Austin Kelly

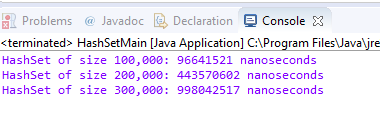
CS249

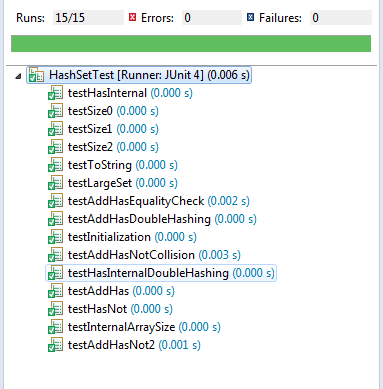
12/8/16

**Project 5**

**Hash Set**

The first section of our project had us create a custom HashSet from the ground up based on an interface given. A HashSet is an unordered collection that cannot have repeated values. I approached this by creating a generic array and having a constructor that initialized it by size. My add method got the hashcode of the parameter and if the element wasn’t already contained, entered the value. I had a separate gethashcode method that made the code and set it nonnegative. In my main method, I timed the speed of these, and added the portion to validate scrabble words. The tests grew linearly, but favored the middle test of 200,000 which can be seen in the analysis below. I was able to pass every single test and get a flawlessly running program.





Big O review:

gethashcode= OnLogn

gethashcode= OnLogn

add() = O(n)

has() = O(n^2)

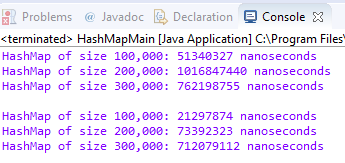
size() = O(1)

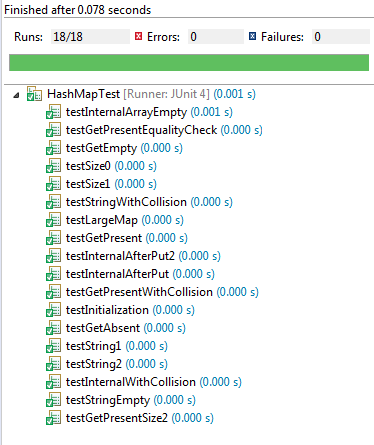
getInternalArray() = O(1)

toString() = O(1)

**Hash Map**

The second section tasked us with creating a Hash Map. Hash Maps are similar to dictionaries in python. I approached this by creating an array of MapPair items which both had a key and element. This was set up in the subclass. I made a constructor that initialized it by size. My put method got the hashcode of the parameter and if the element wasn’t already contained, entered the value. I had a separate gethashcode method that recursively called itself passing in a string and set it nonnegative. In my main method, I timed the speed of these using the required initial sizes, resulting in 6 tests. These grew linearly, but also favored the middle test of each subset slightly which can be seen in the analysis. In the end, I was finally able to pass every single test and get a flawlessly running program.





Big O review:

gethashcode= O(n)

put() = O(n)

get() = O(n^2)

size() = O(1)

toString() = O(1)

getKey = O(1)

getElement = O(1)

next = O(1)

toString() = O(1)