ECS510 Algorithms and Data Structures in an Object-Oriented Framework

Exercise Sheet 5: Implementing a `Dictionary` class

This is an exercise to support the "Implementing Objects" section

Write a class `Dictionary` that represents a set of words stored with their equivalents. The internal data structure should use arrays, but not `ArrayList<E>` or any other built-in type in Java.

The class `Dictionary` should provide methods with the following headers:

```
void add(String word1, String word2)
boolean contains(String word)
String equiv(String word) throws NotFoundException
void remove(String word) throws NotFoundException
boolean same(String word1, String word2) throws NotFoundException
int size()
```

The method `add` takes two arguments, the first is a word and the second is its equivalent to be stored in the dictionary. It could be that the first word is a word in English and the second a word in some other language, or it could be something similar where one `String` value links to another, for example the first `String` could be a module code and the second `String` a module name.

The method `contains` returns `true` if there is an equivalent to its argument in the dictionary, and `false` otherwise. So if its argument has previously been used as the first argument to a call of `add` on the `Dictionary` object it has been called and there has been no subsequent call of `remove` with that argument it should return `true`, otherwise it should return `false`.

The method `equiv` returns the word that is the equivalent stored in the dictionary to its argument. So if `add` was previously called on the `Dictionary` object with the argument to it being the argument to the call to `equiv`, it should return what was the second argument to that call of `add`. If there is no equivalent (that is, no previous call of `add` with the argument to the call of `equiv` as its first argument), it should throw a `NotFoundException`.

The method `remove` should take the word given as its argument out of the dictionary. So, if `add` had been called previously on the `Dictionary` object with the first argument to it being the argument to `remove`, a subsequent call of `contains` with the same argument should return `false`. If there is no equivalent of its argument stored, the method `remove` should throw a `NotFoundException`.

The method `same` should return `true` if the dictionary stores its first argument with its second as its equivalent, and `false` if it stores its first argument but with a different word as its equivalent. If it does not store its first argument with an equivalent, it should throw a `NotFoundException`.

The method `size` should return the total number of words stored in the dictionary.

You should define your own class `NotFoundException`, with its constructor taking the word that is not found as its argument.

Note, there are more efficient data structures than arrays that could be used for implementation here, but for exercise purposes even if you have knowledge of such data structures from previous study, you must use arrays here. Also, it would be trivial to implement it using an appropriate class as provided in Java's Collections Framework, but the point of this exercise is to build an implementation without using anything but arrays. For the purpose of this exercise, a simple implementation is fine, you are not expected to use any technique not covered in the teaching to produce a more efficient implementation.

You should provide the methods as requested. There may be some ambiguities or uncertainties in the specification, if you see them you should note them and how you have handled them.

An example of how the code should work is given overleaf.

Example:

The following code:

```
Dictionary dict = new Dictionary();
String module;
dict.add("ECS510","ADSOOF");
dict.add("ECS414","OOP");
dict.add("ECS505","Software Engineering");
dict.add("ECS607","Data Mining");
System.out.println("Number of modules is: "+dict.size());
try {
   module = dict.equiv("ECS505");
   System.out.println("The module with code ECS505 is: "+module);
   if(dict.contains("ECS410"))
      {
       module=dict.equiv("ECS410");
       System.out.println("The module with code ECS410 is: "+module);
      }
   else
      System.out.println("There is no module with code ECS410");
   if(dict.same("ECS414","OOP"))
      System.out.println("The module with code ECS414 is: OOP");
   else
      System.out.println("The module with code ECS414 is not OOP");
   dict.remove("ECS510");
   module=dict.equiv("ECS510");
   System.out.println("The module with code ECS510 is: "+module);
  }
catch(NotFoundException e)
   {
    module=e.getMessage();
    System.out.println("There is no module with code "+module);
   }
```

when executed should result in:

```
   Number of modules is: 4
   The module with code ECS505 is: Software Engineering
   There is no module with code ECS410
   The module with code ECS414 is: OOP
   There is no module with code ECS510
```

being printed.

This code and further code to test your implementation is available as `DictionaryTest1.java` and `DictionaryTest2.java` from the code folder for the "Implementing Objects" section of the module website.

Matthew Huntbach