HW 1 Big O Notations

1. Compare sequential search and binary search algorithms over 20 arrays.
    a. **Preparation**: Create 20 integer arrays of the same length of 1000. Fill them sequentially starting from index 0 with the following number of random integers: 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, and 1000, respectively. The integers used should be within the range from 0 to 1,000,000 but not include 1,000,000.

    b. **Sequential Search**: Perform a sequential search of 20 integers within each array. Make sure these 20 values are those that were used to fill each corresponding array in step one. Display, in a table format, the average time in nano seconds it takes for searching these 20 values in each array. Discuss the relationship between the average time used and the size of the array using a README.txt file.

    b. **Binary Search**: Sort each of the above 20 arrays using the **Arrays.sort()** method. Then search the same 20 integer values you used in the previous step using the Binary Search algorithm for each array. Get the corresponding average time in nano seconds again. Display, in a table format, the average time used for each array. Discuss the relationship between the average time used and the size of the array as you did in the previous step using the same README.txt file.

    You will write your own implementation of sequential search.

2. Create a class named **Student**. The class contains the following three fields: name (String), gpa (double), and age (int). Create another class called **StudentBag**. The class contains a **Student** array and the following methods:
    a. public void insert(Student student)
    b. public void fill(int numberOfStudents): which fills the array with the specified number of Student objects. Each object stores the following values: a name randomly generated with an upper-case letter followed by four lower-case letters; a gpa between 0.0 and 4.0 (not including 4.0), and an age ranging randomly from 18 to 40 inclusive.
    c. public void displayStudents()
    d. public void bubbleSort()
    e. public void selectionSort()
    f. public void insertionSort()
    g. public int sequentialSearchByName(String name)
    h. public int binarySearchByName(String name)

Write a **Demo** class that creates a **StudentBag** object filled with 1000 student objects using the fill method. Demonstrate all your sorting and searching methods work properly. Discuss the Big O time complexity level for each of the above methods in a README.txt file.