|  |
| --- |
| Universitetet i Sørøst-Norge  Fakultet for Teknologi, Naturvitenskap og Maritime fag  Institutt for Mikrosystemer  –  Obligatorisk oppgave  Ola Slålien Thoen **26/04/2019**  **Oblig 3 vår DA-ALG1000** |
|  |
|  |
|  |
|  |

**Innholdsfortegnelse**

[1 Innledning 3](#_Toc8062918)

[1.1 Oppgaveteksten 3](#_Toc8062919)

[1.2 Løsning av oppgaven 3](#_Toc8062920)

[2 Klasser med constructor og metoder 4](#_Toc8062921)

[2.1 UML representasjon av klassene 4](#_Toc8062922)

[2.2 Node klassen 4](#_Toc8062923)

[2.2.1 Constructor med to parametre 4](#_Toc8062924)

[2.2.2 Constructor med én parameter 5](#_Toc8062925)

[2.3 SingleLinkedList klassen 5](#_Toc8062926)

[2.3.1 SingleLinkedList constructor 5](#_Toc8062927)

[2.3.2 addLast6 metoden 5](#_Toc8062928)

[2.3.3 showList12 metoden 5](#_Toc8062929)

[2.4 HashTable klassen 6](#_Toc8062930)

[2.4.1 Constructor HashTable 6](#_Toc8062931)

[2.4.2 hashToIndex metoden 6](#_Toc8062932)

[2.4.3 add metoden 6](#_Toc8062933)

[2.4.4 printTable 6](#_Toc8062934)

[2.5 Oblig3: Test-metode 7](#_Toc8062935)

[2.5.1 Main metoden 7](#_Toc8062936)

[3 Konklusjon 7](#_Toc8062937)

[4 Vedlegg 7](#_Toc8062938)

# Innledning

## Oppgaveteksten

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 der hashing 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 SingleLinkedList og Node som du brukte i 1. obligatoriske oppgave (1A og 1B). Rapport skal leveres med og denne skal inneholde eksempel på kjøring av programmet. Det er ikke nødvendig med GUI-løsning.

For å gjøre det enklere under uttesting kan tallene du tester med legges direkte inn i en tabell av tall du vil fordele over i de lenkede listene. Forslag: lag 2 metoder: en for å lese inn tallene og en som bare legger tall inn i en tabell. Da kan de byttes om etter behov.

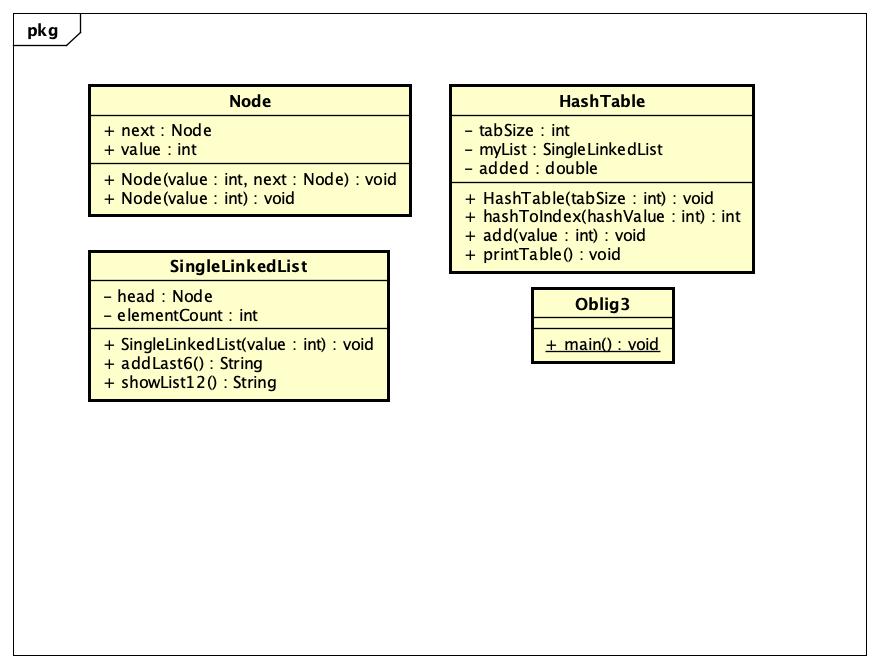
## Løsning av oppgaven

Denne oppgaven er blitt løst ved bruk av en array som består av SinleLinkedList-objekter. Hintet som ble gjort tilgjengelig på canvas ble brukt som en mal på oppgaven, enkelte deler er tatt vekk for de ble ikke brukt. Mens andre løsninger er beholdt. Klassene Node og SingleLinkedList fra oblig 1 er brukt i denne oppgaven, men det er blitt endret litt i SingleLinkedList-klassen for at den skulle fungere bedre i denne oppgaven. Det endelige produktet består av 4 klasser, inkludert Node og SingleLinkedList.

# Klasser med constructor og metoder

## UML representasjon av klassene

Det er bare lagt med de metodene i SingleLinkedList-klassen som blir brukt i denne oppgaven.



## Node klassen

Klasse fra oblig 1 som består av to constructor-er. Én global variabel «int value» og én «Node next».

### Constructor med to parametre

Constructor med «int value» og «Node next» som parametere setter de globale variablene som lokal i constructoren for å brukes som objekt.

### Constructor med én parameter

Constructor med «int value» som parameter og setter den globale variabelen «value» som lokal i constructoren, mens «next» som standard peker på «null».

## SingleLinkedList klassen

Klasse fra oblig 1 som består av én constructor og 15 metoder for bruk. Det blir bare benyttet én constructor og 2 metoder i denne oppgaven. Constructoren og metoden showList12 i denne klassen er forandret litt fra Oblig 1. Klassen inneholder én global variabel «int elementCount» for å telle antall «Noder» som er opprettet og én «Node head».

### SingleLinkedList constructor

Denne constructor-en er forandret litt for å kunne brukes lettere i denne oppgaven. Det er forandret fra at «head = null» til «head = new Node(value)» i constructoren for å gjøre det lett å legge til en integer-verdi.

### addLast6 metoden

Denne metoden legger til en ny «Node» med verdi til slutt i en eksisterende liste, eller lager en hvis den ikke finnes. Og denne metoden brukes for å legge til en ny verdi i den lenkede lista i tabellen.

### showList12 metoden

Det er forandret litt i denne metoden blant annet er det kommentert ut enkelte spesialtilfeller for å ikke få tilbakemeldinger som ikke er passende for denne oppgaven. Det er også fjernet tekst-tilbakemelding i return-statementet, slik at man bare får nodene skrevet ut uten en String.

## HashTable klassen

Denne klassen består av én constructor og 3 metoder for bruk. Klassen inneholder de globale variablene «int tabSize» og «double added» for å styre størrelsen på tabellen og antall elementer som er lagt til i tabellen, og en array med SingleLinkedList-objekter med navnet «myList».

### Constructor HashTable

Denne constructor-en setter den globale variabelen «int tabSize» og array-en «myList» lokal. Constructoren bruker «int tabSize» som parameter for å sette array-en «myList» til en størrelse som er «tabSize» stor.

### hashToIndex metoden

Denne metoden brukes i add-metoden og den hash-er verdien «int hashValue» som står som parameter, og returnerer index-en som vardien skal få. Verdien er hash-et ved å ta «hashValue»-verdien modulo tabelstørrelsen «tabSize», den verdien som kommer da er index-en verdien skal ha.

### add metoden

Denne metoden har «int value» som parameter som kommer fra brukeren. Den bruker metoden «hashToIndex» for å finne index-en verdien skal få. Deretter sjekkes det i en if-test om den index-en til verdien er ledig. Hvis den er det legges det bare til en «Node» i den lenkede lista, hvis ikke opprettes det en ny lenket liste på index-en og deretter legges det til en «Node» med verdien. Tilslutt øker variabelen «added» med en for å vise at det er lagt til en verdi.

### printTable

Metoden består av en for-løkke som først skriver ut hvilken index som er gjeldende. Hvis den gjeldende index-en inneholder en lenket liste vil alle nodene bli skrevet ut ved hjelp av metoden showList12 fra SingleLinkedList- klassen. I tillegg adderes det med 1 i variabelen «keys», som holder tellinga for antall index-er som er opptatt. Derimot hvis index-en er ledig skrives det ut en melding som sier «index is empty». Til slutt regnes «load factor» ut ved å dele antall «Noder» som er lagt til på antall index-er som er brukt. Og dette tallet skrives ut til bruker.

## Oblig3: Test-metode

Denne klassen inneholder bare en main-metode for testing av programmet.

### Main metoden

Her blir bruker først spurt om hvor lang det er ønskelig å ha lista som skal lages. Deretter opprettes det et objekt av typer «HashTable» med størrelsen som er gitt av bruker. Deretter blir det spurt om verdier som skal legges inn. Det kan legges inn verdier frem til det legges inn verdien 0 som avslutter programmet. Helt til slutt benyttes printTable-metoden for å skrive ut hele tabellen.

# Konklusjon

Programmet gjør som oppgaven spør om hvor det spør bruker om størrelsen som er ønskelig og hvilke verdier som vil legges inn. For å gjøre det enklere når programmet er testet er det lagt inn 15 forskjellige verdier som er kommentert ut, men disse kan legges til igjen. Da er det bare å skrive inn 0 når det blir spurt om hvilke verdier som vil legges til så brukes de 15 verdiene i lista.

# Vedlegg

Kildekoden er lagt ved i den leverte .zip-fila.