Programming and Data Structures
Programming Project 3: Hashing

Student Learning Outcomes

Students should demonstrate the following abilities:

1. Implement the class **`HashMapSC`** class using separate chaining for the resolution of the collisions.
2. Implement the class **`HashMapLP`** to use linear probing to resolve collisions.
3. Implement the class **`HashMapQP`** to use quadratic probing to resolve collisions.
4. Compare the performance of the three different implementations of the hash map.

Programming Project

1. Create the generic class **`MapEntry`** as shown in the UML diagram below:

`MapEntry<K, V>`	Generic class with two types
`key: K`	Data member **`key`** of type **`K`**
`value: V`	Data member **`value`** of type **`V`**
`MapEntry(K k, V v)`	Constructor to initialize **`key`** to **`k`** and **`value`** to **`v`**
`getKey(): K`	Returns the value of **`key`**
`getValue(): V`	Returns the value of **`value`**
`toString(): String`	Returns **`key`** and **`value`** in a formatted string

2. Use the implementation of the class **`HashMap`** seen in class to create a new class named **`HashMapSC`** (Hash Map with Separate Chaining)
3. Use the implementation of the class **`HashMapSC`** and modify it to create a new class named **`HashMapLP`** (Hash Map with Linear Probing) that handles collisions using linear probing.
4. Use the implementation of the class **`HashMapLP`** and modify it to create a new class named **`HashMapQP`** (Hash Map with Quadratic Probing) that handles collisions using quadratic probing.
5. The three classes should use the same class **`MapEntry`** described in the UML diagram above (with package access for all its members). Remove the inner private class **`MapEntry`** from the three classes.
6. Test your three versions of the class **`HashMap`** in a test program **`HashMaps`** as follows:

    a. In the main method, Instantiate the three classes for types **`<String, String>`**. Insert the data from the file `dictionary.txt` in the three hash maps.

b. Perform 100 random search operations on the three data structures. Display the number of iterations for each word searched in the three data structures. Finally, display the average number of iterations for the three data structures.

7. Discuss briefly the performance of the three implementations of the hash map for the search methods using the results of your program.

8. Submit the Java files **`HashMapSC.java`**, **`HashMapLP.java`**, **`HashMapQP.java`**, and **`HashMaps.java`** on Github.

9. A sample output is given below for illustration, but the numbers do not need to be the same for your program output.

Sample Output

```
Word                Separate Chaining   Linear Probing     Quadratic Probing
"Vault"             1                   1                   1
"Eccentricity"      1                   1                   1
"Pack"              2                   2                   2
"Salivation"        1                   1                   3
"Wantonize"         3                   1                   1
"Jowler"            1                   1                   1
"Unstudied"         1                   1                   1
"Usher"             1                   1                   1
"Raker"             1                   1                   1
"Group"             1                   1                   1
"Rather"            1                   1                   1
"Habiliment"        1                   1                   1
"Ferial"            1                   1                   1
"Rapper"            1                   2                   2
"Haemin"            1                   1                   1
"Safeguard"         2                   3                   3
"Lees"              1                   1                   1
"Justification"     1                   1                   1
"Foretokened"       1                   1                   1
"Lead"              1                   1                   2
"Utterance"         1                   1                   1
"Haematoidin"       1                   1                   1
"Homopteran"        1                   1                   1
"Rusty"             2                   2                   5
"Unexpensive"       2                   2                   2
"Jesus"             1                   1                   1
"Gainsayer"         1                   1                   1
"Hermitary"         1                   1                   1
"Rabbinism"         1                   1                   1
"Nigromancie"       1                   1                   1
"Gismondite"        1                   1                   1
"Fawning"           2                   2                   2
"Neutrality"        1                   1                   1
"Ribbon"            1                   1                   5
"Raiment"           2                   1                   1
```

"Gad"	1	1	1
"Junk"	2	2	8
"Ghostly"	1	1	1
"Vicious"	3	9	6
"Lapis"	1	1	1
"Fly"	2	1	1
"Gage"	1	1	1
"Aesculin"	1	1	1
"Unstratified"	2	1	1
"Frush"	1	1	1
"Gentile"	1	1	1
"Mace"	1	1	1
"Rhombohedric"	1	1	1
"Leaved"	2	2	2
"Habited"	1	1	1
"Upbraiding"	2	1	2
"Homomorphic"	1	1	1
"Vicissitude"	1	1	2
"Friendship"	1	1	1
"Sainted"	2	1	1
"Jeames"	1	1	2
"Alcyones"	1	1	1
"Alluded"	1	1	1
"Jockeyed"	2	1	1
"Hope"	1	1	1
"Sarcastically"	2	4	3
"Rabidness"	2	1	1
"Save"	1	1	2
"Hypogeous"	2	2	2
"Leisurably"	1	1	1
"Gentianella"	2	2	3
"Neogaean"	2	1	1
"Rehibition"	1	1	1
"Unreverend"	2	1	3
"Gain"	1	1	1
"Nervous"	1	1	1
"Air chamber"	1	1	1
"Lever"	2	1	1
"Unhinge"	1	1	1
"Undecyl"	1	1	2
"Unclothe"	2	1	1
"Rant"	1	1	1
"Abstrude"	1	1	1
"Rise"	2	2	1
"Unchild"	1	1	1
"Lexiconist"	1	1	1
"Federalizing"	1	1	1
"Latakia"	1	1	1
"Reproachablr"	1	1	1
"Yeast"	1	1	1
"Vark"	1	1	1
"Sea maw"	2	1	1
"Adulterate"	1	1	1
"Reading"	1	2	2
"Scleragogy"	3	2	2

"Vacating"	2	1	4
"For"	1	1	1
"Accidental"	1	1	1
"Late"	1	1	1
"Relique"	2	1	2
"Utility"	1	1	1
"Genethliac"	1	1	1
"Affrap"	1	1	1
"Glottis"	1	1	1
"Hert"	1	1	1
Average	1.31	1.25	1.48