University of Southern Denmark • Syddansk Universitet

DM502

Forelæsning 9

Indhold

- Datatypen Mapping (Dictionary)
  - HashMap
- 1. delprojekt

Motivation

Motivation

- Opslag
  - Telefonbog
  - Ordbog
  - Kataloger (fx bibliotekskataloger)
  - Osv...

Motivation

- Opslag
  - Telefonbog
  - Ordbog
  - Kataloger (fx bibliotekskataloger)
  - Osv...
- Vil gerne kunne...

Motivation

- Opslag
  - Telefonbog
  - Ordbog
  - Kataloger (fx bibliotekskataloger)
  - Osv...
- Vil gerne kunne...
  - Opslag - findes person x i telefonbogen og i givet fald hvad er vedkommendes nummer

Motivation

- Opslag
  - Telefonbog
  - Ordbog
  - Kataloger (fx bibliotekskataloger)
  - Osv...
- Vil gerne kunne...
  - Opslag - findes person x i telefonbogen og i givet fald hvad er vedkommendes nummer
  - Indsætte - tilføje en person med telefonnummer til telefonbogen

Motivation

- Opslag
  - Telefonbog
  - Ordbog
  - Kataloger (fx bibliotekskataloger)
  - Osv...
- Vil gerne kunne...
  - Opslag - findes person x i telefonbogen og i givet fald hvad er vedkommendes nummer
  - Indsætte - tilføje en person med telefonnummer til telefonbogen
  - Slette - fjerne en person fra telefonbogen

Motivation

Motivation

- "Hvorfor ikke bruge en ArrayList?"
  - Opslag - løb listen igennem for at finde en person med tilhørende telefonnummer
  - Indsætte - indsæt i listen
  - Slette - løb listen igennem og find personen der skal slettes og fjern elementet fra listen

Motivation

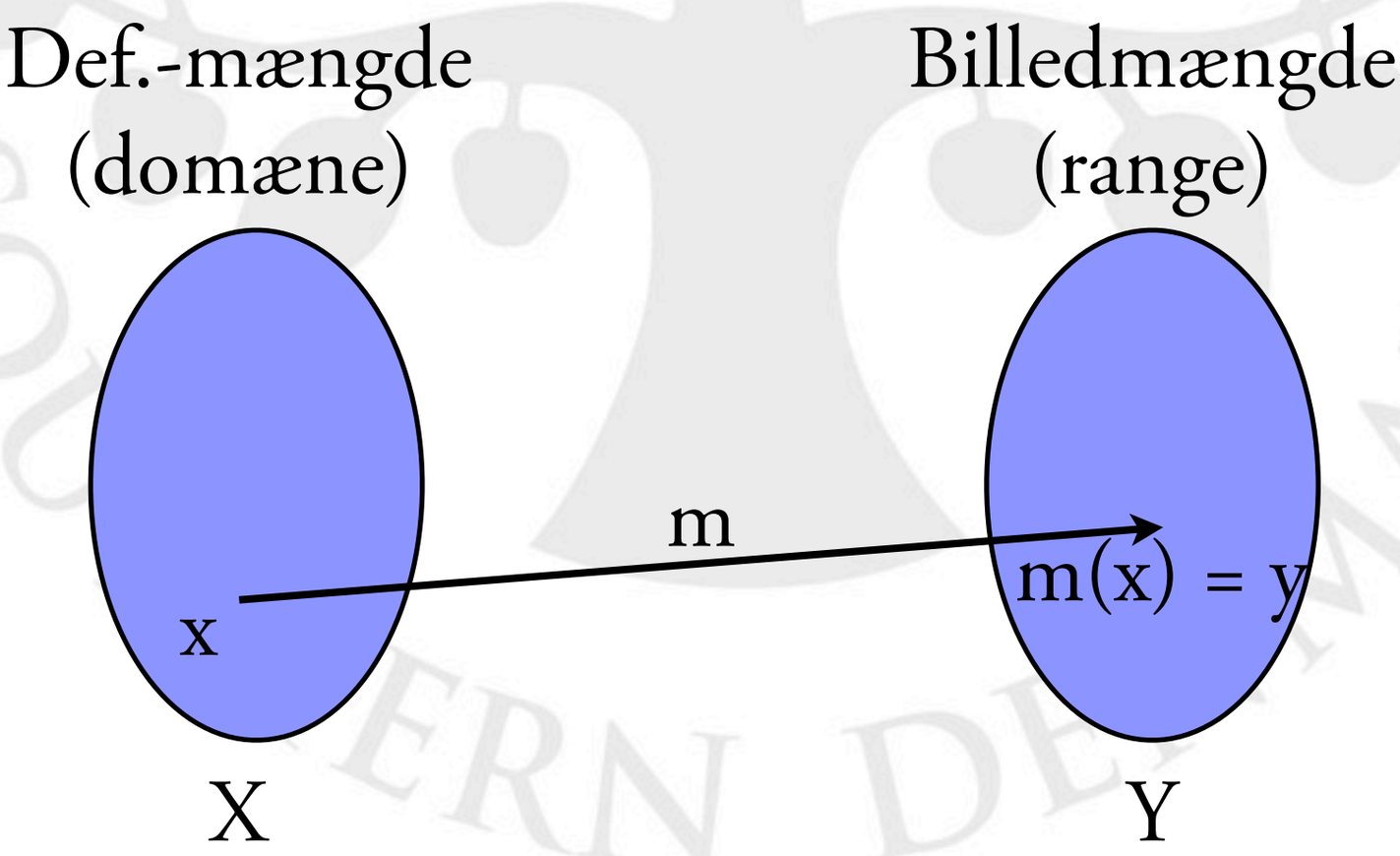
- "Hvorfor ikke bruge en ArrayList?"
  - Opslag - løb listen igennem for at finde en person med tilhørende telefonnummer
  - Indsætte - indsæt i listen
  - Slette - løb listen igennem og find personen der skal slettes og fjern elementet fra listen
- "Fordi...
  - Det tager for lang tid
  - En telefonbog med 5 millioner numre

Mapping

- Løsning
  - En mapping - en afbildning fra et domæne (fx personer) til en billedmænge (fx telefonnumre)
  - Lidt ligesom funktioner, men ikke helt...
  - Kan gøres langt mere effektivt end en liste

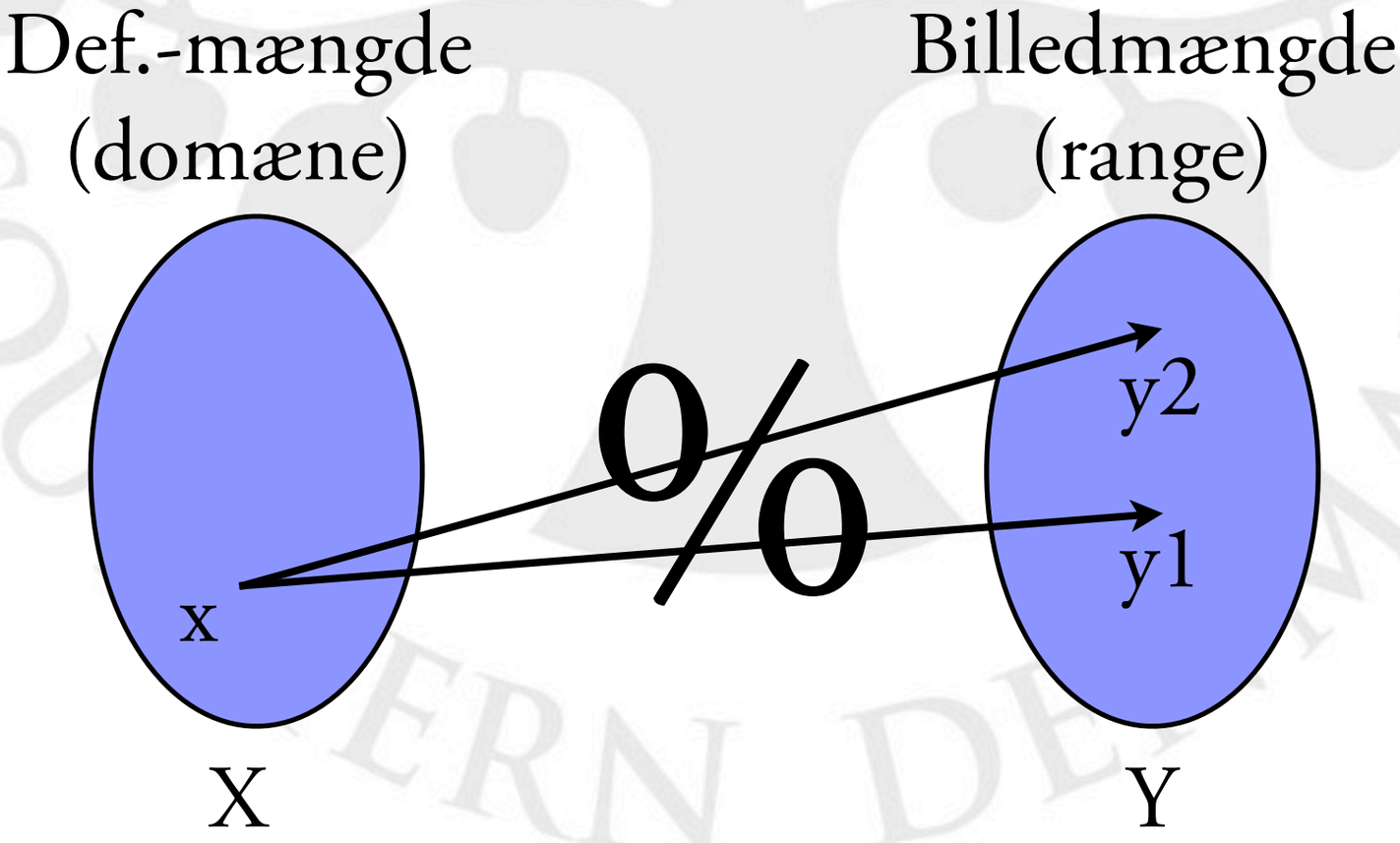
Mappings

- Definitionsmængde (domain)
  - Et element kaldes en nøgle (key)
- Billedmængde (range)
  - Et element kaldes en værdi (value)



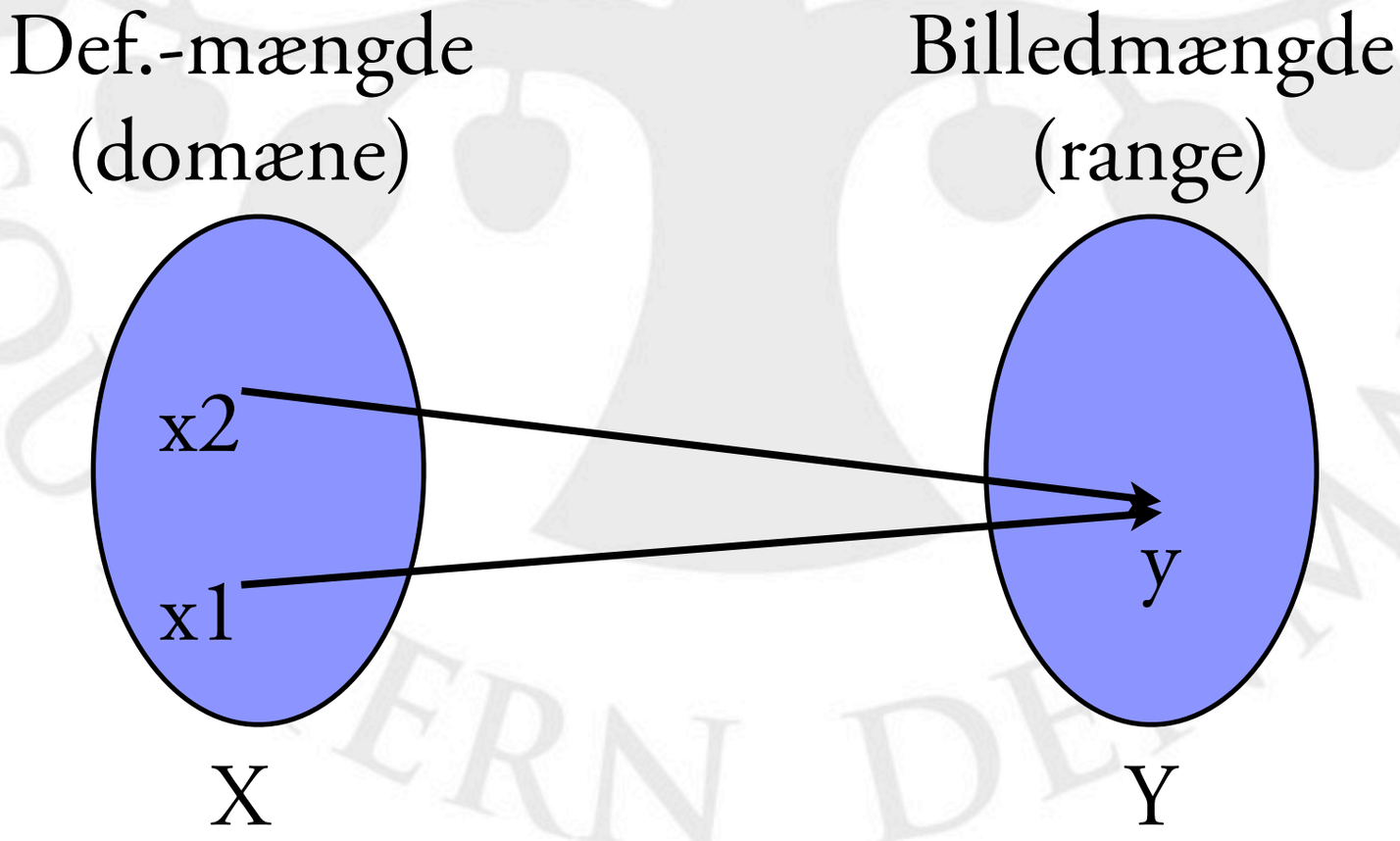
Mappings

- Definitionsmængde (domain)
  - Et element kaldes en nøgle (key)
- Billedmængde (range)
  - Et element kaldes en værdi (value)
- $\forall x \in X \; \exists! y \in Y : m(x) = y$



Mappings

- Definitionsmængde (domain)
  - Et element kaldes en nøgle (key)
- Billedmængde (range)
  - Et element kaldes en værdi (value)
- $\forall x \in X \; \exists! y \in Y : m(x) = y$
  - Men $x_1 \neq x_2 : m(x_1) = y = m(x_2)$ er OK



Mappings

- Bemærk - man kan ikke gå baglæns i en mapping
  - Givet et element y i billedmængden, er ingen mulighed for at finde de(n) nøgle(r) der mapper til y
  - (Kan klares ved et at løbe alle elementer i definitionsmængden igennem,
    men det kan tage meget lang tid)

HashMap

HashMap

- Mappings i Java - `java.util.HashMap`

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`
    - Telefonbog

```
HashMap<String,Integer> m = new
HashMap<String,Integer>();
```

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`
    - Telefonbog

```
HashMap<String,Integer> m = new
HashMap<String,Integer>();
```

- Indsæt i mapping

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`
    - Telefonbog

```
HashMap<String,Integer> m = new
HashMap<String,Integer>();
```

- Indsæt i mapping
  - Typen af den indsatte nøgle og værdi skal passe med det der blev angivet da HashMap'et blev oprettet

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`
    - Telefonbog

```
HashMap<String,Integer> m = new
HashMap<String,Integer>();
```

- Indsæt i mapping
  - Typen af den indsatte nøgle og værdi skal passe med det der blev angivet da HashMap'et blev oprettet
  - Overskriver hvis nøglen allerede findes

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`
    - Telefonbog

```
HashMap<String,Integer> m = new
HashMap<String,Integer>();
```

- Indsæt i mapping
  - Typen af den indsatte nøgle og værdi skal passe med det der blev angivet da HashMap'et blev oprettet
  - Overskriver hvis nøglen allerede findes
  - `m.put( `*`nøgle`*`, `*`værdi`*` );`

HashMap

- Mappings i Java - `java.util.HashMap`
  - Virker kun med objekter (ikke fundamentale typer)
- Ny mapping
  - Skal angive typen for nøglerne og typerne
  - `HashMap<`*`Nøgletype`*`,`*`Værditype`*`> m = new HashMap<`*`Nøgletype`*`,`*`Værditype`*`>();`
    - Telefonbog

```
HashMap<String,Integer> m = new
HashMap<String,Integer>();
```

- Indsæt i mapping
  - Typen af den indsatte nøgle og værdi skal passe med det der blev angivet da HashMap'et blev oprettet
  - Overskriver hvis nøglen allerede findes
  - `m.put(` *`nøgle`*`, `*`værdi`*` );`
    - `m.put( "Homer Simpson", 5556528 );`

HashMap

HashMap

- Opslag

HashMap

- Opslag
  - Slår værdien op for en given nøgle

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( nøgle );`

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( `*`nøgle`*` );`
    - `m.get( "Homer Simpson" );`

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( `*`nøgle`*` );`
    - `m.get( "Homer Simpson" );`
- Slette

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( `*`nøgle`*` );`
    - `m.get( "Homer Simpson" );`
- Slette
  - Fjerner en nøgle fra mapping'en

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( nøgle );`
    - `m.get( "Homer Simpson" );`
- Slette
  - Fjerner en nøgle fra mapping'en
  - Den mappede værdi fjernes kun hvis samtlige nøgler der mapper til værdien er fjernet

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( nøgle );`
    - `m.get( "Homer Simpson" );`
- Slette
  - Fjerner en nøgle fra mapping'en
  - Den mappede værdi fjernes kun hvis samtlige nøgler der mapper til værdien er fjernet
  - `m.remove( nøgle );`

HashMap

- Opslag
  - Slår værdien op for en given nøgle
  - Returnerer `null` hvis nøglen ikke findes
  - `m.get( nøgle );`
    - `m.get( "Homer Simpson" );`
- Slette
  - Fjerner en nøgle fra mapping'en
  - Den mappede værdi fjernes kun hvis samtlige nøgler der mapper til værdien er fjernet
  - `m.remove( nøgle );`
    - `m.remove( "Homer Simpson" );`

HashMap

HashMap

- Opslag i nøglerne

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en
  - `m.containsKey( nøgle );`

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en
  - `m.containsKey( nøgle );`
    - `m.containsKey( "Homer Simpson" );`

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en
  - `m.containsKey( nøgle );`
    - `m.containsKey( "Homer Simpson" );`
- Opslag i værdierne

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en
  - `m.containsKey( nøgle );`
    - `m.containsKey( "Homer Simpson" );`
- Opslag i værdierne
  - Tjek om en given værdi findes i mapping'en

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en
  - `m.containsKey( nøgle );`
    - `m.containsKey( "Homer Simpson" );`
- Opslag i værdierne
  - Tjek om en given værdi findes i mapping'en
  - `m.containsValue( værdi );`

HashMap

- Opslag i nøglerne
  - Tjek om en given nøgle findes i mapping'en
  - `m.containsKey( nøgle );`
    - `m.containsKey( "Homer Simpson" );`
- Opslag i værdierne
  - Tjek om en given værdi findes i mapping'en
  - `m.containsValue( værdi );`
    - `m.containsValue( 5556528 );`

HashMap

HashMap

- List alle nøgler

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)
  - ```
    for( String key : m.keySet() ) {
        System.out.println(key + " -> " + m.get(key));
    }
    ```
    ```

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)
  - ```
    for( String key : m.keySet() ) {
        System.out.println(key + " -> " + m.get(key));
    }
    ```
  - Bemærk at hver nøgle kun findes én gang, derfor

    `m.key`**`Set`**`()`
    ```

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)
  - ```
    for( String key : m.keySet() ) {
        System.out.println(key + " -> " + m.get(key));
    }
    ```
  - Bemærk at hver nøgle kun findes én gang, derfor `m.keySet()`
- List alle værdier
    ```

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)
  - ```
    for( String key : m.keySet() ) {
        System.out.println(key + " -> " + m.get(key));
    }
    ```
  - Bemærk at hver nøgle kun findes én gang, derfor `m.keySet()`
- List alle værdier
  - Gennemløb hele billedmængden (range)
    ```

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)
  - ```
    for( String key : m.keySet() ) {
        System.out.println(key + " -> " + m.get(key));
    }
    ```
  - Bemærk at hver nøgle kun findes én gang, derfor `m.key`**`Set`**`()`
- List alle værdier
  - Gennemløb hele billedmængden (range)
  - ```
    for( Integer value : m.values() ) {
        System.out.println( value );
    }
    ```

HashMap

- List alle nøgler
  - Gennemløb hele definitionsmængden (domain)
  - ```
    for( String key : m.keySet() ) {
        System.out.println(key + " -> " + m.get(key));
    }
    ```
  - Bemærk at hver nøgle kun findes én gang, derfor `m.keySet()`
- List alle værdier
  - Gennemløb hele billedmængden (range)
  - ```
    for( Integer value : m.values() ) {
        System.out.println( value );
    }
    ```
  - Bemærk at hver værdi kan optræde flere gange (for forskellige nøgler)

Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();
```

Eksempel

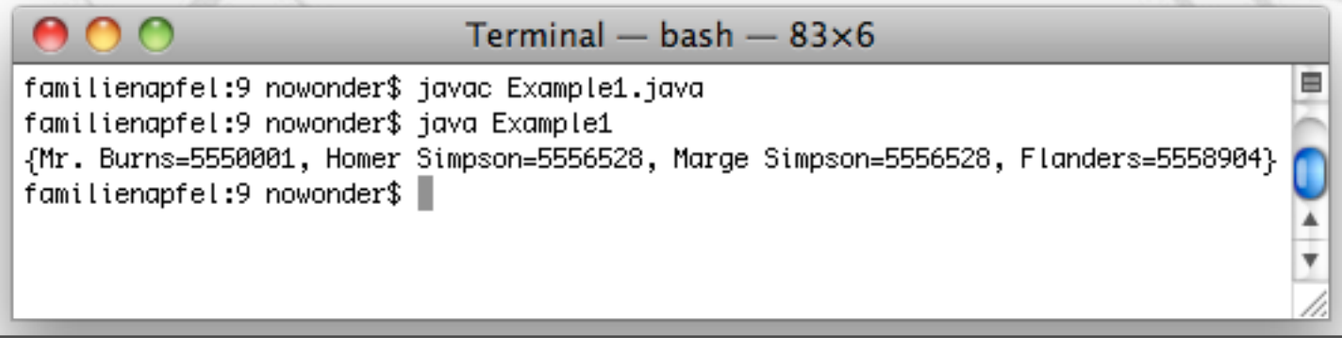
```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );
```

Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );
```

Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );
```

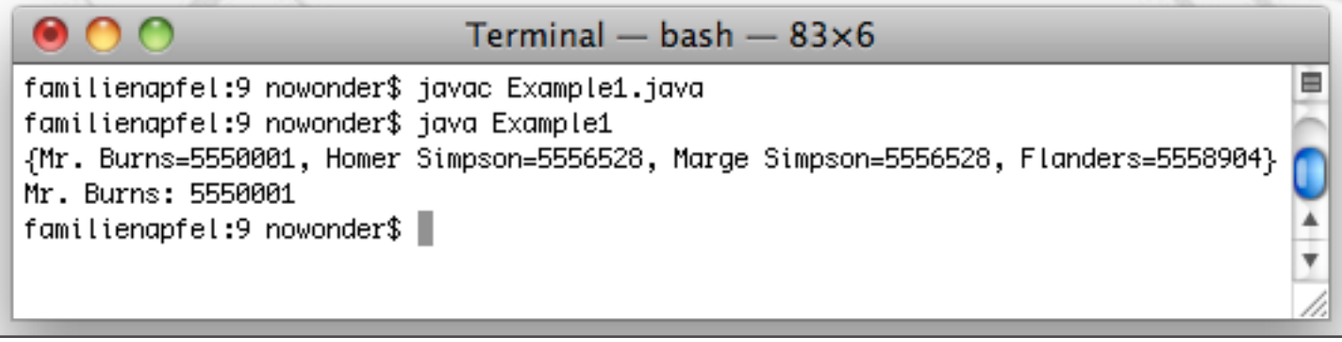


Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );

        int number = m.get( "Mr. Burns" );
        System.out.println( "Mr. Burns: " + number );
```

Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );

        int number = m.get( "Mr. Burns" );
        System.out.println( "Mr. Burns: " + number );
```

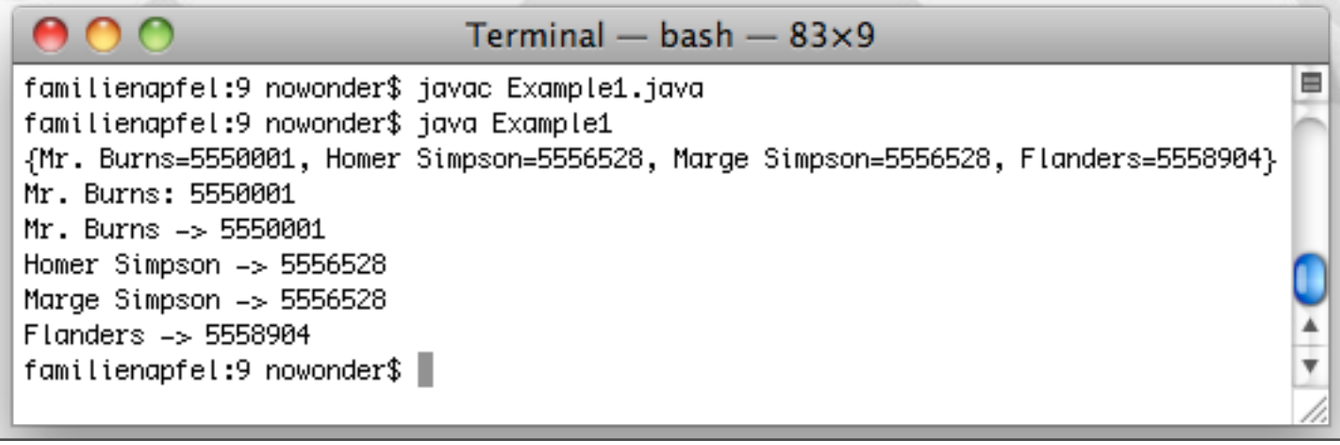


Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );

        int number = m.get( "Mr. Burns" );
        System.out.println( "Mr. Burns: " + number );

        for( String key : m.keySet() ) {
            System.out.println( key + " -> " + m.get(key) );
        }
```

Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );

        int number = m.get( "Mr. Burns" );
        System.out.println( "Mr. Burns: " + number );

        for( String key : m.keySet() ) {
            System.out.println( key + " -> " + m.get(key) );
        }
```

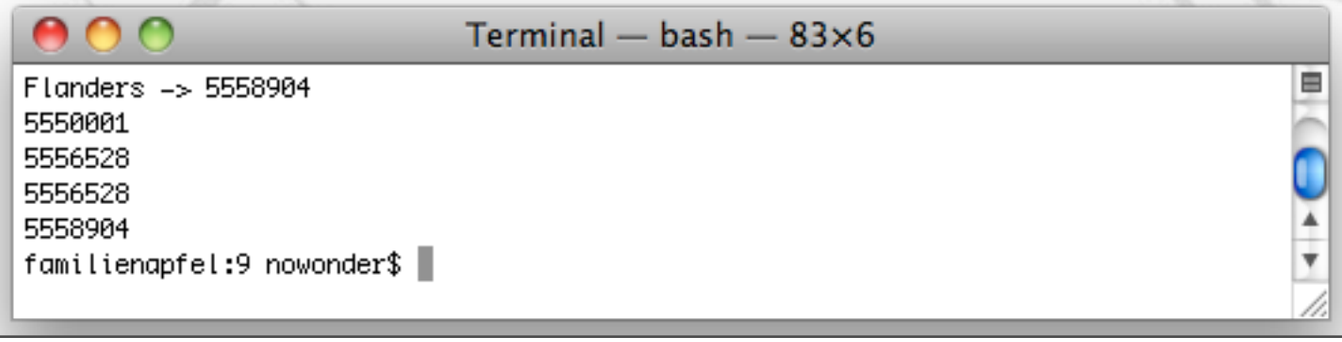


Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );

        int number = m.get( "Mr. Burns" );
        System.out.println( "Mr. Burns: " + number );

        for( String key : m.keySet() ) {
            System.out.println( key + " -> " + m.get(key) );
        }

        for( int value : m.values() ) {
            System.out.println( value );
        }
```

Eksempel

```
import java.util.HashMap;

public class Example1 {
    public static void main( String[] args ) {
        HashMap<String,Integer> m = new HashMap<String,Integer>();

        m.put( "Homer Simpson", 5556528 );
        m.put( "Mr. Burns", 5550001 );
        m.put( "Marge Simpson", 5556528 );
        m.put( "Flanders", 5558904 );

        System.out.println( m );

        int number = m.get( "Mr. Burns" );
        System.out.println( "Mr. Burns: " + number );

        for( String key : m.keySet() ) {
            System.out.println( key + " -> " + m.get(key) );
        }

        for( int value : m.values() ) {
            System.out.println( value );
        }
```



Eksempel - fortsat...

```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}
```

Eksempel - fortsat...

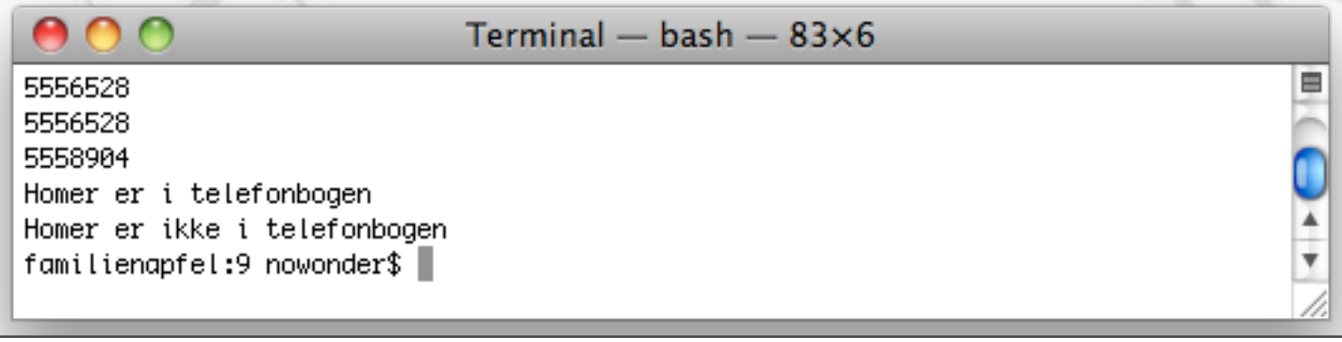
```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );
```

Eksempel - fortsat...

```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );

if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
} else {
    System.out.println( "Homer er ikke i telefonbogen" );
}
```

Eksempel - fortsat...

```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );

if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
} else {
    System.out.println( "Homer er ikke i telefonbogen" );
}
```



Eksempel - fortsat...

```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );

if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
} else {
    System.out.println( "Homer er ikke i telefonbogen" );
}

if( m.containsValue( 5556528 ) ) {
    System.out.println( "Nummeret 555-6528 er i telefonbogen" );
}
```

Eksempel - fortsat...

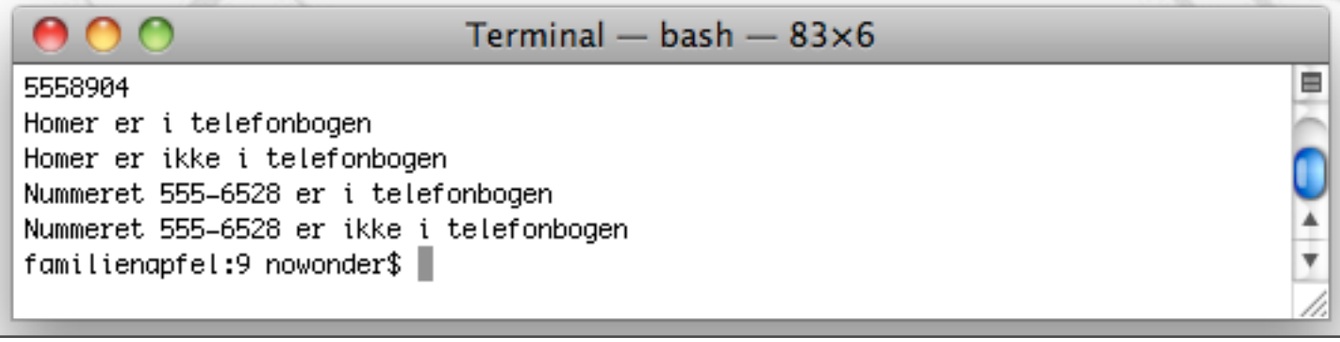
```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );

if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
} else {
    System.out.println( "Homer er ikke i telefonbogen" );
}

if( m.containsValue( 5556528 ) ) {
    System.out.println( "Nummeret 555-6528 er i telefonbogen" );
}

m.remove( "Marge Simpson" );
```

Eksempel - fortsat...

```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );

if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
} else {
    System.out.println( "Homer er ikke i telefonbogen" );
}

if( m.containsValue( 5556528 ) ) {
    System.out.println( "Nummeret 555-6528 er i telefonbogen" );
}

m.remove( "Marge Simpson" );

if( m.containsValue( 5556528 ) ) {
    System.out.println( "Nummeret 555-6528 er i telefonbogen" );
} else {
    System.out.println( "Nummeret 555-6528 er ikke i telefonbogen" );
}
    }
}
```

Eksempel - fortsat...

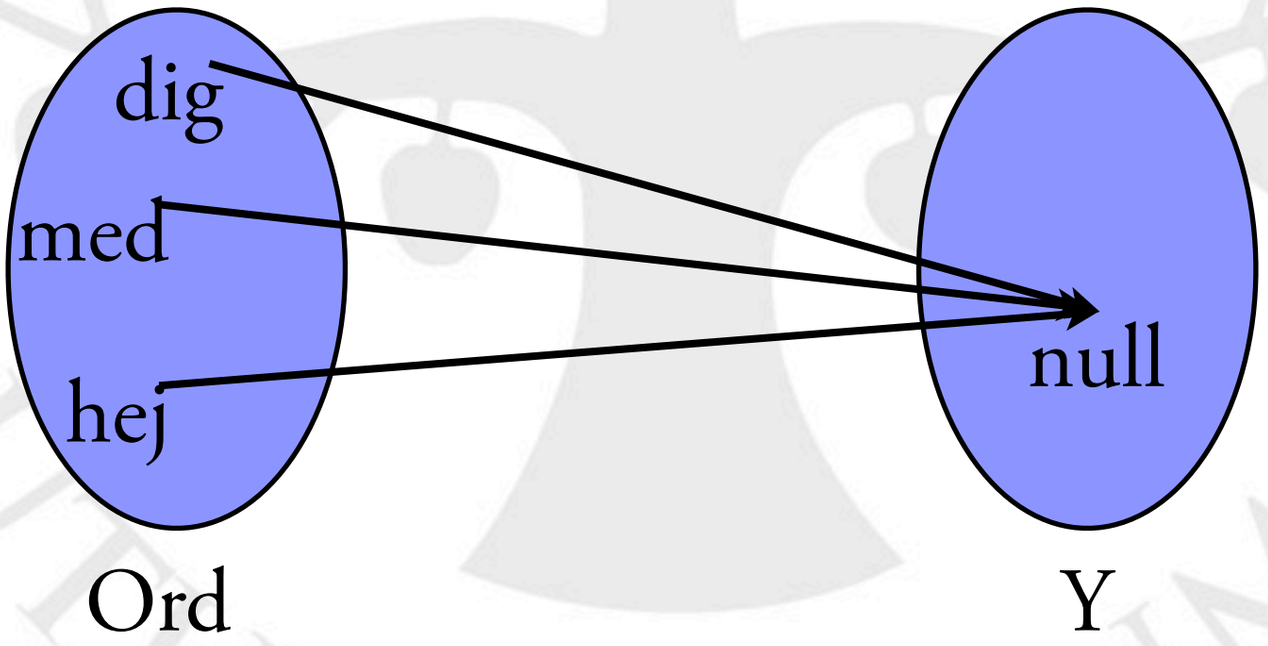
```
if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
}

m.remove( "Homer Simpson" );

if( m.containsKey( "Homer Simpson" ) ) {
    System.out.println( "Homer er i telefonbogen" );
} else {
    System.out.println( "Homer er ikke i telefonbogen" );
}

if( m.containsValue( 5556528 ) ) {
    System.out.println( "Nummeret 555-6528 er i telefonbogen" );
}

m.remove( "Marge Simpson" );

if( m.containsValue( 5556528 ) ) {
    System.out.println( "Nummeret 555-6528 er i telefonbogen" );
} else {
    System.out.println( "Nummeret 555-6528 er ikke i telefonbogen" );
}
    }
}
```



Autoboxing action

- Bemærk den udvidede anvendelse af autoboxing
  - `HashMap<String,Integer> m = new HashMap<String,Integer>();`
  - `m.put( "Homer Simpson", 5556528 );`
  - Java forventer at det andet argument til `put` er af typen `Integer`, men vi giver den en `int` (`5556528`)
  - Den pakker selv værdien ind i en `Integer` (autoboxing)
  - Identificer selv flere tilfælde (der er mange) hvor autoboxing anvendes i ovenstående eksempel

Autoboxing & Iterable

- ```
  for( int value : m.values() ) {
      System.out.println( value );
  }
  ```
  - `m.values()` returner en `Collection` over værdierne i HashMap'et.
  - Collection implementerer interfacet `Iterable`
    - Så den udvidede for-løkke er altså OK
  - Elementerne i den returnerede Collection er af typen `Integer` (sådan specificerede vi jo HashMap'et)
  - Vha. autoboxing kan vi trække hver `Integer` ud og de bliver automatisk oversat til en `int`.
  - Overvej hvert skridt i ovenstående kompakte kode
  ```

Smart anvendelse

- En mapping kan bruges til genkendelses-opslag
  - Antag vi vil tælle antallet af forskellige ord i en tekst
  - Kan klares med en mapping
    - Mapper hvert læst ord til...?!?
    - Ingenting (et element)



Smart anvendelse

- Antallet af forskellige ord i teksten kun nu findes som:
  - `m.size();`
- Hvert ord kan findes ved at gennemløbe alle nøgler (ord):
  - ```
    for( String key : m.keySet() ) {
        System.out.println( key );
    }
    ```
    ```

1. delprojekt

1. delprojekt

- Mulige bedømmelser
  - Godkendt
    - Super! - vær glad :-)
    - Kig på kommentarene alligvel
  - Genaflevering
    - Det var næsten godt nok, men ikke helt :-|
    - Du skal rette de fejl og mangler der bliver gjort opmærksom på i rettelserne
    - Genaflever inden for tidsfristen
  - Ikke godkendt
    - Det var ikke godt nok - øv :-(
    - Du er desværre dumpet kurset...

Forslag til løsning

- Mit forslag til en delvis løsning
- Der er (næsten) ingen fejlhåndtering
  - Dvs. det er ikke helt godt nok :-)

TOP SECRET

Information

- Husk
  - Ingen forelæsninger på torsdag
  - Sidste forelæsninger i næste uge,
  - hvor den anden (og sidste) projektopgave udleveres