Practical 5

Comment

The purpose of this prac is to get experience with writing `JUnit` tests. The test-writing method used in this prac is called Test Driven Development, where the test class methods are written to define the desired behaviour of a class (with these tests initially failing). The class code is then written after the tests, ensuring that the pre-written tests all pass.

1 DistinctCounter

The first part of this practical will deal with a class called `DistinctCounter`. The user can add strings to this counter, and it will keep track of the distinct strings added to it in lexicographical order. It should have the following methods:

- `void add(String s);`
- `int getDistinctCount();`
- `String[] getStrings();` // in lexicographical order

Example usage:

```
DistinctCounter distinct = new DistinctCounter();
distinct.add("Z");
distinct.add("Hello");
distinct.add("Z");
distinct.add("Hello ");
distinct.getDistinctCount(); // 3
distinct.getStrings(); // {"Hello", "Hello ", "Z"}
```

1. Write a stubbed version of this class. A stubbed class has all public members declared and the code compiles. (Methods that have a `void` return type should have empty implementations. Methods that have any other return type should return a simple result (e.g. `return null`)).
2. Create a JUnit 4 test class for `DistinctCounter`.
3. Write tests which define how *you want the class to behave* (at this point you will expect that most of your tests will fail against the stubbed class, but your tests should compile at least).
4. Properly implement the methods of `DistinctCounter` – make sure they pass all your tests (Hint: look at `TreeSet` in the Java Collections library.)
5. Reimplement `getStrings()` without using `Set.toArray() or List.toArray()`. Alternatively, if you have not used either of these, research them and attempt to use them in your `getStrings()` method.

2 PalindromeCounter

The `PalindromeCounter` class will inherit from `DistinctCounter` and will add the following extra methods:

- int getPalindromeCount(); // number of unique palindromes
- String[] getPalindromes(); // distinct palindromes
- String[] getNonPalindromes(); // distinct non-palindromes

1. Write an implementation with stubs for the new methods.
2. Create a JUnit 4 test class for `PalindromeCounter`.
3. Write test methods.
4. Implement and test `PalindromeCounter`'s functionality.