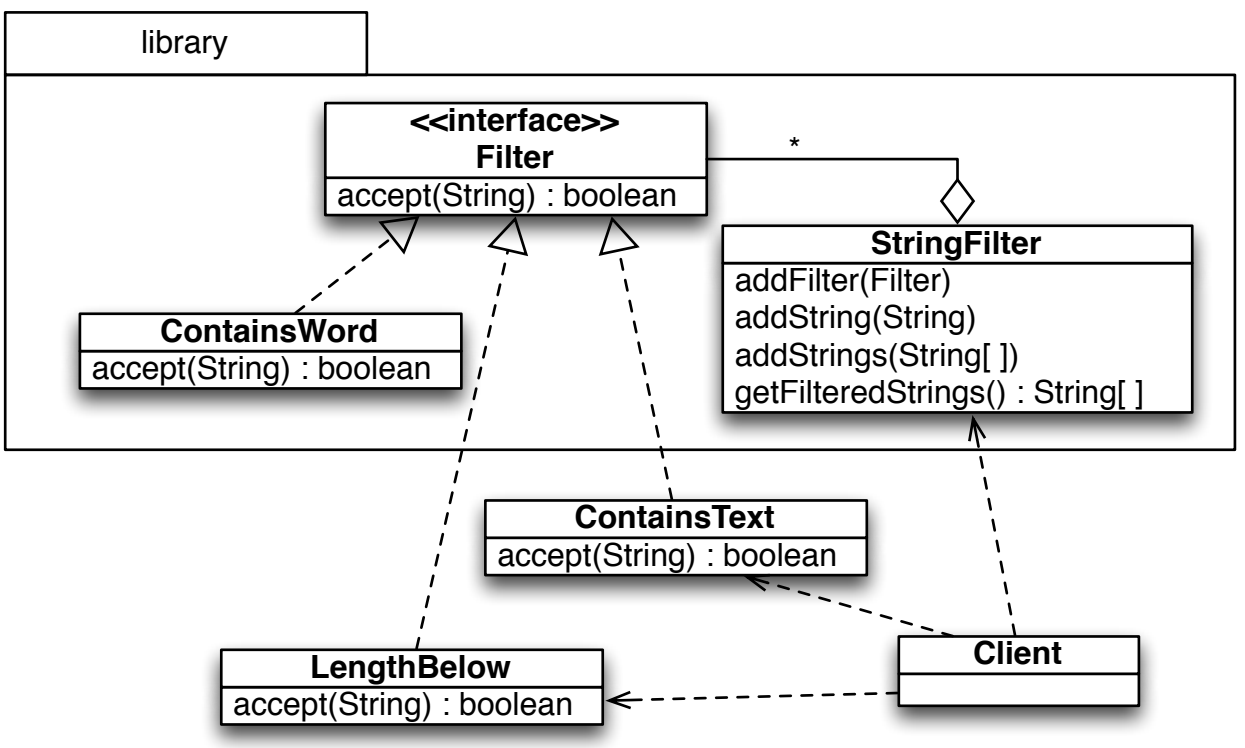
Java review exercise

The goal is to develop a little toy-library for String filtering. Clients of this library should be able to configure the library with arbitrary String filters, so that the solution is flexible regarding the types of filters on can use.

The diagram illustrates a possible design solution. The **StringFilter** is the class where clients may add several specific word filters, represented by the interface **Filter**. Notice that the **StringFilter** does not depend on any specific filter. As an example, the **ContainsWord** filter is a standard filter provided by the library, while **ContainsText** and **LengthBelow** are developed by the client.



Example of how the library can be used:

```
import library.Filter;
import library.StringFilter;

public class Client {

	public static void main(String[] args) {

		StringFilter sFilter = new StringFilter();

		sFilter.addWord("UM");
		sFilter.addWord("DOIS TRES");
		sFilter.addFilter(new ContainsText("B")); // custom filter
		System.out.println("1: " + sFilter.getFilteredStrings()); //  [UM, DOIS TRES]

		sFilter.addFilter(new ContainsWord("TRES", "U")); // filter offered by the library
		System.out.println("2: " + sFilter.getFilteredStrings()); // [UM]

		ArrayList<String> list = new ArrayList<>();
		list.add("QUATRO");
		list.add("CINCO");
		sFilter.addStrings(list);
		System.out.println("3: " + sFilter.getFilteredStrings()); // [QUATRO, UM, CINCO]

		sFilter.addFilter(new LengthBelow(3)); // custom filter
		System.out.println("4: " + sFilter.getFilteredStrings()); // [QUATRO, CINCO]
	}
}
```

Output:

```
1: [UM, DOIS TRES]
2: [UM]
3: [QUATRO, UM, CINCO]
4: [QUATRO, CINCO]
```