

Kan sjekke om hashtable inneholder et spesifikt navn. Testet med mitt eget navn (Alper Yarenbasi) samt en person som ikke går i klassen (Elper Yarenbasi)

```
Index 22: null
Index 23: Emil Ruud -> null
Index 24: null
Index 25: null
Index 26: null
Index 27: null
Index 28: Jon Martinus Rødtang -> null
Index 29: Elise Christine Gjestad -> Henrik Sund -> null
Index 30: null
Index 31: null
Index 32: Charlotte Adele Heidenstrøm Corapi -> null
Index 33: Markus Evald Dalbakk -> null
Index 34: Gard Alhaug -> Callum Gran -> null
Index 35: null
Index 36: Marcus Westum -> null
Index 37: null
Index 38: null
Index 39: Håkon Sørli -> null
Index 40: null
Index 41: null
Index 42: null
Index 43: Pedro Pablo Cardona Arroyave -> Anders Nikolai Holsten -> Aryan Malekian -> null
Index 44: null
Index 45: null
```

Mulighet til å printe ut hashtable også (brukt for manuell debugging og for å se visuelt om programmet fungerer som det skal).

Krav for godkjenning del 1:

- ✓ Programmet leser fila med navn, og klarer å legge alle i hashtabellen.
- ✓ Kollisjoner håndteres med lenke lister.
- ✓ Programmet skriver ut alle kollisjoner, og lastfaktoren til slutt
- ✓ Programmet klarer å slå opp personer i faget ved hjelp av hashtabellen. (Bruk gjerne oppslag på eget navn.)
- ✓ Programmet bruker «navn.txt», ikke «mappe/navn.txt». Retting tar for lang tid, om jeg må håndtere hundrevis av mapper. Så det skjer ikke.
- ✓ Legg ved utskrift fra kjøringen.
- ✓ Antall kollisjoner pr. person er under 0,4 (men ikke eksakt 0, regn med desimaltall...)

Deloppgave 2: Hashtabeller med heltallsnøkler – og ytelse

Filer tilknyttet opg2: doubleHash.java, Main.java

```
Custom HashTable: Time taken = 554 ms  
Load factor = 0.7547171520114306  
Total collisions = 3780664  
-----  
Java HashMap: Time taken = 1599 ms
```

Veldig overasket selv, men egen Hash er 3x kjappere enn java sin. Implementert doublehash fra boka/forelesning samt tipset om at h2 ikke burde regnes med mindre h1 gir kollisjon. Ga en forbedring i ytselen drastisk.

Krav til godkjenning av del 2:

- ✓ Et program som legger 10 millioner tilfeldige tall inn i en hashtabell, og måler tiden. (Python/javascript kan bruke en million tall, hvis 10 mill går tregt.)
- ✓ Kollisjoner håndteres med dobbel hashing
- ✓ Sammenligning med tid for å sette tallene inn i Hashmap. (Det er ikke nødvendig å «slå java», men det er mulig!)
- ✓ Utskrift som viser tidsmålingene, lastfaktor og antall kollisjoner.