNull (Use.replace( null, same , displace));
   }

@Test
 public void testSameIsNull() {
String[] first = { "I", "know" };
String[] displace = { "You" };
Assert.assertNull (Use.replace( first, null, displace ));
   }

@Test
 public void testDisplaceIsNull() {
String[] first = { "I", "know" };
String[] same = { "I" };
Assert.assertNull (Use.replace( first, same , null ) );
   }

@Test
 public void testInAndSameAreNull() {
String[] displace = { "You" };
Assert.assertNull (Use.replace( null, null, displace ));
   }

@Test
  public void testInAndDisplaceAreNull() {
String[] same= { "I" };
Assert.assertNull (Use.replace( null, same, null ) );
   }

@Test
  public void testSameAndDisplaceAreNull() {
String[] text = { "I", "know" };
Assert.assertNull (Use.replace( text, null, null ) );
   }

@Test
public void testAllNull() {
Assert.assertNull (Use.replace( null, null, null ) );
   }

@Test
 public void testNotSameLength() {
String[] first = { "I", "know" };
String[] same= { "I" };
String[] displace = { "You", "They" };
Assert.assertNull (Use.replace( first, same, displace));
   }
```

```
@Test
 public void testNullInIn() {
String[] first = { "I", null };
String[] same= { "I" };
String[] displace = { "You" };
Assert.assertNull (Use.replace( first, same, displace ) );
   }

@Test
 public void testNullInSame() {
String[] first = { "I", "know" };
String[] same= { "I", null };
String[] displace  = { "You", "They" };
Assert.assertNull (Use.replace( first, same, displace ) );
   }

@Test
 public void testNullInDisplace() {
String[] first = { "I", "know" };
String[] same= { "I", "We" };
String[] displace  = { null, "They" };
Assert.assertNull (Use.replace( first, same, displace ) );
   }

@Test
public void testNoChange() {
String[] first = { "I", "know" };
String[] same= { "You" };
String[] displace = { "I" };
String[] result = Use.replace( first, same, displace );
Assert.assertNotNull (result );
Assert.assertFalse (first == result );
Assert.assertTrue (first.length == result.length );
for ( int i=0; i<first.length; i++ ) {
   Assert.assertEquals (first[ i ], result[ i ] );
 }
}

@Test
 public void testChange() {
String[] first = { "I", "know" };
String[] same = { new String( "know" ) };
String[] displace = { "see" };
String[] expected = { "I", "see" };
String[] result = Use.replace( first, same, displace );
Assert.assertNotNull( result );
Assert.assertFalse (first == result );
Assert.assertTrue (first.length == result.length );
for ( int i=0; i<first.length; i++ ) {
   Assert.assertEquals (expected[ i ], result[ i ] );
 }
}
}
/*****************************************************************************/
```

II. Deuxième partie - Programmation orientée objet

Reprendre les classes `Book` et `TestBook` du LAB2 précédent (Partie I).

Question 2 : (3 POINTS)

Ajoutez à votre classe `Book` :

- une variable `price` de type `double` (un prix aux livres)
- 2 méthodes `getPrice` et `setPrice` pour obtenir le prix et le modifier.
- au moins un constructeur qui prend le prix en paramètre (`price`).

Testez.

Si le prix d'un livre n'a pas été fourni, la description du livre (`toString()`) devra indiquer "Prix pas encore fourni ". La valeur -1 indiquera que le prix n'a pas encore été fourni.

On bloque complètement les prix : un prix ne peut être donné qu'une seule fois et ne peut être modifié ensuite. Une tentative pour changer le prix doit afficher un message d'erreur.

Récrire la méthode *`setPrice`*.

Vous ajouterez une variable booléenne *fixedPrice* (pour "prix fixe") qui indiquera que le prix ne peut plus être modifié.

Ajoutez une méthode "`isFixedPrice`" qui renvoie vrai si le prix a déjà été fixé.

Exemple de sortie :

Error : negative price !
Error : negative price !
Book[title=Abstraction and Design Using Java, author=E.B.Koffman , fixedPrice = true, price = $100.0]
Book[title= Data Structures in Java for Principled Programmer , author=Duane A.Bailey, fixedPrice = true, price = $120.0]

Question 3 : (2 POINTS)

Dans cette question, pour simplifier, vous vous placerez dans le cas où les prix des livres sont **bloqués :** le prix d'un livre ne peut être donné qu'une seule fois et ne peut être modifié ensuite.

Créez une classe `Accountant` (à part et pas dans le fichier `Book.java`) qui possède une variable *totalPrice*, deux méthodes `getTotalPrice` et `count` :

```
void count(Book b) ;
```

Une instance de cette classe permettra de calculer le prix total de tous les livres qu'on lui aura passé par la méthode `count`.

Testez votre nouvelle classe dans la méthode `main` de `TestBook`. Deux objets de type `Accountant` seront créés. Ils comptabiliseront chacun quelques livres.

Afficher le total des prix ainsi comptabilisés par chacun des 2 objets de type `Accountant`.

Exemple de sortie :

Price is fixed !
Book[title=Abstraction and Design Using Java, author=E.B.Koffman , fixedPrice = true, price = $0.0]
Book[title= Data Structures in Java for Principled Programmer , author=Duane A.Bailey, fixedPrice = true, price = $120.0]
Book[title=Mark Grand, author=Pattern in Java, fixedPrice = true, price = $250.0]
total book prices recorded by the 1st accountant is : $ 120.0
total book prices recorded by the second accountant is : $ 250.0

Question 4 : (2 POINTS)

Dans cette question, on veut cacher la classe `Accountant` aux "clients" de la classe `Book`, sans modifier la classe `Accountant`. Les livres vont faire enregistrer automatiquement leur prix par un seul comptable (`Accountant`) : appelez directement la méthode `count` depuis la méthode `setPrice` (et depuis les constructeurs si nécessaire). Utiliser un `this` explicite. A tout moment on peut demander le total des prix comptabilisés. Testez.

Exemple de sortie :

Book[title=Abstraction and Design Using Java, author=E.B.Koffman , fixedPrice = true, price = $100.0]
Book[title= Data Structures in Java for Principled Programmer , author=Duane A.Bailey, fixedPrice = true, price = $120.0]
Book[title=Mark Grand, author=Pattern in Java, fixedPrice = true, price = $250.0]
total book prices recorded by the accountant is : $ 470.0

Créer et soumettre un fichier zip comme d'habitude (Q1, …Q4)