Final & Static

Gegeven is een Date-klasse die datums (dag, maand, jaar) voorstelt. Maak een nieuw project in IntelliJ en check dat de tests slagen.

1 Omzetten objectmethodes naar klassemethodes

Er zijn drie objectmethodes in de klasse Date die beter als klassemethodes zouden geïmplementeerd zijn. Het gaat om

- isLeapYear
- numberOfDaysInMonth
- isValidDate

Als je kijkt naar de tests zal je opmerken dat om deze drie methodes te testen, er telkens eerst een Date-object diende aangemaakt te worden, wat vrij onzinnig is: waarom zouden we een Date-object nodig hebben om te weten of 2000 een schrikkeljaar is?
Maak van deze drie methodes klassemethodes.

2 Aanpassen tests

Je zal merken dat de tests blijven draaien desondanks het feit dat je van isLeapYear, numberOfDaysInMonth en isValidDate klassemethodes hebt gemaakt. Dit komt doordat Java toelaat klassemethodes op dezelfde manier op te roepen als objectmethodes, maar dit is ten zeerste afgeraden. Je herschrijft
dus best de oorspronkelijke code

```
Date date = new Date(1 , 1, 1);
assertTrue( date.isLeapYear( 2000 ) );
```

naar

```
assertTrue( Date. isLeapYear( 2000 ) );
```

Pas de tests aan zodat de juiste syntax wordt gebruikt. Verwijder tevens het veld date en de setUp methode uit DateTests. Check dat de tests nog steeds slagen.

3 De Datevelden final maken

We willen Date-objecten immutable maken. Dit betekent dat we Date-objecten onwijzigbaar willen maken en dat alle velden final worden. We werken stapsgewijs.

3.1 Aanpassen advanceSingleDay

Om Date-objecten immutable te maken, moeten we alle methodes aanpassen die de toestand van het object wijzigen. advanceSingleDay is zulk een methode: deze wijzigt het Date-object zodat het verwijst naar de volgende dag.

Bijvoorbeeld, de volgende code

```
Date date = new Date(1 , 1, 2000);
date.advanceSingleDay();
```

maakt een object aan dat de datum 1/1/2000 voorstelt en nadien vraagt aan

dit object om te verwijzen naar de volgende dag. Na uitvoer van deze code zal date dus de datum 2/1/2000 voorstellen.

We willen advanceSingleDay nu zo aanpassen dat het een *nieuw* Date-object aanmaakt i.p.v. het huidige wijzigt. Beschouw de code

```
Date date = new Date(1 , 1,  2000);
Date nextDay = date. advanceSingleDay();
```

Na uitvoering van deze code stelt datede datum 1/1/2000 voor, en nextDay 2/1/2000.

Pas advanceSingleDay alsook de tests aan: kijk na dat deze laatste nog steeds slagen.

3.2 **Aanpassen** goBackSingleDay

Doe hetzelfde voor goBangSingleDay.

3.3 **Verwijderen setters**

Verwijder alle setters uit de klasse Date. Je zal alle tests die deze setters checken eveneens moeten verwijderen.

3.4 Final **invoeren**

Nergens is er nog code die de velden wijzigt. Je kan de velden day, month en year nu final maken.

4 Subklassen verbieden

We willen verbieden dat er subklassen aangemaakt worden van onze klasse Date. Voeg hiervoor waar nodig final toe.

5 Static Factory Method

We zijn geen fan van de huidige constructor. Stel dat we volgende code zien:

```
Date date = new Date(8 , 5,  2014);
```

Wij schrijven standaard de dag gevolgd door de maand, maar we weten dat die in de Amerikaanse notatie eerst de maand en dan de dag wordt gegeven. Maakt bovenstaande code dan 8 mei aan, of 5 augustus? Er staat niks anders op dan het op te zoeken in de documentatie. . . We kunnen echter wel een zogenaamde "static factory method" definiëren. Dit is gewoon een moeilijke naam voor een klassemethode die een object van die klasse aanmaakt. In ons geval betekent dat dus een statische methode die een Date object aanmaakt.

Static factory methodes hebben een aantal voordelen t.o.v. constructoren. Eén ervan is dat we ze een descriptieve naam kunnen geven, daar waar constructoren verplicht zijn de naam van hun klasse te dragen. We maken gebruik van deze vrijheid om de volgende static factory methods aan te maken:

```
public class Date {
  public static Date createDMY( int day , int month , int year) {
    // . . .
  }

  public static Date createMDY( int month , int day , int year) {
    // . . .
  }
}
```

Beide deze methodes hoeven enkel een nieuw Date-object aan te maken via new.

Hierna kan je de constructor privatemaken om gebruikers van de Date-klasse te verplichten gebruik te maken van de duidelijke createDMY of createMDY methodes. Pas de tests aan opdat ze compileren en slagen.