DA-ALG1000
Oblig 3

Amer Sisic

Summary:
This report is showing how chaining technique is implemented in java, using classes from previous tasks such as SingleLinkedList and Node.

Contents

Task given

Obligatorisk oppgave nr. 3 i DA-ALG1000 vår 2015

Oppgave innen Hashing.

Skriv et program som benytter teknikken for 'Chaining' når det gjelder hashing. Programmet skal lese inn lengden på tabellen som skal inneholde referanse til hver enkelt lenket liste som skal genereres. Videre skal alle verdier, som skal lagres, leses inn. En kjøring bør typisk lese inn ca. 15-20 verdier. Programmet skal ha en egen funksjon/metode for hashing der indeksen finnes. Når programmet har generert de lenkete listene, skal teoretisk 'load factor' beregnes og skrives ut. Hele tabellen skal skrives ut på enkel måte. Programmet skal benytte klassene for SingelLinkedList og Node som du brukte i 1. obligatoriske oppgave. Rapport skal leveres med og denne skal inneholde eksempel på kjøring av programmet. Det er ikke nødvendig med GUI-løsning.

What is hashing and chaining

Storing data in a way user can find it again easy and efficient can be tricky. One way to do this is by using hashing methods in java or other languages. The way hashing works is by making an array with a fixed size, and when user puts in values the hashing method can calculate a "key" by using modulo of the table size with the given value (key=givenValue%tableSize).

That key is then used to determine in which index the value should be stored in, and that way we can easily find the value by checking if the index of the key for that value. However this can cause some problems if there is duplicates of the value give, or it gets the same key. If that happens a lot of times then hashing method is not that effective.

Chaining uses the same principle as the hashing, because it is a part of hashing. The difference is that using chaining method, it stores lists instead of single elements. In that way one index could store more then one value.

To check efficiency of hashing a load-factor is used. Load-factor can be calculated by taking number of elements and divide that by the size of the array (load-factor = elements/array.length).

Implementation

By using SingleLinkedList and Node from previous tasks, a new class is made called HashTableChaining. In this class a constructor is made which has one input where it defines the size of the array that should be used.

```
HashTableChaining(int s)
	{
		hashTableSize = s;
		hashTable = new SingleLinkedList[hashTableSize];
		for(int i=0;i<hashTableSize;i++)
			hashTable[i] = new SingleLinkedList();
	}
```

To get the key, which is used to decide in which index the value should be stored in another method is created called getKey(int key)

```
public int getKey(int key)
	{
		return key%hashTableSize;
	}
```

Inserting values is done by using insert(int a) method, it uses method from SingleLinkedList addInFront(int data) and adding lists in different indexes. The method looks like this:

```
public void insert(int a)
	{
		int key = getKey(a);
		hashTable[key].addInFront(a);
	}
```

And last to show the result of chaining a method called showHashTable is called

```
public void showHashTable(){//For å printe ut indeksene med listene.
		for(int i=0;i<hashTableSize;i++)
		{
```

```
                System.out.print("Index "+i+": ");
                System.out.print(hashTable[i].toString());
            }
        }
```

Inside main class user have two choices, 1 is by using an array of elements to test the chaining function by chaining the array.length. This way the same numbers are used all the time, and user can see the difference by changing the length of the array.

User can also use random numbers and specify how many numbers should be made. This can be done by removing the comment felts from

```
//System.out.print("Hvor mange elementer vil du generere? :");
//int amountOfElements = input.nextInt();
//Random randomNr = new Random();
//hT.insert(randomNr);
```

And adding the comments for for

```
hT.insert(elements[i]);
```

Testing

Two test were done both by using the same numbers from an array, only thing that changed between those two test are hashTable length.

1: by using length of 9

```
Hvor stor tabell ønsker du å lage?9
Index 0: (999)-->NULL
Index 1: (1324)-->NULL
Index 2: (2)-->NULL
Index 3: (876)-->(777)-->(66)-->(30)-->(1002)-->NULL
Index 4: (202)-->(4)-->(22)-->NULL
Index 5: (59)-->NULL
Index 6: NULL
Index 7: (7)-->(115)-->NULL
Index 8: (89)-->(44)-->(8)-->NULL
Den totale loadfactoren er: 1.8888888888888888
```

2: by using length of 17

```
Hvor stor tabell ønsker du å lage?17
Index 0: NULL
Index 1: NULL
Index 2: (2)-->NULL
Index 3: NULL
Index 4: (4)-->(89)-->NULL
Index 5: (22)-->NULL
Index 6: NULL
Index 7: (7)-->NULL
Index 8: (59)-->(8)-->NULL
Index 9: (876)-->NULL
Index 10: (44)-->NULL
Index 11: NULL
Index 12: (777)-->NULL
Index 13: (999)-->(30)-->(115)-->NULL
Index 14: NULL
Index 15: (202)-->(1324)-->(66)-->NULL
Index 16: (1002)-->NULL
Den totale loadfactoren er: 1.0
```

Conclusion

Chaining function works fine, it makes an array of the SingleLinkedList class and adds Lists to each index, the results is then printed on the screen and its easy to see where different values are stored. Im not sure if the key function can be improved without knowing which values is going to be stored beforehand.