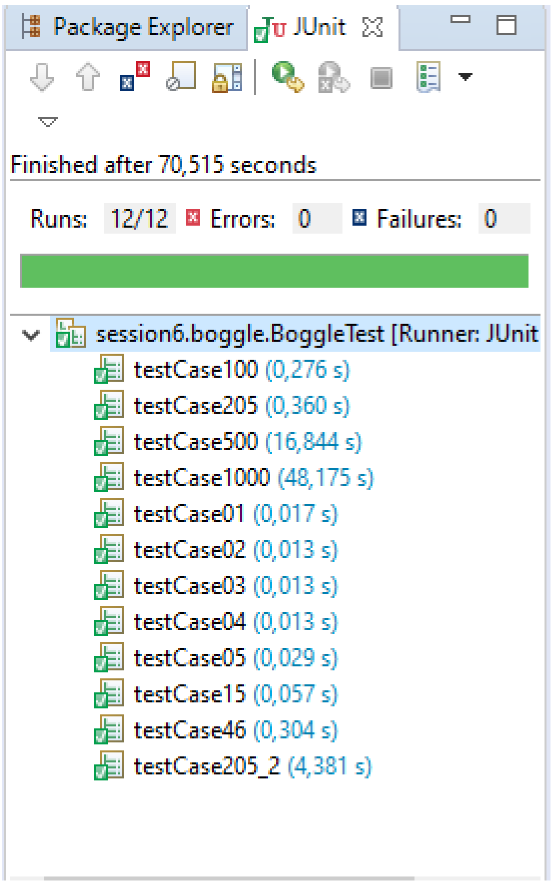
Algorithmics	Student information	Date	Number of session
	UO: 271580	01/04/2020	6
	Surname: Lopez Amado		
	Name: Pablo		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Test Cases



```
Finished after 70,515 seconds

Runs: 12/12   Errors: 0   Failures: 0

session6.boggle.BoggleTest [Runner: JUnit
    testCase100 (0,276 s)
    testCase205 (0,360 s)
    testCase500 (16,844 s)
    testCase1000 (48,175 s)
    testCase01 (0,017 s)
    testCase02 (0,013 s)
    testCase03 (0,013 s)
    testCase04 (0,013 s)
    testCase05 (0,029 s)
    testCase15 (0,057 s)
    testCase46 (0,304 s)
    testCase205_2 (4,381 s)
```

Algorithmics	Student information	Date	Number of session
	UO: 271580	01/04/2020	6
	Surname: Lopez Amado		
	Name: Pablo		

Activity 2. Times for different executions

n	time
8	20
16	100
32	158
64	723
128	1707
256	8769
512	20629
1024	42939
2048	71772

Considering the results obtained it looks like this algorithm follows a factorial complexity but of course the times are quite lower than the maximum expected since strings are maximum of length ten, then not the hole matrix is inspected on each iteration.

Activity 3. Code improvements.

For the storage of the dictionary I choose a mixed data structure. For one length and two length words I created an arraylist per each. Then, for words having 3 or more letters I created a HashMap because is quite fast for searching, where the key is the first 3 characters of the string and the value is an arraylist (because is also fast for searching) where all strings starting by the key are stored.

Related to the boggle implementation I made also some improvements. Of course I put a maximum word length of 10 for bigger words not to be taken into account. Then every time I get a word I check in my data structures if that word explicitly exists or if there is a word which starts with the same letters the current string is starting (f.e *pre* will return true because *pregnant* exists). With this improvement I get I huge reduction in the times. Finally I achieved to reduce times much more (as shown before testCase1000 is faster than the one you showed us an I'm very proud 😊) by removing those strings that I already have in my solutions because that way when searching for a new string that will not appear and times will be better.