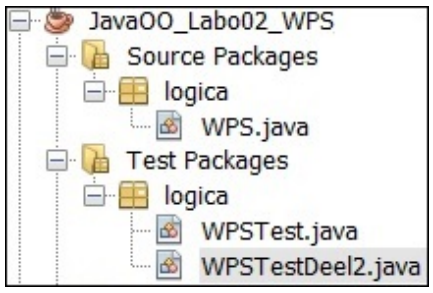
HOGESCHOOL

Labo Java OO programming - Labo03

Academiejaar 2018-2019 - Semester2

1 WPS - Deel2

Pas je klasse `WPS` uit het vorige labo aan zodat het niet meer mogelijk is om een `WPS`-object aan te maken met een ongeldige pincode. Zorg er ook voor dat het niet meer mogelijk is om de pincode van een reeds bestaand `WPS`-object te wijzigen naar een ongeldige pincode. Werp m.a.w. in beide gevallen een gepaste exceptie.



Figuur 1: WPS-klasse: labo2 > labo3

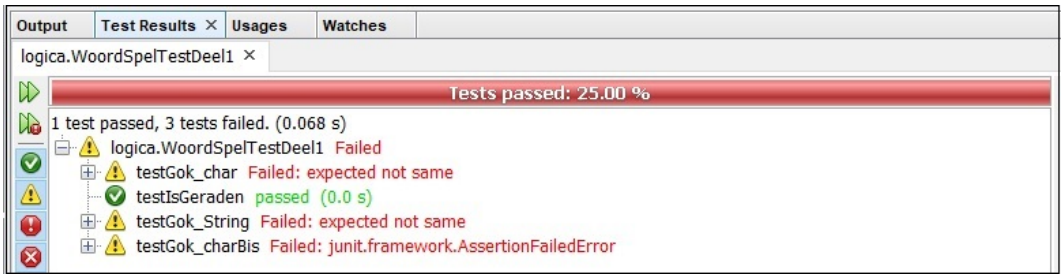
Verifieer de correcte implementatie van de gevraagde aanpassing aan de hand van de extra JUnit klasse `WPSTestDeel2` die je op Toledo terugvindt. Plaats deze klasse in de `test`-map van je Netbeans project.

2 WoordSpel

2.1 JUnit test voor labo11 - Java Fundamentals

Verifieer je oplossing van labo11 uit Java Fundamentals via de aangeleverde `JUnit` test `WoordSpelTestDeel1`.

Op Toledo vind je een Netbeans project terug: `JavaOO_WoordSpel (startversie).zip`. Daarin kan je, naast een aantal `JUnit` testen ook een startversie van de klasse `WoordSpel` zelf terugvinden. Deze startversie kan zinvol zijn voor de studenten die deze oefening nog niet eerder hebben gemaakt of die vorig semester problemen hebben ondervonden om de opgave correct op te



Figuur 2: Resultaat startversie

bouwen. Noteer dat de eerste `JUnit` test binnen het opgegeven Netbeans project alvast een slaagpercentage van `25%` oplevert (zie figuur 2).

In labo11 bij `Java Fundamentals` luidde de opgave als volgt:

`WoordSpel` - Schrijf een programma dat een gebruiker moet laten raden naar een woord. Het te raden woord wordt willekeurig gekozen uit een array die vooraf gevuld werd met mogelijke woorden. De gebruiker geeft ofwel een letter in ofwel het volledige woord wanneer hij denkt het woord te kennen. Het programmaverloop ziet er als volgt uit :

```
Geef je letter of het volledige woord : t
beurt 1: t__t__
Geef je letter of het volledige woord : n
beurt 2: t__t_n
Geef je letter of het volledige woord : s
beurt 3: t_st_n
Geef je letter of het volledige woord : testen
beurt 4: Proficiat !
Je vond het woord in 4 trials
```

Voorzie een constante om het maximaal aantal beurten vast te leggen. Zolang het woord niet geraden werd en het maximum aantal beurten niet overschreden werd mag de speler verder raden. Maak een klasse `WoordSpel` waarin je de volgende datavelden bijhoudt:

- een constante met het maximum aantal beurten
- een array opgevuld met 10 verschillende woorden
- het huidige woord dat geraden moet worden
- een booleaanse array die bijhoudt welke letter uit het huidige woord al geraden werd.
- het aantal gokbeurten

Voorzie tevens volgende methoden :

- een constructormethode om je datavelden te initialiseren.
- een `isGeraden` methode die test of alle booleans van de booleaanse array op true staan. Deze methode neemt geen parameters en geeft zelf ook een booleaanse waarde terug.
- een `gok` methode die test of de meegegeven letter in het woord voorkomt. Indien ja, dan moet de booleaanse array aangepast worden. De methode geeft een String terug die alle tot dan toe correct geraden letters weergeeft. De andere letters worden verstopt achter een underscore karakter.

- een `gok` methode die test of het meegegeven woord het te raden woord is. Deze versie van de gok methode is een overload versie van de vorige. Deze methode neemt nu een string als parameter ipv een karakter. Als output geeft deze methode opnieuw een String terug. Wanneer de gok correct was, wordt het te raden woord volledig weergegeven en de booleaanse array wordt aangepast. Indien de gok niet juist was, wordt opnieuw de string met alle tot dan toe geraden karakters weergegeven.

2.2 Uitbreiding - Java OO

Herschik de structuur van je toepassing in 2 packages: In de package `logica` behoud je de logische klasse `WoordSpel`. De klasse `WoordSpelConsole` verplaats je naar een package met naam `presentatie`.

Doe volgende aanpassingen aan je logische klasse:

- Maak alle velden privaat en maak de waarde van het veld `aantalBeurten` extern toegankelijk via de getter-methode `getAantalBeurten()`. Om je toepassing werkende te houden zal je in de presentatieklasse gebruik moeten maken van deze nieuwe getter-methode.
- Definieer een methode `nogBeurten()` waarmee je kan opvragen of er nog beurten over zijn om het woord te raden.
- Voorzie een niet-default constructor (constructor overloading!) waarmee je de constante met een gewenst maximum aantal toegestane beurten kan instellen.

Verifieer je oplossing aan de hand van de aangeleverde `JUnit` test `WoordSpelTestDeel2`.

2.3 Woordenlijst

Definieer een extra klasse `Woordenlijst` en breng er je array met vooraf gevulde woorden (eerder gedefinieerd binnen de klasse `WoordSpel`) in onder.

- Voorzie in deze klasse een statische methode `haalWoord`, zonder parameters, waarmee het mogelijk wordt om een random woord uit de beschikbare woordenlijst op te vragen.

  Zorg ervoor dat je logische klasse `WoordSpel` nu deze methode gebruikt om een random woord te kiezen uit de woordenlijst.

- Definieer een overloaded versie van de methode `haalWoord` waarmee het mogelijk wordt om een random woord op te vragen dat maximaal uit een opgegeven aantal letters bestaat. Indien geen woord gevonden kan worden dat aan het gestelde criterium voldoet, werp je een `IllegalArgumentException`.

  Voorzie in je logische klasse `WoordSpel` een extra constructor, zodat je een woordspel kan starten voor een gewenst maximaal aantal beurten én een gewenst maximaal aantal letters.

Verifieer je oplossing via de aangeleverde `JUnit` test `WoordSpelTestDeel3`.

Figuur 3: KlassenDiagramma

Noot: In bovenstaand klassendiagramma zijn enkele private hulpmethoden opgenomen:

- De methode `geefTijdelijkWoord` in klasse `WoordSpel`, waarmee je steeds de actuele status van het spel kan teruggeven, zoals bv. `"t_st_n"` in beurt 3 van het gegeven programmaverloop.
- De statische methode `maakSubset` in klasse `Woordenlijst`, waarmee je een subset maakt van de woorden met gewenste maximale lengte.