HOGESCHOOL

Labo Java OO programming - Labo02

Academiejaar 2018-2019 - Semester2

1 Opgave - WPS - Toepassing op getters en setters

Routers zijn vaak voorzien van de functie `WPS` (Wi-Fi Protected Setup). Deze functie is bedoeld om de procedure van het verbinden met een beveiligd draadloos netwerk vanaf een computer of een ander apparaat te vergemakkelijken. Een beschrijving van de wifi code vind je terug in `https://nl.wikipedia.org/wiki/Wi-Fi`:

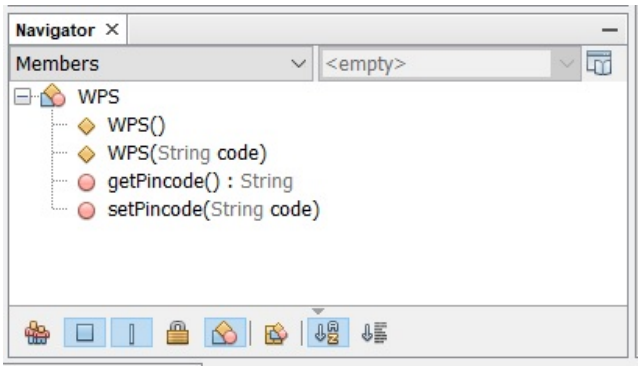
> De wifi pincode is geen willekeurig getal maar bestaat uit 8 cijfers. De eerste 7 mogen willekeurig gekozen worden (0..9), de laatste is een "checksum digit". Deze wordt als volgt berekend: tel de cijfers op de oneven posities bij elkaar op, vermenigvuldig het getal met 3, tel daarbij de cijfers op de even posities, bereken vervolgens de rest na deling door 10, en trek die rest af van het getal 10. Voorbeeld: PIN=66323211. De eerste 7 cijfers zijn 6632321. De som van de cijfers op oneven posities is 6+3+3+1=13. Vermenigvuldiging met 3 levert 39. De som van de cijfers op de even posities is 6+2+2=10. Opgeteld levert dit 39+10=49. Delen door tien levert rest 9. Rest 9 aftrekken van 10 levert 1; het laatste cijfer van de PIN is dus 1.

Noot: Een bijzonder geval is wanneer de bekomen som van de cijfers een tiental oplevert, dan is rest na deling door 10 gelijk aan 0. Die rest 0 aftrekken van 10 levert 10!!!! Dit is echter geen geldige digit om op te nemen als laatste digit van de pincode. Je neemt bijgevolg in dat geval als laatste digit niet 10 op, maar wel 0!

Maak een nieuw project aan in de `Netbeans` IDE voor deze opgave: `Labo02_WPS`. Voor deze opgave wordt jou gevraagd om logica te schrijven rond de wifi pincode. Breng je code onder in een package met naam `logica`. De goede werking van de geschreven functionaliteit zal je vervolgens kunnen verifiëren aan de hand van het `JUnit` testbestand dat je op `Toledo` bij de opgave terugvindt. Je hoeft voor deze opgave geen presentatieklasse te schrijven.

1.1 De logische klasse WPS

Hieronder zie je een screenshot van het Navigator venster van de `Netbeans` IDE, waarbij de private members van de klasse verborgen zijn:



Figuur 1: de klasse WPS

Het publieke deel van de klasse bestaat uit volgende onderdelen:

- 2 constructoren, eentje waarbij extern een pincode wordt bepaald en eentje waarmee intern een random pincode wordt gegenereerd.
- een getter. Aan de hand van deze methode kan de, evt. intern gegenereerde, pincode opgevraagd worden.
- een setter, waarmee het mogelijk is om een andere pincode in te stellen.

Bepaal zelf welke private datavelden en methoden je nodig hebt/zinvol zijn om de gevraagde functionaliteit te kunnen leveren.

1.2 Test je logische klasse

Verifieer de correcte werking van je labo-opgave aan de hand van de opgegeven `JUnit` testen.

2 Opgave - Tellen van letters in woorden

2.1 Basis scenario

Schrijf een console toepassing die telt hoeveel keer bepaalde letters voorkomen in opgegeven woorden. Het te implementeren scenario verloopt als volgt: Voor een ingelezen woord (bv. *paddenstoel*) zal je achtereenvolgens het aantal voorkomens van een ingelezen letter (bv. respectievelijk *p, d, ...*) tellen, en dat tot het karakter '.' wordt ingelezen als nieuwe letter. Vervolgens wordt een nieuw woord ingelezen, waarvoor je opnieuw ingelezen letters kan beginnen tellen. Wanneer je als nieuw woord de tekst `"."` ingeeft, stopt je programma. Een concreet voorbeeld:

```
1) Geef woord: paddenstoel
        Geef een letter: p
        De letter p komt 1 keer voor in woord paddenstoel
        Geef een letter: d
        De letter d komt 2 keer voor in woord paddenstoel
        Geef een letter: .

2) Geef woord: kabouter
        Geef een letter: i
        De letter i komt 0 keer voor in woord kabouter
        Geef een letter: .

3) Geef woord: .
```

2.2 Structuur van je toepassing

In de `Netbeans` IDE maak je een nieuw project aan voor deze opgave: `Labo02_WoordTeller`.

Definieer 2 packages in dit project:

- In de package `logica` implementeer je de logische klasse `Woord`.
- In de package `presentatie` breng je het eigenlijke scenario onder. Voorzie 2 implementaties van de applicatie:
  - een eerste implementatie realiseer je via de presentatieklasse `ConsoleStatisch`
  - een tweede implementatie realiseer je via de presentatieklasse `ConsoleDynamisch`

  Beide presentatieklassen roepen logica van de logische klasse `Woord` op om het opgegeven scenario te realiseren. De ene gebruikt alleen de statische methode van de logische klasse om de applicatie te bouwen, de andere gebruikt deze net niet.

2.3 Logica voor de statische oplossing

Schrijf hiervoor een statische methode `telLetter` in de logische klasse `Woord`:

```
public static int telLetter(String woord, char letter)
```

Gebruik deze methode vanuit je presentatieklasse `ConsoleStatisch` om het opgegeven scenario te realiseren.

2.4 Logica voor de object georiënteerde oplossing

Breid je logische klasse `Woord` uit met een niet-standaard constructor. Aan de hand van deze constructor zal je een woord (dataveld van het type `String`) initialiseren. Voorzie daarnaast ook een niet-statische methode `telLetter` (*i.e. method overloading t.o.v. je eerder geschreven statische methode*) die zal tellen hoeveel keer een opgegeven letter in dat geïnitialiseerde woord voorkomt.

```
public Woord(String woord)
public int telLetter(char letter)
```

Roep deze constructor en de tweede `telLetter` methode op in je presentatieklasse `ConsoleDynamisch` om het opgegeven scenario te realiseren.

2.5 Uitgebreid scenario

Bovenstaande OO-oplossing leert dat je een eenmalig aangemaakt `Woord`-object meerdere keren kan aanspreken om er telkens een andere letter uit te gaan tellen! Zo wordt het ook mogelijk om in de klasse `Woord` bij te houden welke letters uit het woord reeds geteld geweest zijn. Voorzie hiervoor een extra dataveld `letterStatus` (van het type `boolean[]`) in je logische klasse, alsook een extra methode `geefOverzichtStatus`, met volgende signatuur:

```
public String geefOverzichtStatus()
```

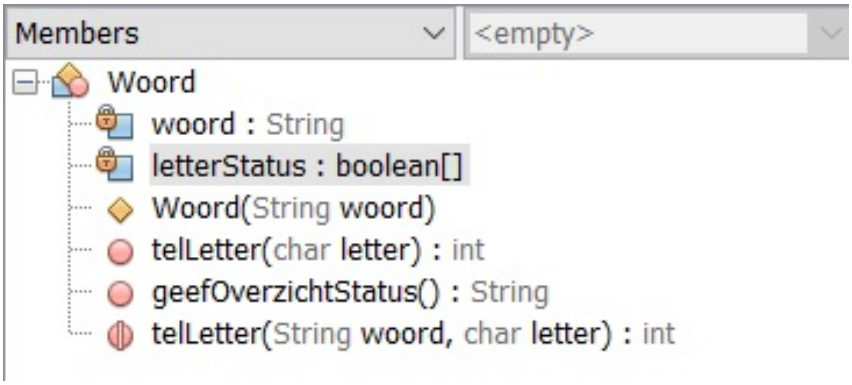
Laat je toepassing nu ook deze extra functionaliteit oproepen telkens nadat het tellen voor een bepaald woord beëindigd is:

```
1) Geef woord: paddenstoel
       Geef een letter: p
       De letter p komt 1 keer voor in woord paddenstoel
       Geef een letter: d
       De letter d komt 2 keer voor in woord paddenstoel
       Geef een letter: .
   -->> Nog niet geteld: *a**enstoel

2) Geef woord: kabouter
       Geef een letter: k
       De letter k komt 1 keer voor in woord kabouter
       Geef een letter: .
   -->> Nog niet geteld: *abouter

3) Geef woord: .
```

Het klassendiagramma van de klasse `Woord` vind je terug in het navigatorvenster van je `Netbeans` project (zie figuur 2)

2.6 Test je toepassing

Verifieer de correcte werking van je labo-opgave aan de hand van de opgegeven `JUnit` testen.



Figuur 2: Netbeans Navigatorvenster - klasse Woord