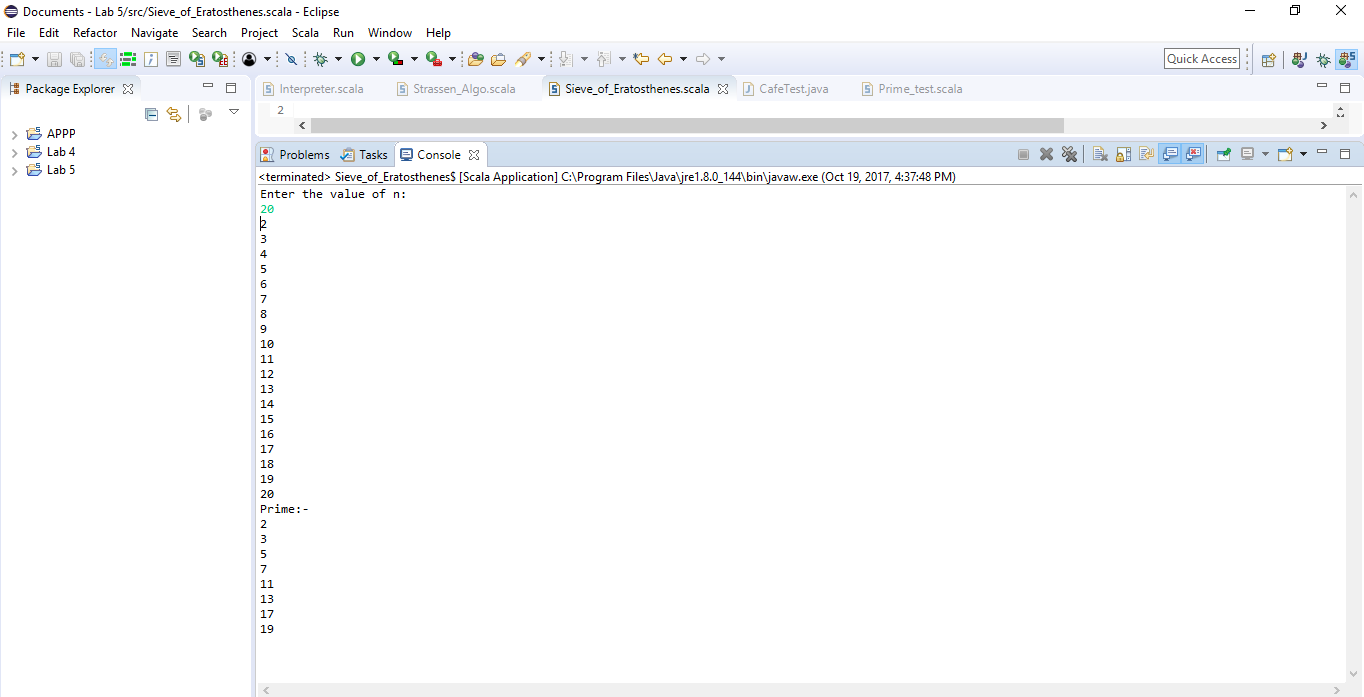
**Github link**: https://github.com/mujtaba-faizi/Lab-5

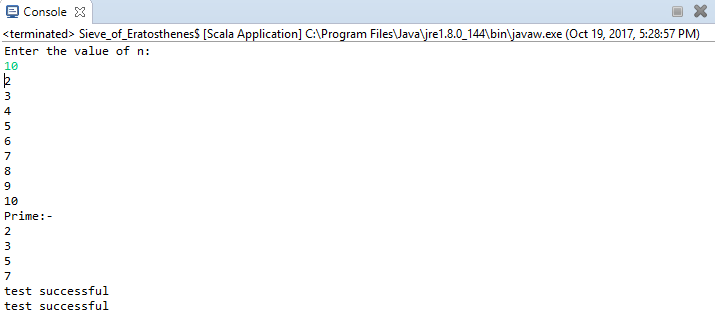
**Introduction**: In this lab, we should develop and test a Scala based application, using functional programming paradigm, to calculate prime numbers between 1 to a given large n, using the Sieve of Eratosthenes, method.

**Approach**: In Scala, I have used the functional paradigm to implement the algorithm Sieve of Eratosthenes. I have used lists as datastructure to hold integers; recursion two times and the built-in method filterNot to filter out the integers whose multiples doesn’t exist in the list (i.e. whose mod with the iterating values (2 onwards till n) is 0); and the integer itself is not removed as well. .

**Design:** I have defined a getList() function to take a list and recursively, append the integers from 2 to n in a new list each time and returns the new list with all integers from 2 to n when the iterator reaches value greater than n. I defined a remMultiple() function that recursively, removes all non-prime numbers. I have written unit tests for getList() and remMultiple() functions in the same Scala file containing code for implementing the algo: Sieve of Eratosthenes.

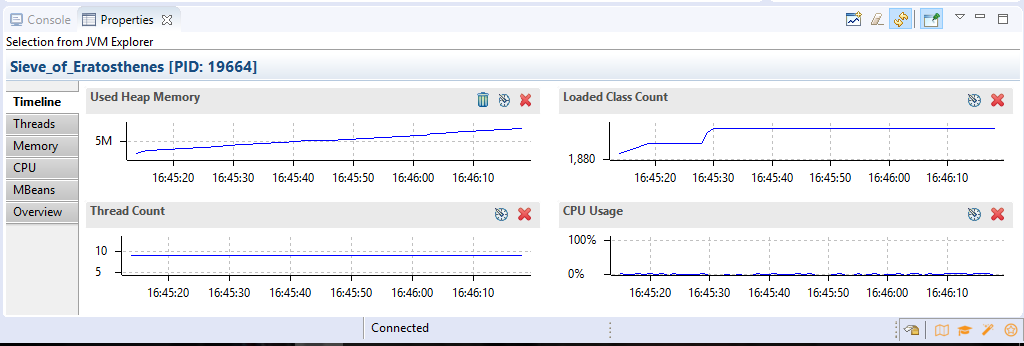
**How to Run:** The Scala code file will execute and ask for input. User has to enter the number till which he wants to access prime numbers from 0 i.e. “n”. The program will first display all numbers from 2 to n i.e. in form of elements of list and then the prime numbers from 2 to n.



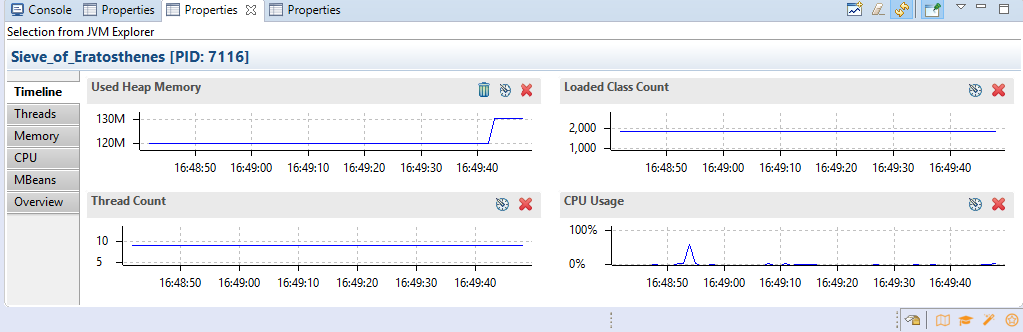


**Analysis:** Profiling of used Heap memory and CPU Usage for different input schemes are shown below.

For n=1000



For n=10000



For n=1000000

