Coloured maps

In mathematics, the **four colour theorem** is a proposition that states that it is possible to colour any land map in which the countries form one unit (without the exclaves) using only four colours without giving two neighbouring countries the same colour. Here, two countries are neighbouring if they have a boundary in common, not if they are only connected by one point. You will see below that this isn't possible when using less than four colours. Luxemburg, Germany, France and Belgium are all neighbouring countries, so at least four colours are needed.

Assignment

Define a class CountryMap which can be used to represent coloured maps. The objects of this class must contain at least the following methods:

- An initializing method that allows to initialize objects based on two given text files. The locations of both these files must be given when making a new object. The first file contains a ist of all countries and the colour that the countries were coloured with. Every line of the file contains the name of a country and the name of a colour, separated by a tab. The second file contains a list of all neighbouring countries. Every line of the file contains the names of two neighbouring countries, separated by a tab.
- A method numberCountries that prints the number of countries that occur on the map. No arguments must be given to this method.
- A method numberColours that prints the number of colours that are used to colour the countries on the map. No arguments must be given to this method.
- A method colour to which the name of a country must be given. The method must print the colour that was used for the country on the map.
- A method neighbours to which the name of a country must be given. The method must print a collection that contains all neighbouring countries of the given country.
- A method areNeighbours to which the names of two countries must be given. The method must print a Boolean value, that indicates whether two countries are neighbours on the map.
- A method invalidNeighbours to which no arguments must be given. The method must print a list of all neighbouring countries that are indicated with the same colour on the map. Every element of this list is a tuple of three elements, where the first two elements represent the names of the neighbouring countries that were indicated with the same colour on the map.

This colour is the third element in the tuple. The names of the neighbouring countries must be ordered alphabetically in the tuples, and the list must be sorted, first on the name of the first country and then on the name of the second country.

When teh name of a country is given as an argument to a method, the method must verify whether that name occurs in the list of countries from the first text file that was given when making the object of the class CountryMap. If this isn't the case, the method should raise an AssertionError with the text unknown country.

Example

In the example session below, we assume that the text files colours1.txt, colours2.txt and neighbouringcountries.txt are situated in the current directory.

```
>>> countrymap = CountryMap('colours1.txt', 'neighbouringcountries.txt')
>>> countrymap.numberCountries()
208
>>> countrymap.numberColours()
5
>>> countrymap.colour('Belgium')
'rood'
>>> countrymap.neighbours('Belgium')
{'Netherlands', 'Germany', 'Luxembourg', 'France'}
>>> countrymap.areNeighbours('Belgium', 'Germany')
True
>>> countrymap.areNeighbours('Belgium', 'Italy')
False
>>> countrymap.invalidNeighbours()
[]

>>> countrymap = CountryMap('kleuren2.txt', 'buurlanden.txt')
>>> countrymap.invalidNeighbours()
[('Angola', 'Democratic Republic of the Congo', 'blauw'), ('Democratic Republic of the Congo', 'Uganda', 'blauw')]

>>> countrymap.colour('Oz')
Traceback (most recent call last):
AssertionError: unknown country
>>> countrymap.neighbours('Oz')
Traceback (most recent call last):
AssertionError: unknown country
>>> countrymap.areNeighbours('Belgium', 'Oz')
Traceback (most recent call last):
AssertionError: unknown country
>>> countrymap.areNeighbours('Oz', 'Belgium')
Traceback (most recent call last):
AssertionError: unknown country
```

De **vierkleurenstelling** is een stelling uit de wiskunde die zegt dat het mogelijk is om elke willekeurige landkaart waarin de landen elk een geheel vormen (dus zonder exclaves), met behulp van slechts vier kleuren zo kunnen ingekleurd worden dat geen twee aangrenzende landen dezelfde kleur krijgen. Twee landen gelden hierbij als aangrenzend als ze een stuk grens gemeen hebben, niet als ze slechts met een punt aan elkaar verbonden zijn. Dat het met minder dan vier kleuren niet mogelijk is, zie je direct in onderstaande kaart met Luxemburg, Duitsland, Frankrijk en België. Elk land grenst aan alle andere landen, dus zijn er minstens vier kleuren nodig.

Opgave

Definieer een klasse Landkaart waarmee ingekleurde landkaarten kunnen voorgesteld worden. De objecten van deze klasse moeten minstens over de volgende methoden beschikken:

- Een initialisatiemethode die toelaat om objecten te initialiseren op basis van twee gegeven tekstbestanden. De bestandslocaties van deze twee bestanden moeten doorgegeven worden bij het aanmaken van een nieuw object. Het eerste bestand bevat een lijst van alle landen en de kleur waarmee de landen werden ingekleurd. Elke regel van het bestand bevat de naam van een land en de naam van een kleur, van elkaar gescheiden door een tab. Het tweede bestand bevat een lijst van alle buurlanden. Elke regel van het bestand bevat de namen van twee buurlanden, van elkaar gescheiden door een tab.
- Een methode aantalLanden die teruggeeft hoeveel verschillende landen er op de landkaart voorkomen. Er moeten geen argumenten aan deze methode doorgegeven worden.
- Een methode aantalKleuren die teruggeeft hoeveel verschillende kleuren er gebruikt werden om de landen op de landkaart in te kleuren. Er moeten geen argumenten aan deze methode doorgegeven worden.
- Een methode kleur waaraan de naam van een land moet doorgegeven worden. De methode moet de kleur teruggeven waarmee het land op de kaart werd ingekleurd.
- Een methode buren waaraan de naam van een land moet doorgegeven worden. De methode moet een verzameling teruggeven die alle buurlanden bevat van het opgegeven land.
- Een methode zijnBuren waaraan de namen van twee landen moeten doorgegeven worden. De methode moet een Booleaanse waarde teruggeven, die aangeeft of de twee landen al dan niet buurlanden zijn op de kaart.
- Een methode ongeldigeBuren waaraan geen argumenten moeten doorgegeven worden. De methode moet een lijst van alle buurlanden teruggeven die met dezelfde kleur werden ingekleurd op de kaart. Elk element van deze lijst is een tuple van drie elementen, waarvan de eerste twee elementen namen van buurlanden voorstellen die met dezelfde kleur werden ingekleurd. Deze kleur is dan het derde element van het tuple. De namen van de buurlanden moeten alfabetisch gerangschikt staan in de tuples, en de lijst moet gesorteerd zijn, eerst op naam van het eerste land en vervolgens op naam van het tweede land.

Telkens wanneer de naam van een land als argument aan een methode doorgegeven wordt, moet de methode nagaan of die naam voorkwam in de lijst van landen uit het eerste tekstbestand dat werd doorgegeven bij het aanmaken van het object van de klasse Landkaart. Indien dit niet het

geval is, dan moet de methode een AssertionError opwerpen met de tekst onbekend land.

Voorbeeld

Bij onderstaande voorbeeldsessie gaan we ervan uit dat de tekstbestanden kleuren1.txt, kleuren2.txt en buurlanden.txt zich in de huidige directory bevinden.

```
>>> kaart = Landkaart('kleuren1.txt', 'buurlanden.txt')
>>> kaart.aantalLanden()
208
>>> kaart.aantalKleuren()
5
>>> kaart.kleur('Belgium')
'rood'
>>> kaart.buren('Belgium')
{'Netherlands', 'Germany', 'Luxembourg', 'France'}
>>> kaart.zijnBuren('Belgium', 'Germany')
True
>>> kaart.zijnBuren('Belgium', 'Italy')
False
>>> kaart.ongeldigeBuren()
[]

>>> kaart = Landkaart('kleuren2.txt', 'buurlanden.txt')
>>> kaart.ongeldigeBuren()
[('Angola', 'Democratic Republic of the Congo', 'blauw'), ('Democratic Republic of the Congo', 'Uganda', 'blauw')]

>>> kaart.kleur('Oz')
Traceback (most recent call last):
AssertionError: onbekend land
>>> kaart.buren('Oz')
Traceback (most recent call last):
AssertionError: onbekend land
>>> kaart.zijnBuren('Belgium', 'Oz')
Traceback (most recent call last):
AssertionError: onbekend land
>>> kaart.zijnBuren('Oz', 'Belgium')
Traceback (most recent call last):
AssertionError: onbekend land
```