CS 223 : Data Structure II

**Perfect Hashing**

Hisham Osama Ghazy (81)

Ahmed Moustafa El-Naggar (16)

# 

# Problem Statement

In this assignment, you’re required to implement a perfect hashing data structure. We say a hash function is perfect for S if all lookups involve O(1) work. In section 2, background about universal hashing is provided. Sections 3 and 4 describe two methods for constructing perfect hash functions for a given set S. You’re required to design, analyze and implement a perfect hash table as described in sections 3 and 4.

## Implementation Details

Both versions of perfect hash tables ( O(n2) , O(n) ) are implemented using the same class : PerfectHashTable. For choosing one of the two available versions, a builder class following the builder design pattern is used for that purpose with two static factory methods available:

1. buildSquareSpaceHashTable : returns a perfect hash table that uses O(n2) space.
2. buildLinearSpaceHashTable : returns a perfect hash table that uses O(n) space.

Hash functions used in hashing are universal hash functions.

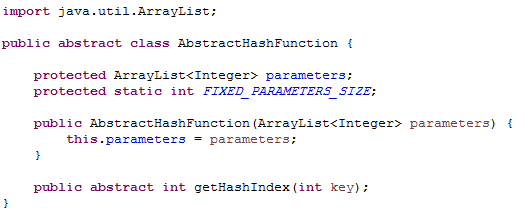
**PerfectHashTable Class:**



Attributes:

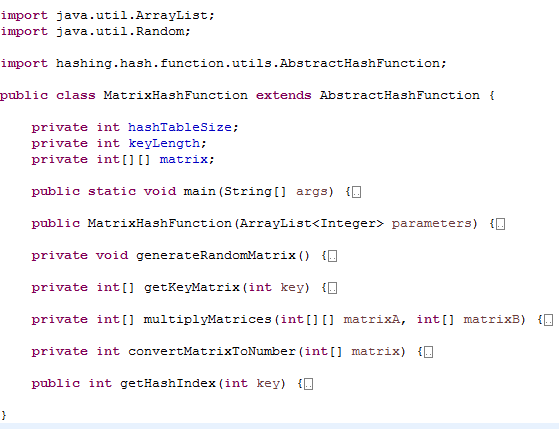
1. hashFunction : the hash function to be used.
2. hashTables : the second level of hash tables in case of collisions.
3. Values : array storing the keys in case of no collisions in some table cells.

**Abstract Hash Function:**



Attributes:

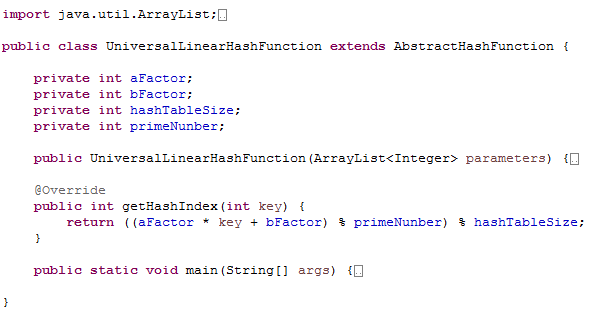
1. Parameters: list of the parameters needed for any hash function
2. FIXED\_PARAMETERS\_SIZE : size of the parameters list needed for any hash function

**Matric Hash Function:**

Attributes:

1. hashTableSize : the log of the size of the hash table.
2. keyLenght : the number of bits in the input keys.
3. Matrix : the matrix to be multiplied by the key matrix.

**UniversalLinearHashFunction**

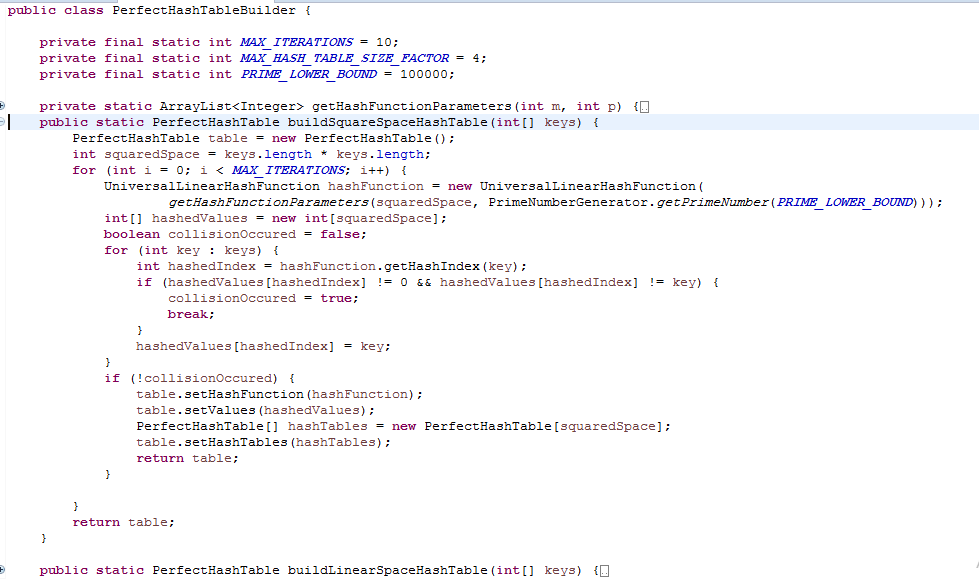


**Equation =**

Attributes:

1. primeNumber : prime number greater that all input keys.
2. hashTableSize : the size of the hash table.

**PerfectHashTableBuilder**



Static Factory methods for Hash Tables creation.

### 

### 

### 

### 

### Test Cases

**In the Square-Sized Hashing:**

In most of the test cases, **no collisions** have occurred in a table of space.

In few cases, the no collision cases have been achieved after rebuilding the hash table 2 or 3 times.

Rebuilding the hash tables many times has occurred in the cases where the number of keys is relatively small ( keys )

**In the Linear-Sized Hashing**

**By calculating where is the number of collided elements in the row of the space table**

|  |  |
| --- | --- |
| File Name |  |
| keys.txt  Keys1001000.txt  Keys10001000.txt  Keys100001000000.txt | 15  100  750  10000 |

I.e.: This results are obtained after building the linear spaced table **10 times** and taking the average value of

### 

### Assumptions

* Duplicates keys are ignored and only **one value** of the key is taken into consideration.
* No negative keys are allowed.
* In spaced hash table, several trials are made until value is less than 4.