```java
import java.util.HashMap;
import java.util.Random;

class Node {
    int value;
    Node next;

    Node(int value) {
        this.value = value;
        this.next = null;
    }
}

class Hashing {

    static int getRandomStudentId() {
        Random rand = new Random();
        return rand.nextInt(1000, 9999);
    }

    static int hashFunction(int num, int size) {
        return num % size;
    }

    static int searchIdUsingHashFunction(int studentId, int size, Node[] hashTable) {
        int counter = 0;
        boolean idFound = false;

        int hashIndex = hashFunction(studentId, size);

        Node travesalNode = hashTable[hashIndex];

        if (travesalNode != null) {
            counter++;
            while(travesalNode != null) {
                if (travesalNode.value == studentId) {
                    idFound = true;
                    break;
                }

                travesalNode = travesalNode.next;
                counter++;
            }

            if (idFound) {
                return counter;
            } else {
                return 0;
            }

```

```
51          }
52          return 0;
53      }
54
55
56      public static void main(String[] args) {
57          int[] studentArray = new int[500];
58
59          HashMap<Integer, Integer> hashSearchCounter = new HashMap<>();
60          HashMap<Integer, Integer> linearSearchCounter = new HashMap<>();
61
62          // Generate an array of students with random roll numbers
63          for(int i = 0; i < 500; i++) {
64              studentArray[i] = Hashing.getRandomStudentId();
65          }
66
67          System.out.print("Printing roll numbers of all students: ");
68          for(int i = 0; i < 500; i++) {
69              System.out.print(studentArray[i] + ", ");
70          }
71
72          // Prime number between 128 and 256
73          int size = 139;
74          Node[] hashTable = new Node[size];
75
76          // Filling the hashtable
77          int hashIndex;
78          for(int i = 0; i < 500; i++) {
79              hashIndex = hashFunction(studentArray[i], size);
80              Node tempNode = new Node(studentArray[i]);
81
82              // If the hash index is not found set the tempNode as the first element
83              if(hashTable[hashIndex] == null) {
84                  hashTable[hashIndex] = tempNode;
85              } else {
86                  // If the hash index is found, traverse till the end of the list and
    then add the new node
87                  Node traversalNode = hashTable[hashIndex];
88                  while(traversalNode.next != null) {
89                      traversalNode = traversalNode.next;
90                  }
91
92                  traversalNode.next = tempNode;
93              }
94          }
95
96          System.out.println("\n");
97          System.out.println("Displaying the Hashtable Index with all the contents");
98
99          for(int i = 0; i < size; i++) {
```

```
100             Node temp = hashTable[i];
101 
102             if (temp != null) {
103                 System.out.print(i + "-> \t");
104 
105                 while(temp.next != null) {
106                     System.out.print(temp.value + ", ");
107                     temp = temp.next;
108                 }
109                 System.out.println(temp.value);
110             }
111         }
112 
113         int foundCounter = 0, notFoundCounter = 0;
114 
115         int[] searchArray = new int[20];
116         int searchArrayIndex = 0;
117 
118 
119         System.out.println("\nSearching the elements using Hash Function");
120         while (foundCounter < 17 || notFoundCounter < 3) {
121             int searchId = getRandomStudentId();
122 
123             int counter = searchIdUsingHashFunction(searchId, size, hashTable);
124 
125             if(counter == 0 && notFoundCounter < 3) {
126                 System.out.println("ID: " + searchId + " was not found in hash table
    with counter: " + counter);
127                 hashSearchCounter.put(searchId, counter);
128                 notFoundCounter++;
129 
130                 searchArray[searchArrayIndex] = searchId;
131                 searchArrayIndex++;
132 
133             } else if (counter > 0 && foundCounter < 17) {
134                 System.out.println("ID: " + searchId + " was found in hash table with
    counter: " + counter);
135                 hashSearchCounter.put(searchId, counter);
136                 foundCounter++;
137 
138                 searchArray[searchArrayIndex] = searchId;
139                 searchArrayIndex++;
140             }
141         }
142 
143 
144         System.out.println("\nSearching the elements using Sequential Search
    Function");
145 
146         for(int i = 0; i < searchArray.length; i++) {
147             int searchId = searchArray[i];
```

```
148             int counter = 0;
149             boolean idFound = false;
150 
151             for (int j = 0; j < size; j++) {
152                 Node traversalNode = hashTable[j];
153 
154                 while(traversalNode != null) {
155 
156                     if(traversalNode.value == searchId) {
157                         idFound = true;
158                         break;
159                     }
160 
161                     traversalNode = traversalNode.next;
162                     counter++;
163                 }
164 
165                 if (idFound) {
166                     break;
167                 }
168             }
169 
170             if (idFound) {
171                 System.out.println("ID: " + searchId + " was found in hash table with
    counter: " + counter);
172                 linearSearchCounter.put(searchId, counter);
173             } else {
174                 System.out.println("ID: " + searchId + " was not found in hash table
    with counter: " + counter);
175                 linearSearchCounter.put(searchId, counter);
176             }
177         }
178 
179 
180         System.out.println("\n\nComputational complexity of Linear and Sequential
    Search");
181         System.out.println("\tNumber  |  Sequential  |  \tHash  ");
182 
183         for(int i = 0; i < searchArray.length; i++) {
184             System.out.println("\t"+ searchArray[i] + "    |\t" +
    linearSearchCounter.get(searchArray[i]) + "    |\t" +
    hashSearchCounter.get(searchArray[i]));
185         }
186     }
187 }
```