INFO-H-200 : Programmation orientée objet
TP 3 - Exceptions et tests unitaires - Exercices

Professeur : Hugues Bersini
http://cs.ulb.ac.be/public/teaching/infoh200

Année académique 2015-2016

Exercice 3.1

Ajouter à la classe `Date` les méthodes statiques `isLeapYear(int year)` et `daysInMonth(int month, int year)`. Utiliser ces méthodes dans les méthodes non statiques correspondantes.

Exercice 3.2

Générer des `RuntimeException` lors d'erreurs dans les méthodes de la classe `Date`.
Tip (package nécessaire aux exceptions) :

```
import java.lang.RuntimeException;
```

Exercice 3.3

Ajouter dans votre package la classe `DateTest` contenant des tests *JUnit*. Exécuter ces tests et modifier votre classe `Date` jusqu'à ce que tous les tests passent. Ensuite, implémenter et exécuter les tests proposés dans les commentaires à la fin de la classe `DateTest`.
Tip (package nécessaire aux tests) :

```
import static org.junit.Assert.*;
import org.junit.Test;
```

Exercice 3.4

Ajouter à la classe `Date` une méthode `public void writeToFile(String fileName)` qui écrit les données de l'objet dans un fichier texte dont le nom est donné en paramètre. Créer ce fichier s'il n'existe pas encore. Gérer les exceptions.
Tip (package nécessaire aux exceptions sur fichier) :

```
import java.io.FileNotFoundException;
import java.io.IOException;
```

INFO-H-200 - Programmation orientée objet
TP 3

Corrections

Exercice 3.1

```
/**
 * La classe Date est la même que dans le corrigé précédent !
 * Seules les méthodes 'daysInMonth' et 'dayOfYear' changent !
 * La méthode 'isLeapYear' est créée.
 */

package Example;

public class Date {
  . . .
  . . .

  /**
   *  @returns true if it is a leap year.
   */
  private static Boolean isLeapYear(Integer year){
    return (year%400==0 || (year%100 !=0 && year%4==0));
  }

  /**
   *  @returns The number of days in the month this date belongs to.
   **/
  private Integer daysInMonth(Integer month, Integer year){
    Integer value = 0;
    if (isLeapYear(year) && month == 2){
      value = 0;
    }else{
      value = 1;
    }
    return daysInMonths[month-1] + value;
  }

  /**
   * @returns the day number in the current year this date represents.
   */
  public Integer dayOfYear(){
    Integer dayOfYear=this.day;
    for(Integer i=0; i<month-1;i++){
      dayOfYear += daysInMonth(month, year);
    }
    return dayOfYear;
  }
}
```

Exercice 3.2

```
/**
 * La classe Date est la même que dans le corrigé précédent !
 * Seules les méthodes 'setDay', 'setMonth' et 'setYear' sont rajoutées.
 * Le reste du code doit être modifié de manière à remplacer les assignations
 * sur les attributs de la classe par des appels au méthodes 'SET'.
 */

import java.lang.RuntimeException;

package Example;

public class Date {
  . . .
  . . .

  /**
   *  @returns true if it is a leap year.
   */
  private Boolean isLeapYear(Integer year){
    return year%400==0 || (year%100 !=0 && year%4==0);
  }

  /**
   * Sets the Year.
   */
  public void setYear(Integer year){
    if(isLeapYear(this.year) && !isLeapYear(year) && lastDayOfMonth()){
      throw new RuntimeException();
    }
    else{
      this.year=year;
    }
  }

  /**
   * Sets the month.
   */
  public void setMonth(Integer month){
    if(month>=1 && month <=12){
      this.month = month;
```

```
    }else{
      throw new RuntimeException("Bad month !");
    }
  }

  /**
   * Sets the day.
   * Year and Month must be set already BEFORE calling this method.
   */
  public void setDay(Integer day){
    if(day>=1 && day<=this.daysInMonth()){
      this.day=day;
    }else{
      throw new RuntimeException("Bad day !");
    }
  }
}
```

Exercice 3.3

```
import static org.junit.Assert.*;
import org.junit.Test;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class DateTest {

  @Test
  public void testIncrementSimple() {
    Date d = new Date(1,1,2012);
    d.increment();
    assertEquals(2,d.getDay());
    assertEquals(1,d.getMonth());
    assertEquals(2012,d.getYear());
  }

  @Test
  public void testIncrementEndMonth() {
    Date d = new Date(31,1,2012);
    d.increment();
    assertEquals(1, d.getDay());
    assertEquals(2, d.getMonth());
    assertEquals(2012, d.getYear());
  }

  @Test
  public void testIncrementEndYear() {
    Date d = new Date(31,12,2012);
    d.increment();
    assertEquals(1, d.getDay());
    assertEquals(1, d.getMonth());
    assertEquals(2013, d.getYear());
  }

  @Test(expected=RuntimeException.class)
  public void testConstructorInvalidMonth(){
    new Date(1,-1,2012);
  }

  @Test(expected=RuntimeException.class)
  public void testConstructorInvalidMonth1(){
    new Date(1,0,2012);
  }

  @Test(expected=RuntimeException.class)
  public void testConstructorInvalidMonth2(){
    new Date(1,13,2012);
  }

  @Test
  public void testConstructorValid(){
    Date d = new Date(16,2,2012);
    assertNotNull(d);
  }

  @Test
  public void testFebruaryLeapYear2012(){
    Date d = new Date(29,02,2012);
    assertNotNull(d);
    d.increment();
    assertEquals(1, d.getDay());
    assertEquals(3, d.getMonth());
  }

  @Test
  public void testFebruaryLeapYear2000(){
    Date d = new Date(29,02,2000);
    assertNotNull(d);
  }

  @Test(expected=RuntimeException.class)
  public void testFebruaryLeapYear1900(){
    new Date(29,02,1900);
  }

  @Test
```

```
  public void testFebruary2012Increment(){
    Date d = new Date(29,02,2012);
    d.increment();
    assertEquals(1, d.getDay());
    assertEquals(3, d.getMonth());
  }

  @Test
  public void testFebruary1900Increment(){
    Date d = new Date(28,02,1900);
    d.increment();
    assertEquals(1, d.getDay());
    assertEquals(3, d.getMonth());
  }

  @Test
  public void testToString(){
    Date d = new Date(1,03,2012);
    assertEquals("Jeudi 1 Mars 2012 le 61 ième jour de l'année",d.toString());
  }

  @Test
  public void testWriteToFile(){
    Date d = new Date(1,03,2012);
    d.writeToFile("test.txt");
    try {
      BufferedReader in = new BufferedReader(new FileReader("test.txt"));
      String test = in.readLine();
      assertEquals("Jeudi 1 Mars 2012 le 61 ième jour de l'année", test);
    } catch (FileNotFoundException e) {
      fail();
    } catch (IOException e) {
      fail();
    }
    //TODO : il faudrait supprimer le fichier test.txt
  }

  @Test(expected=RuntimeException.class)
  public void testSetYearLeapToNotLeap(){
    Date d = new Date(29,02,2012);
    d.setYear(2011);
  }


  /* TODO : ajouter des tests pour aller plus loin !:
   *   * Tester un changement de mois.
   *     Par exemple new Date(31,01,2012).setMonth(2) -> Exception
   *   * Tester dayOfYear() (faire un cas également les années bisextiles)
   *   * Tester dayOfWeek()
   */
}
```

Exercice 3.4

```
/**
 * La classe Date est la même que dans le corrigé précédent !
 * Seules les méthodes 'daysInMonth' et 'dayOfYear' changent !
 * La méthode 'isLeapYear' est créée.
 */

public class Date {
  . . .
  . . .

  public void writeToFile(String fileName){
    try {
      BufferedWriter out = new BufferedWriter(new FileWriter(fileName));
      out.write(this.toString());
      out.close();
    } catch (IOException e) {
      System.out.println("Problème lors de l'enregisrement de la date : \n"+e.getMessage());
    }
  }
}
```