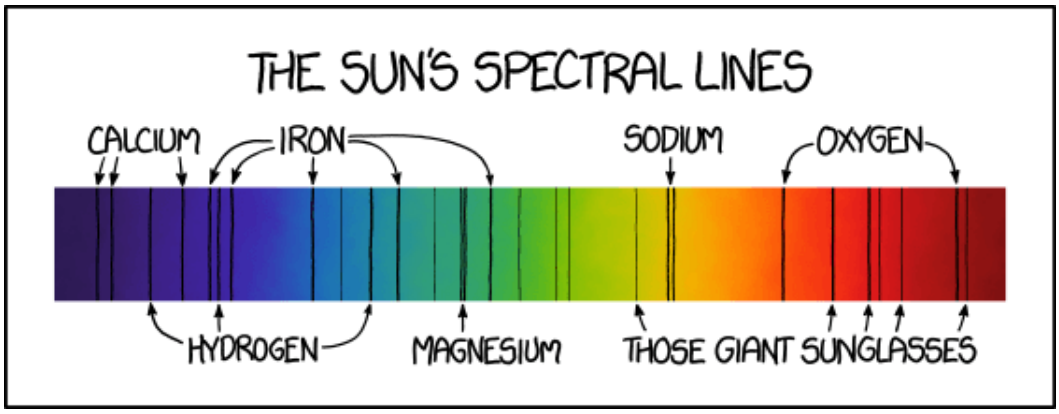
Fraunhofer lines

According to Wikipedia:

In physics and optics, Fraunhofer lines are a set of spectral lines named after the German physicist Joseph von Fraunhofer (1787–1826). The lines were originally observed as dark features (absorption lines) in the optical spectrum of the Sun.

This comic depicts the Fraunhofer lines, i.e. the spectral lines seen when sunlight is split in a spectrometer. These appear as black gaps in the rainbow of light, caused by light being absorbed by elements in the Sun.



The motivation for this problem comes from Fraunhofer lines. We simulate a light spectrum as an array of 20 boolean values representing the 20 bands in the spectrum. The bands are numbered 0 through 19 and a true value indicates light **IS NOT** absorbed and a false value indicates light **IS** absorbed.

In this problem you have been given the completed `Element` class. (You are **NOT** to modify the `Element` class). The `Element` class is used to represent each element containing the name (or atomic symbol) and the bands, an `int array` containing the band(s) absorbed by the element.

You are to complete the `FraunhoferLines` class which is used to extract information from an optical spectrum. When a `FraunhoferLines` object is created it is passed a `List<Elements>`. This List contains all the Elements to be considered in solving each method. In this problem you are to implement three methods in the `FraunhoferLines` class.

The first method to implement is the `getPossibleElements(int[]spectrum)`. `getPossibleElements` returns a `List<Element>` containing all `Element`s whose absorption bands coincide with the optical spectrum represented by the parameter `spectrum`. That is, a List of all `Element`s, `elem`, such that:

```
spectrum[elem.getBand()[k]] == false,  0 <= k < elem.getBands().length.
```

In the following example, 12 Elements are passed to the constructor. The optical spectrum being processed by `getPossibleElements` contains 6 bands being absorbed. The six bands are bands: 2, 4, 5, 7, 10 and 11. O and Si are the only two elements whose bands are a subset of those six bands. Therefore `getPossibleElements` returns a List containing only two elements.

See next page for sample code.

Fraunhofer lines

The following code shows the results of the `getPossibleElements` method.

The following code	Returns

`List<Element> els = new ArrayList<Element>();`
`els.add(new Element("H",  new int[] {2, 5, 16} ));`
`els.add(new Element("He", new int[] {1, 5, 15} ));`
`els.add(new Element("C",  new int[] {1, 13, 19} ));`
`els.add(new Element("N",  new int[] {2, 8, 17} ));`
`els.add(new Element("O",  new int[] {2, 5, 7} ));`
`els.add(new Element("Ne", new int[] {0, 4, 6, 11, 17} ));`
`els.add(new Element("Si", new int[] {4, 10, 11} ));`
`els.add(new Element("Cl", new int[] {1, 5, 14, 18} ));`
`els.add(new Element("Ar", new int[] {0, 3, 9, 12, 17, 18} ));`
`els.add(new Element("Zn", new int[] {0, 3, 9, 18, 19} ));`
`els.add(new Element("Kr", new int[] {1, 9, 17} ));`
`els.add(new Element("Xe", new int[] {0, 3, 9, 11, 18} ));`
 	
`FraunhoferLines fl = new FraunhoferLines(els);`
 	
`int[] spectrum = { true,  true,  false, true, false,`
`false, true,  false, true, true,`
`false, false, true,  true, true,`
`true,  true,  true,  true, true  };`
 	
`List<Element> ans = fl.getPossibleElements(spectrum);`	

`ans.size());`	2
`ans.contains(new Element("O",  new int[] {2, 5, 7} ));`	`true`
`ans.contains(new Element("Si", new int[] {4, 10, 11}));`	`true`
`ans.contains(new Element("Ar", new int[] {0, 3, 9, 12, 17, 18}));`	`false`
`ans.contains(new Element("Cl", new int[] {1, 5, 14, 18}));`	`false`

Problem continues on next page

Fraunhofer lines

The second method to implement is the `getRequiredElements(int[]spectrum)`. `getRequiredElements` returns a `List<Element>` containing all `Element`s that must be included to create the absorption bands in the parameter `spectrum`. That is, a `List` of all `Element`s such that if not included, the optical spectrum represented by the parameter `spectrum` would not be possible.

In the following example, 12 Elements are passed to the constructor. The optical spectrum being processed by `getRequiredElements` shows bands 0, 3, 9, 11, 12, 17, 18, 19 being absorbed. In this optical spectrum, `C` and `Zn` are the only two elements that absorb band 19. However, `C` also absorbs band 1 and 13 which are not being absorbed in this optical spectrum therefore eliminating `C` as a possible Element. Zn is the only remaining element that absorbs band 19 and is a possible element, therefore `getRequiredElements` returns a List containing only the element Zn. Please note, that while

The following code shows the results of the `getRequiredElements` method.

The following code	Returns

`List<Element> els = new ArrayList<Element>();`
`els.add(new Element("H", new int[] {2, 5, 16} ));`
`els.add(new Element("He", new int[] {1, 5, 15} ));`
`els.add(new Element("C", new int[] {1, 13, 19} ));`
`els.add(new Element("N", new int[] {2, 8, 17} ));`
`els.add(new Element("O", new int[] {2, 5, 7} ));`
`els.add(new Element("Ne", new int[] {0, 4, 6, 11, 17} ));`
`els.add(new Element("Si", new int[] {4, 10, 11} ));`
`els.add(new Element("Cl", new int[] {1, 5, 14, 18} ));`
`els.add(new Element("Ar", new int[] {0, 3, 9, 12, 17, 18} ));`
`els.add(new Element("Zn", new int[] {0, 3, 9, 18, 19} ));`
`els.add(new Element("Kr", new int[] {1, 9, 17} ));`
`els.add(new Element("Xe", new int[] {0, 3, 9, 11, 18} ));`
 	
`FraunhoferLines fl = new FraunhoferLines(els);`
 	
`boolean[] spectrum1 = { false, true, true, false, true,`
`true, true, true, true, false,`
`true, false, false, true, true,`
`true, true, false, false, false };`
 	
`List<Element> ans = fl.getRequiredElements(spectrum1);`	

`ans.size();`	1
`ans.contains(new Element("Zn", new int[] {0, 3, 9, 18, 19} ));`	true

Problem continues on next page

Fraunhofer lines

The third method to implement is the `getMissingBands(int[]spectrum)`. `getMissingBands` returns a `List<Integer>` containing all bands that cannot be accounted for from the `List<Element>` passed to the constructor. In the following example, bands 0, 2, 5, 6, 7, 8, 16, and 17 are being absorbed. Six of the bands can be accounted for by the elements H, N, and O. Bands 2, 5 and 16 can be accounted for by the Element H. Bands 2, 8 and 17 can be accounted for by the Element N. Bands 2, 5 and 7 can be accounted for by the Element O. This leaves bands 0, and 6 as unaccounted or missing. Since the only elements which absorb band 0 also absorb band 3 or 4, band 0 is a missing band. The only elements that absorb band 6 also absorb bands 4, 11 and 17, band 6 is a missing band. Therefore, `getMissingBands` returns a List containing only the `Integer`s 0 and 6

The following code shows the results of the `getRequiredElements` method.

The following code	Returns

`List<Element> els = new ArrayList<Element>();`
`els.add(new Element("H",  new int[] {2, 5, 16} ));`
`els.add(new Element("He", new int[] {1, 5, 15} ));`
`els.add(new Element("C",  new int[] {1, 13, 19} ));`
`els.add(new Element("N",  new int[] {2, 8, 17} ));`
`els.add(new Element("O",  new int[] {2, 5, 7} ));`
`els.add(new Element("Ne", new int[] {0, 4, 6, 11, 17} ));`
`els.add(new Element("Si", new int[] {4, 10, 11} ));`
`els.add(new Element("Cl", new int[] {1, 5, 14, 18} ));`
`els.add(new Element("Ar", new int[] {0, 3, 9, 12, 17, 18} ));`
`els.add(new Element("Zn", new int[] {0, 3, 9, 18, 19} ));`
`els.add(new Element("Kr", new int[] {1, 9, 17} ));`
`els.add(new Element("Xe", new int[] {0, 3, 9, 11, 18} ));`
 	
`FraunhoferLines fl = new FraunhoferLines(els);`
 	
`boolean[] spectrum2 = {false, true,  false, true,  true,`
`false, false, false, false, true,`
`true,  true,  true,  true,  true,`
`true,  false, false, true,  true};`
 	
`List<Integer> bandAns = fl.getMissingBands(spectrum2);`	

`bandAns.size();`	2
`bandAns.contains( new Integer(0) )`	`true`
`bandAns.contains( new Integer(6) )`	`true`