



DM502

Forelæsning 7



Indhold

- Information
- Mere klasser og objekter - med fokus på keywords
 - Genudsendelse
 - Public og private
 - Static variable og metoder



Information



Information

- Husk at kigge på opgaverne inden timerne
 - Det er ikke muligt at nå at lave det hele til øvelserne
 - Man får kun noget ud af opgaverne til eksaminatorietimerne, hvis man har kigget på dem i forvejen



Information

- Husk at kigge på opgaverne inden timerne
 - Det er ikke muligt at nå at lave det hele til øvelserne
 - Man får kun noget ud af opgaverne til eksaminatorietimerne, hvis man har kigget på dem i forvejen
- Husk bogen og noter!



Information

- Husk at kigge på opgaverne inden timerne
 - Det er ikke muligt at nå at lave det hele til øvelserne
 - Man får kun noget ud af opgaverne til eksaminatorietimerne, hvis man har kigget på dem i forvejen
- Husk bogen og noter!
- Bogens afsnit om interfaces er mildest talt dårligt!



Information

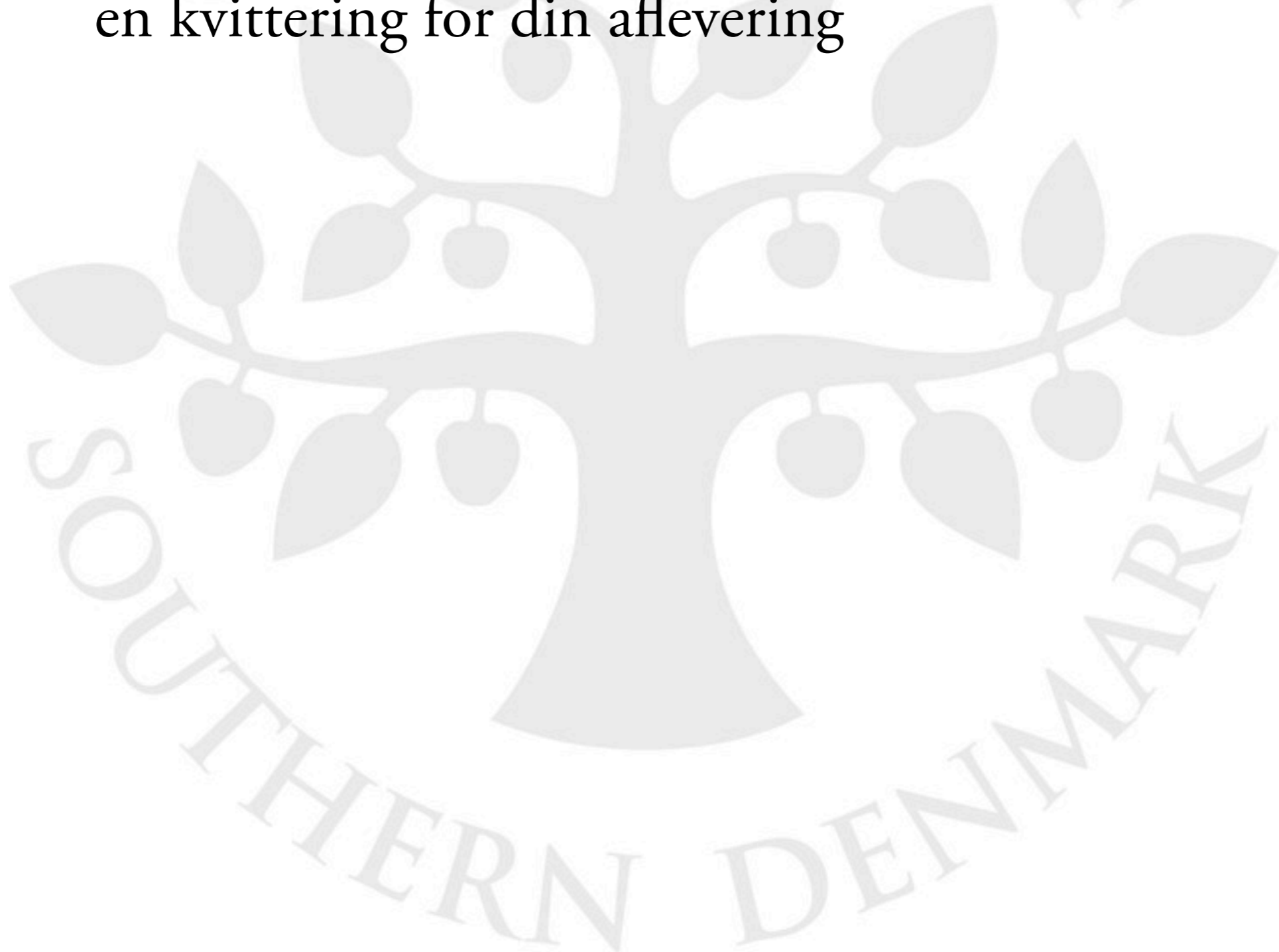
- Husk at kigge på opgaverne inden timerne
 - Det er ikke muligt at nå at lave det hele til øvelserne
 - Man får kun noget ud af opgaverne til eksaminatorietimerne, hvis man har kigget på dem i forvejen
- Husk bogen og noter!
- Bogens afsnit om interfaces er mildest talt dårligt!
- Sidste uge med to forelæsninger

Information



Information

- Kvittering for aflevering af obligatorisk opgave
 - Du har mulighed for (men det er ikke et krav) at få en kvittering for din aflevering



Information

- Kvittering for aflevering af obligatorisk opgave
 - Du har mulighed for (men det er ikke et krav) at få en kvittering for din aflevering
- Procedure (alt skal foretages indenfor tidsfristen)
 - Aflevering via Blackboard
 - Klik på DM502 under blackboard
 - Klik på "Tools" i menuen



Information

- Kvittering for aflevering af obligatorisk opgave
 - Du har mulighed for (men det er ikke et krav) at få en kvittering for din aflevering
- Procedure (alt skal foretages indenfor tidsfristen)
 - Aflevering via Blackboard
 - Klik på DM502 under blackboard
 - Klik på "Tools" i menuen
 - Vælg "Assignment hand in"

Information

- Kvittering for aflevering af obligatorisk opgave
 - Du har mulighed for (men det er ikke et krav) at få en kvittering for din aflevering
- Procedure (alt skal foretages indenfor tidsfristen)
 - Aflevering via Blackboard
 - Klik på DM502 under blackboard
 - Klik på "Tools" i menuen
 - Vælg "Assignment hand in"
 - Udfyld formularen og afslut med submit

Information

- Kvittering for aflevering af obligatorisk opgave
 - Du har mulighed for (men det er ikke et krav) at få en kvittering for din aflevering
- Procedure (alt skal foretages indenfor tidsfristen)
 - Aflevering via Blackboard
 - Klik på DM502 under blackboard
 - Klik på "Tools" i menuen
 - Vælg "Assignment hand in"
 - Udfyld formularen og afslut med submit
 - Udskriv kvitteringen

Genudsendelse



Genudsendelse

- Klasse
 - Abstrakt beskrivelse
 - Tomme variable
 - Metoder
 - `public class Car { ... }`



Genudsendelse

- Klasse
 - Abstrakt beskrivelse
 - Tomme variable
 - Metoder
 - `public class Car { ... }`
- Objekt
 - Konkret instans af en klasse
 - Variable har værdier
 - `Car audi = new Car("Audi A4", "rød");`

Car.java

```
public class Car {
    private String model;
    private int year;
    private String color;
    private int mileage;

    public Car( String carModel, String carColor ) {
        model = carModel;
        year = 2010;
        color = carColor;
        mileage = 0;
    }

    public int getMileage() {
        return mileage;
    }

    public void drive( int distance ) {
        mileage = mileage + distance;
    }

    public void setColor( String newColor ) {
        color = newColor;
    }
}
```

CarMaker.java

```
public class CarMaker {  
    public static void main( String[] args ) {  
        Car audi;  
        Car skoda;  
        int mileage;  
  
        audi = new Car( "Audi A4", "sølvgrå" );  
        skoda = new Car( "Skoda Fabia", "rød" );  
  
        mileage = audi.getMileage();  
        System.out.println( "Audi: " + mileage );  
  
        audi.drive( 30 );  
        mileage = audi.getMileage();  
        System.out.println( "Audi: " + mileage );  
  
        audi.drive( 15 );  
        mileage = audi.getMileage();  
        System.out.println( "Audi: " + mileage );  
    }  
}
```



Public og private



Public og private

- Lad os tage et tættere kig på klassen Car



Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`



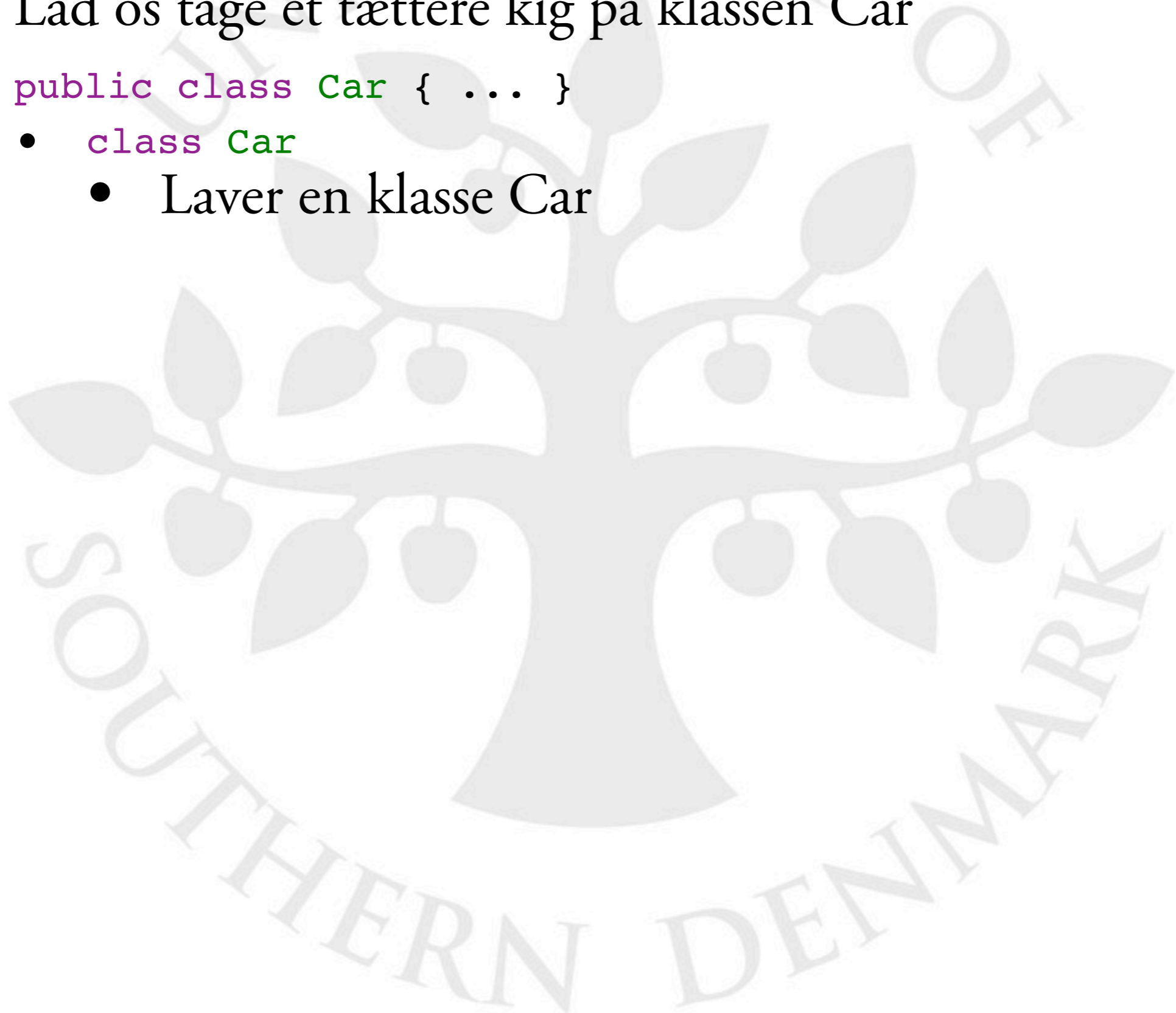
Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`



Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`
 - Laver en klasse Car



Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`
 - Laver en klasse Car
 - Skal ligge i Car.java



Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`
 - Laver en klasse Car
 - Skal ligge i Car.java
 - `public`



Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`
 - Laver en klasse Car
 - Skal ligge i Car.java
 - `public`
 - En klasse som er offentligt tilgængelig for andre



Public og private

- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`
 - Laver en klasse Car
 - Skal ligge i Car.java
 - `public`
 - En klasse som er offentligt tilgængelig for andre
 - Vi kan bruge den i vores carMaker-klasse

Public og private

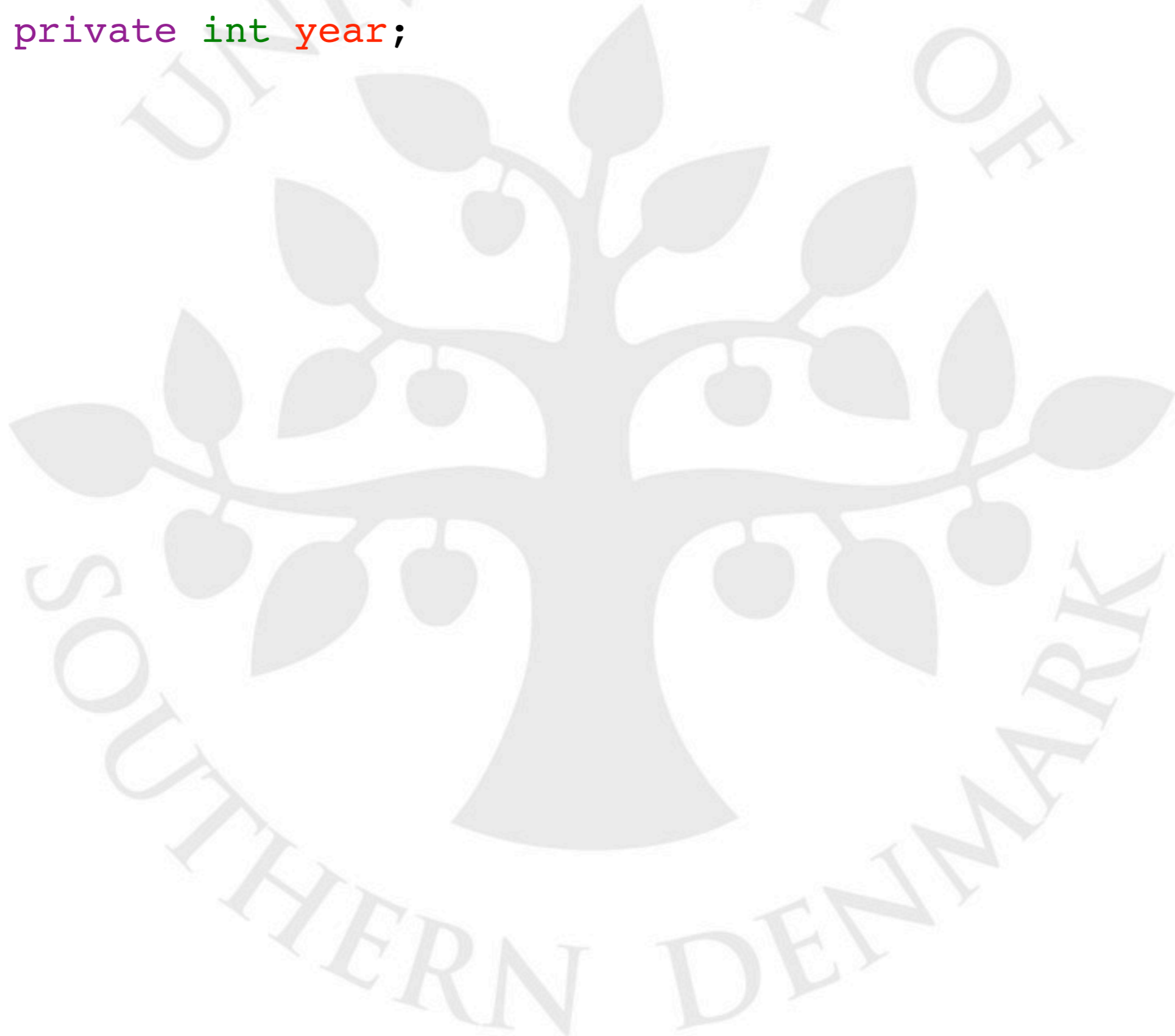
- Lad os tage et tættere kig på klassen Car
- `public class Car { ... }`
 - `class Car`
 - Laver en klasse Car
 - Skal ligge i Car.java
 - `public`
 - En klasse som er offentligt tilgængelig for andre
 - Vi kan bruge den i vores carMaker-klasse
 - I modsætning til `private`

Public og private



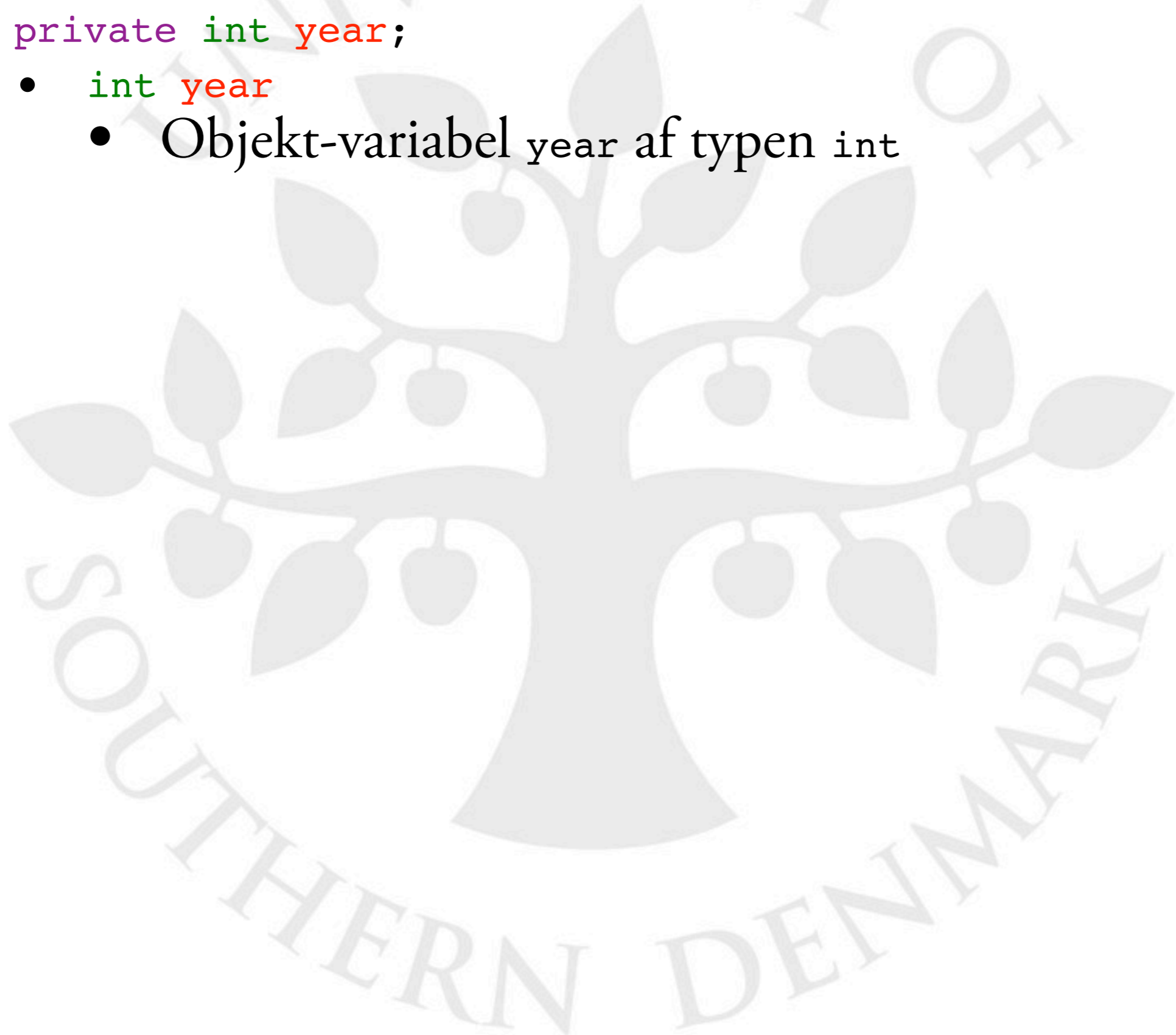
Public og private

- `private int year;`



Public og private

- `private int year;`
 - `int year`
 - Objekt-variabel `year` af typen `int`



Public og private

- `private int year;`
 - `int year`
 - Objekt-variabel `year` af typen `int`
 - `private`
 - Variablen kan kun bruges internt i klassen
 - I modsætning til `public`



Public og private

- `private int year;`
 - `int year`
 - Objekt-variabel `year` af typen `int`
 - `private`
 - Variablen kan kun bruges internt i klassen
 - I modsætning til `public`
- Eksempel på `public`
 - Math-klassens `PI` og `E` er `public`



Public og private

- `private int year;`
 - `int year`
 - Objekt-variabel `year` af typen `int`
 - `private`
 - Variablen kan kun bruges internt i klassen
 - I modsætning til `public`
- Eksempel på `public`
 - Math-klassens `PI` og `E` er `public`
 - `Math.PI`



Public og private

- `private int year;`
 - `int year`
 - Objekt-variabel `year` af typen `int`
 - `private`
 - Variablen kan kun bruges internt i klassen
 - I modsætning til `public`
- Eksempel på `public`
 - Math-klassens `PI` og `E` er `public`
 - `Math.PI`
 - `Math.E`

Public og private



Public og private

- ```
public int getMileage() {
 return mileage;
}
```



# Public og private

- ```
public int getMileage() {  
    return mileage;  
}
```
- ```
int getMileage()
```

  - Metode med navnet `getMileage` der returnerer en `int` og som ikke tager nogen parametre



# Public og private

- ```
public int getMileage() {  
    return mileage;  
}
```
- ```
int getMileage()
```

  - Metode med navnet `getMileage` der returnerer en `int` og som ikke tager nogen parametre
- ```
public
```

 - Metoden kan bruges uden for klassen
 - I modsætning til `private`
 - Bruges til metoder der kun skal anvendes internt i klassen

Public og private

- ```
public int getMileage() {
 return mileage;
}
```
- ```
int getMileage()
```

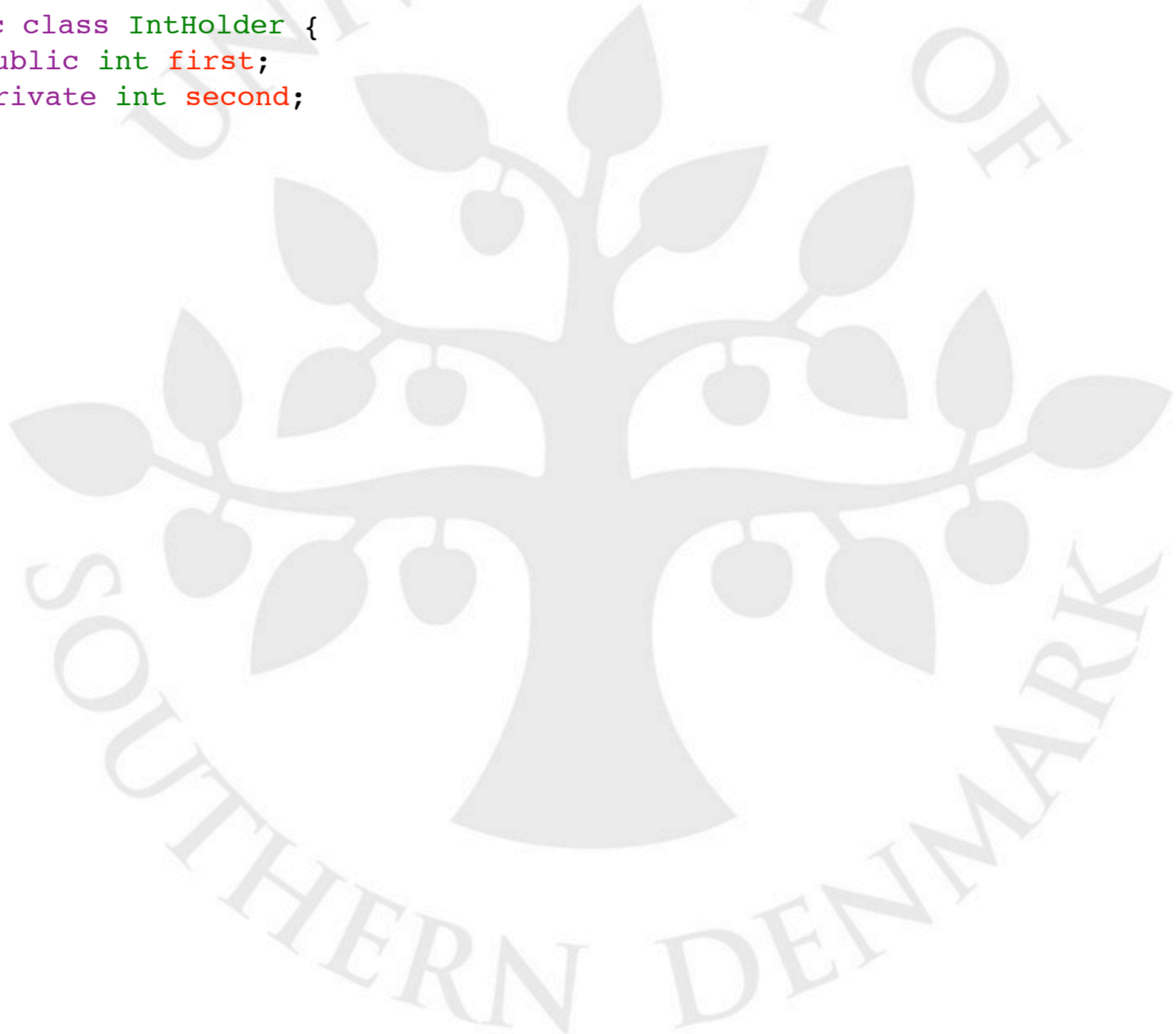
 - Metode med navnet `getMileage` der returnerer en `int` og som ikke tager nogen parametre
- ```
public
```

  - Metoden kan bruges uden for klassen
  - I modsætning til `private`
    - Bruges til metoder der kun skal anvendes internt i klassen
- Bare rolig ... eksempel på næste slide :-)



# Eksempel

```
public class IntHolder {
 public int first;
 private int second;
}
```



# Eksempel

```
public class IntHolder {
 public int first;
 private int second;

 public IntHolder(int f, int s) {
 first = f;
 second = s;
 }
}
```



# Eksempel

```
public class IntHolder {
 public int first;
 private int second;

 public IntHolder(int f, int s) {
 first = f;
 second = s;
 }

 public void updateFirst(int f) {
 first = f;
 }
}
```



# Eksempel

```
public class IntHolder {
 public int first;
 private int second;

 public IntHolder(int f, int s) {
 first = f;
 second = s;
 }

 public void updateFirst(int f) {
 first = f;
 }

 private void updateSecond(int s) {
 second = s;
 }
}
```

# Eksempel

```
public class IntHolder {
 public int first;
 private int second;

 public IntHolder(int f, int s) {
 first = f;
 second = s;
 }

 public void updateFirst(int f) {
 first = f;
 }

 private void updateSecond(int s) {
 second = s;
 }

 public void updateBoth(int f, int s) {
 updateFirst(f);
 updateSecond(s);
 }
}
```

# Eksempel

```
public class IntHolder {
 public int first;
 private int second;

 public IntHolder(int f, int s) {
 first = f;
 second = s;
 }

 public void updateFirst(int f) {
 first = f;
 }

 private void updateSecond(int s) {
 second = s;
 }

 public void updateBoth(int f, int s) {
 updateFirst(f);
 updateSecond(s);
 }

 public void printValues() {
 System.out.println("First: " + first);
 System.out.println("Second: " + second);
 }
}
```

# Eksempel

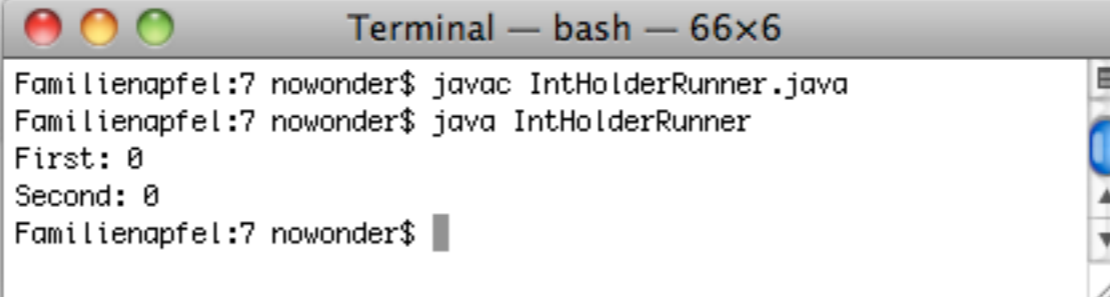
```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);
 }
}
```



# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.printValues();
 }
}
```



```
Terminal — bash — 66x6
Familienapfel:7 nowonder$ javac IntHolderRunner.java
Familienapfel:7 nowonder$ java IntHolderRunner
First: 0
Second: 0
Familienapfel:7 nowonder$
```



# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.printValues();
 }
}
```

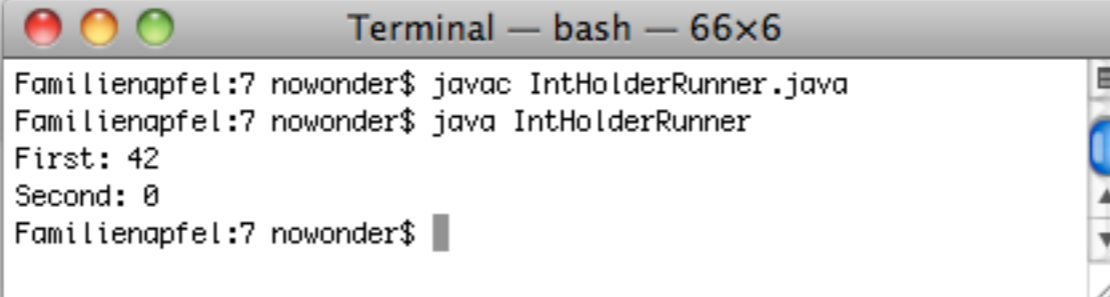


# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.printValues();
 }
}
```



Terminal — bash — 66x6

```
Familienapfel:7 nowonder$ javac IntHolderRunner.java
Familienapfel:7 nowonder$ java IntHolderRunner
First: 42
Second: 0
Familienapfel:7 nowonder$
```

# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.second = 7;

 h.printValues();
 }
}
```



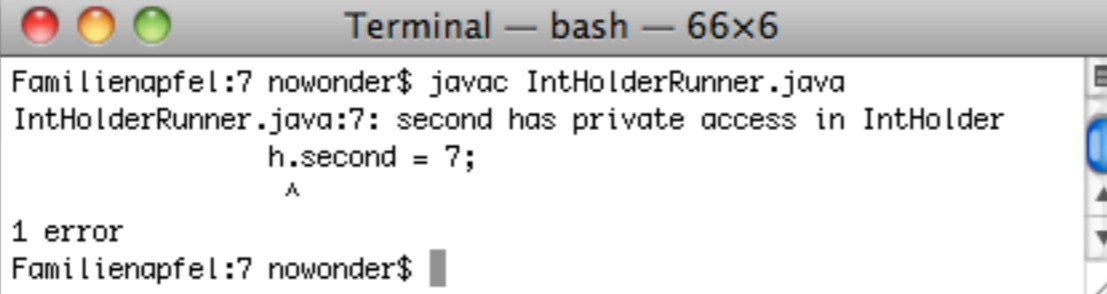
# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.second = 7;

 h.printValues();
 }
}
```

A terminal window titled "Terminal — bash — 66x6" showing the compilation of the provided Java code. The command executed is "javac IntHolderRunner.java". The output shows a compilation error: "IntHolderRunner.java:7: second has private access in IntHolder" with a caret pointing to "h.second = 7;". Below the error message, it says "1 error" and the prompt "Familienapfel:7 nowonder\$" is shown again.

```
Terminal — bash — 66x6
Familienapfel:7 nowonder$ javac IntHolderRunner.java
IntHolderRunner.java:7: second has private access in IntHolder
 h.second = 7;
 ^
1 error
Familienapfel:7 nowonder$
```

# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.updateFirst(3);

 h.printValues();
 }
}
```



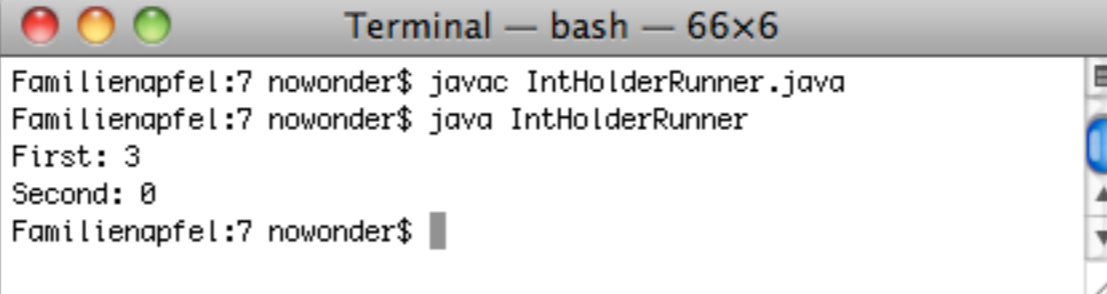
# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.updateFirst(3);

 h.printValues();
 }
}
```



```
Terminal — bash — 66x6
Familienapfel:7 nowonder$ javac IntHolderRunner.java
Familienapfel:7 nowonder$ java IntHolderRunner
First: 3
Second: 0
Familienapfel:7 nowonder$
```

# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.updateFirst(3);

 h.updateSecond(5);

 h.printValues();
 }
}
```



# Eksempel

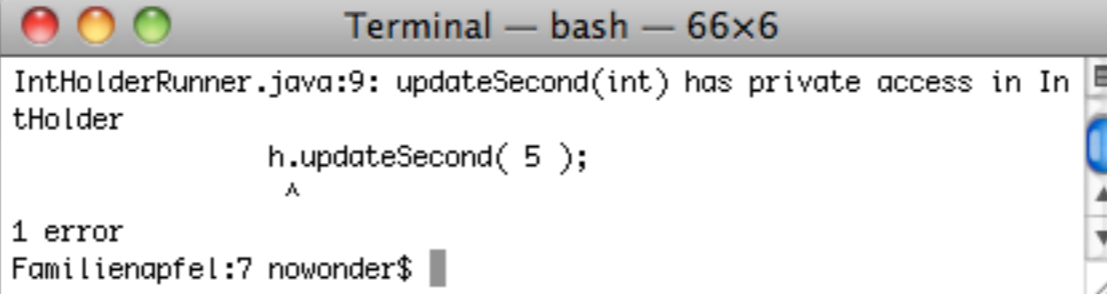
```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.updateFirst(3);

 h.updateSecond(5);

 h.printValues();
 }
}
```



Terminal — bash — 66x6

```
IntHolderRunner.java:9: updateSecond(int) has private access in In
tHolder
 h.updateSecond(5);
 ^
1 error
Familienapfel:7 nowonder$
```



# Eksempel

```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.updateFirst(3);

 h.updateBoth(7, 5);

 h.printValues();
 }
}
```



# Eksempel

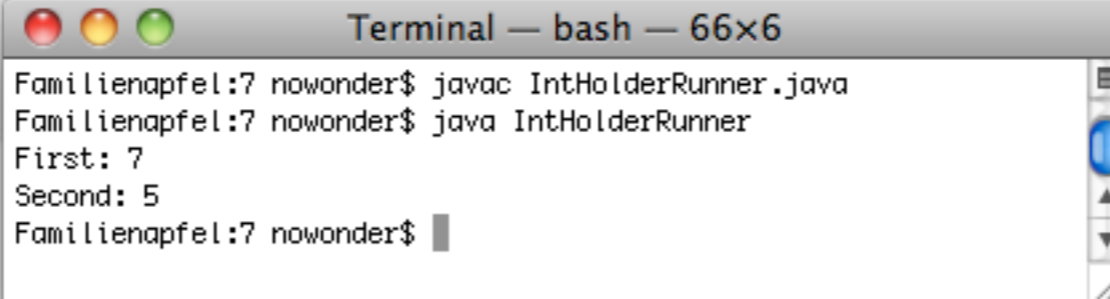
```
public class IntHolderRunner {
 public static void main(String[] args) {
 IntHolder h = new IntHolder(0, 0);

 h.first = 42;

 h.updateFirst(3);

 h.updateBoth(7, 5);

 h.printValues();
 }
}
```



```
Terminal — bash — 66x6
Familienapfel:7 nowonder$ javac IntHolderRunner.java
Familienapfel:7 nowonder$ java IntHolderRunner
First: 7
Second: 5
Familienapfel:7 nowonder$
```

# Public og private



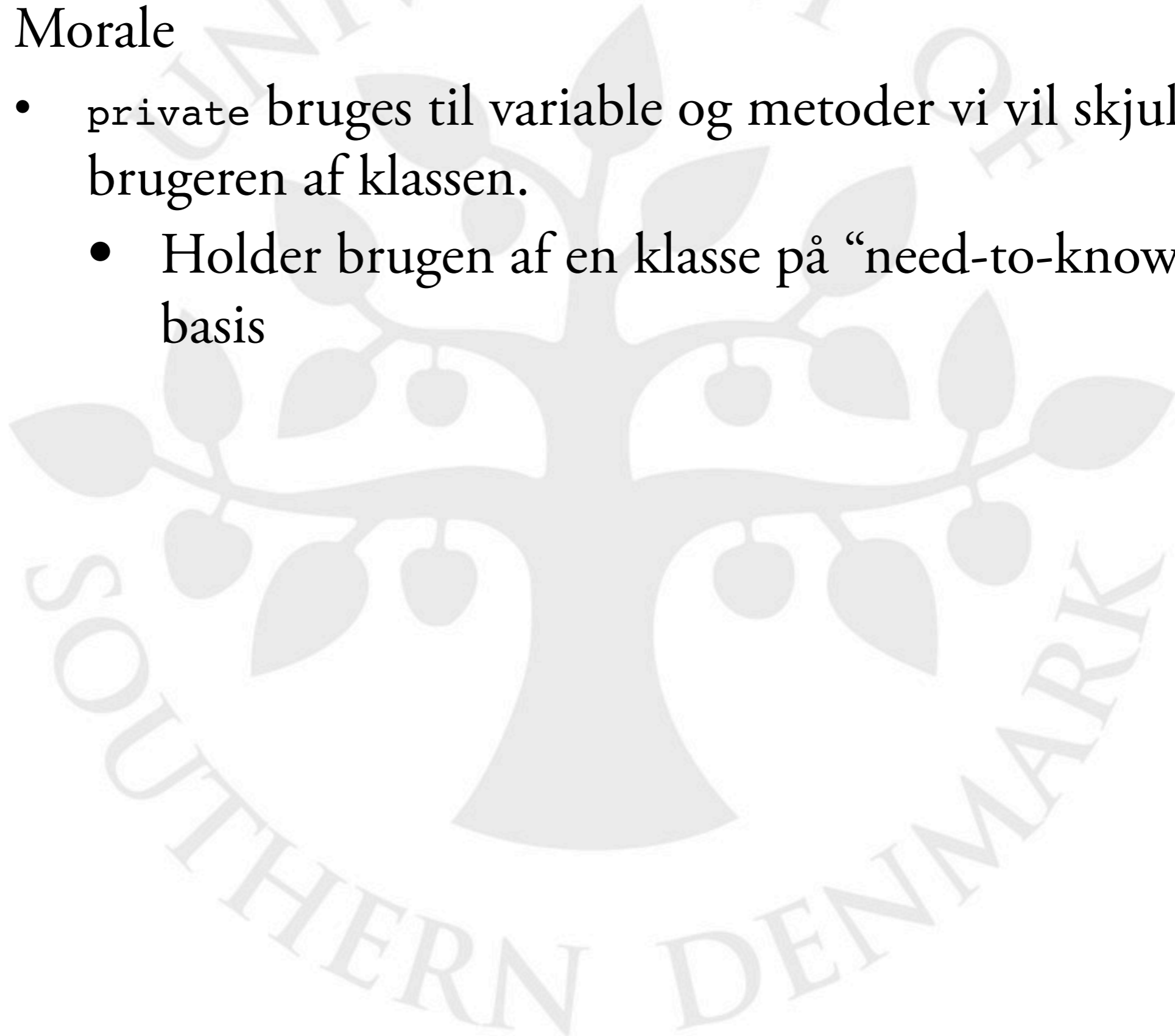
# Public og private

- Morale



# Public og private

- Morale
  - private bruges til variable og metoder vi vil skjule fra brugeren af klassen.
    - Holder brugen af en klasse på “need-to-know” basis



# Public og private

- Morale
  - `private` bruges til variable og metoder vi vil skjule fra brugeren af klassen.
    - Holder brugen af en klasse på “need-to-know” basis
  - `public` bruges til variable og metoder som brugeren skal have adgang til

# Public og private

- Morale
  - `private` bruges til variable og metoder vi vil skjule fra brugeren af klassen.
    - Holder brugen af en klasse på “need-to-know” basis
  - `public` bruges til variable og metoder som brugeren skal have adgang til
- Giver en veldefineret adgang til en klasse

# Public og private

- Morale
  - `private` bruges til variable og metoder vi vil skjule fra brugeren af klassen.
    - Holder brugen af en klasse på “need-to-know” basis
  - `public` bruges til variable og metoder som brugeren skal have adgang til
- Giver en veldefineret adgang til en klasse
- Eksempel - `car`
  - `drejTilHøjre` skal nok være `public`
  - `sprøjtBenzinIForbrændingskammer` skal nok være `private`





# Static



# Static

- Keywordet kan bruges på både metoder og variable
  - `FX public static void main( String[] args )`
  - `FX public static double PI`
  - `FX public static double sin( double a )`

# Static

- Keywordet kan bruges på både metoder og variable
  - `FX public static void main( String[] args )`
  - `FX public static double PI`
  - `FX public static double sin( double a )`
- Angiver at metoden eller variabelen hører til klassen
  - I modsætning til at metoden eller variabelen hører til et objekt
  - Der findes kun ét (totalt set) eksemplar af statiske variable eller metoder

# Static

- Keywordet kan bruges på både metoder og variable
  - Fx `public static void main( String[] args )`
  - Fx `public static double PI`
  - Fx `public static double sin( double a )`
- Angiver at metoden eller variabelen hører til klassen
  - I modsætning til at metoden eller variabelen hører til et objekt
  - Der findes kun ét (totalt set) eksemplar af statiske variable eller metoder
- Metoderne og variablerne i Math-klassen

# Eksempel

```
public class Car {
 private String model;
 private int year;
 private String color;
 private int mileage;

 private static int totalCars = 0;

 public Car(String carModel, String carColor) {
 model = carModel;
 year = 2008;
 color = carColor;
 mileage = 0;
 Car.totalCars = Car.totalCars + 1;
 }

 public static int getTotalCars() {
 return Car.totalCars;
 }

 ...
}
```

# Eksempel

```
public class Car {
 private String model;
 private int year;
 private String color;
 private int mileage;

 private static int totalCars = 0;

 public Car(String carModel, String carColor) {
 model = carModel;
 year = 2008;
 color = carColor;
 mileage = 0;
 Car.totalCars = Car.totalCars + 1;
 }

 public static int getTotalCars () {
 return Car.totalCars;
 }

 ...
}
```



# Eksempel

```
public class CarMaker {
 public static void main(String[] args) {
 Car car1, car2, car3;

 System.out.println("Total number of cars: " + Car.getTotalCars());

 car1 = new Car("Audi A4", "sølvgrå");

 System.out.println("Total number of cars: " + Car.getTotalCars());

 car2 = new Car("Skoda Fabia", "rød");

 System.out.println("Total number of cars: " + Car.getTotalCars());

 car3 = new Car("VW Polo", "blå");

 System.out.println("Total number of cars: " + Car.getTotalCars());
 }
}
```

# Eksempel

```
public class CarMaker {
 public static void main(String[] args) {
 Car car1, car2, car3;

 System.out.println("Total number of cars: " + Car.getTotalCars());

 car1 = new Car("Audi A4", "sølvgrå");

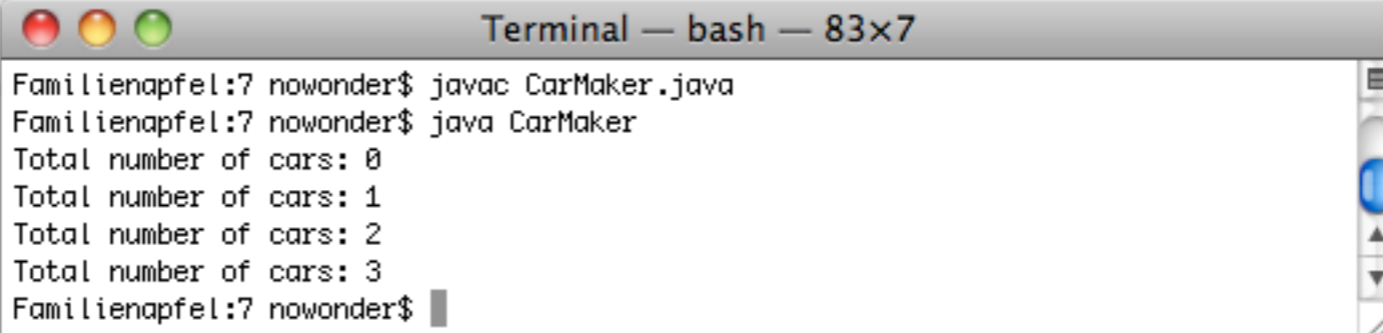
 System.out.println("Total number of cars: " + Car.getTotalCars());

 car2 = new Car("Skoda Fabia", "rød");

 System.out.println("Total number of cars: " + Car.getTotalCars());

 car3 = new Car("VW Polo", "blå");

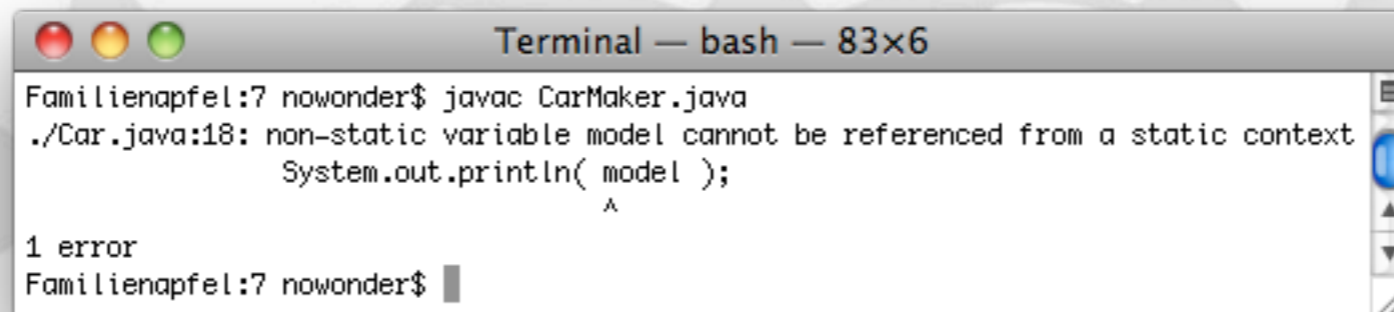
 System.out.println("Total number of cars: " + Car.getTotalCars());
 }
}
```



```
Terminal — bash — 83x7
Familienapfel:7 nowonder$ javac CarMaker.java
Familienapfel:7 nowonder$ java CarMaker
Total number of cars: 0
Total number of cars: 1
Total number of cars: 2
Total number of cars: 3
Familienapfel:7 nowonder$
```



# Static



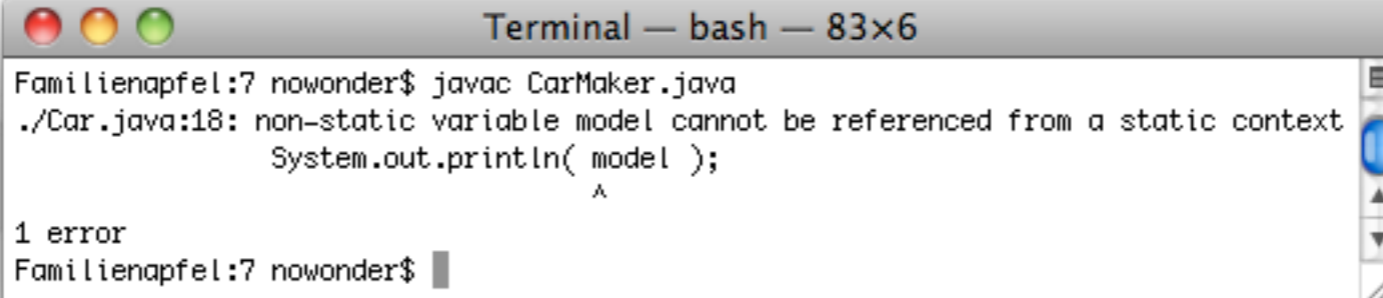
A terminal window titled "Terminal — bash — 83x6" showing the output of a Java compilation. The prompt is "Familienapfel:7 nowonder\$". The command executed is "javac CarMaker.java". The output shows a compilation error: "./Car.java:18: non-static variable model cannot be referenced from a static context". The error message is followed by the code snippet "System.out.println( model );" with a caret (^) pointing to the variable "model". Below the error message, it says "1 error" and "Familienapfel:7 nowonder\$".

```
Familienapfel:7 nowonder$ javac CarMaker.java
./Car.java:18: non-static variable model cannot be referenced from a static context
 System.out.println(model);
 ^
1 error
Familienapfel:7 nowonder$
```



# Static

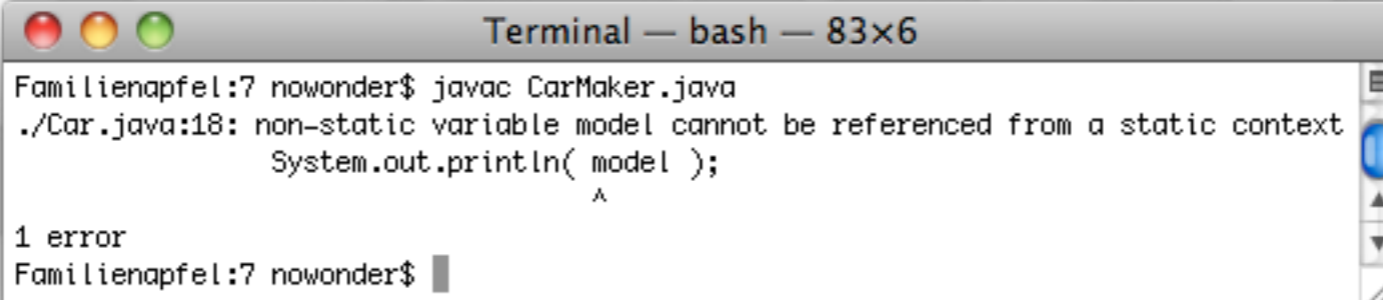
- Fra metoden `getTotalCars` kan vi ikke tilgå variablene `model`, `year`, `color` og `mileage`



```
Terminal — bash — 83x6
Familienapfel:7 nowonder$ javac CarMaker.java
./Car.java:18: non-static variable model cannot be referenced from a static context
 System.out.println(model);
 ^
1 error
Familienapfel:7 nowonder$
```

# Static

- Fra metoden `getTotalCars` kan vi ikke tilgå variablene `model`, `year`, `color` og `mileage`
  - ```
public static int getTotalCars() {  
    System.out.println( model );  
    return Car.totalCars;  
}
```

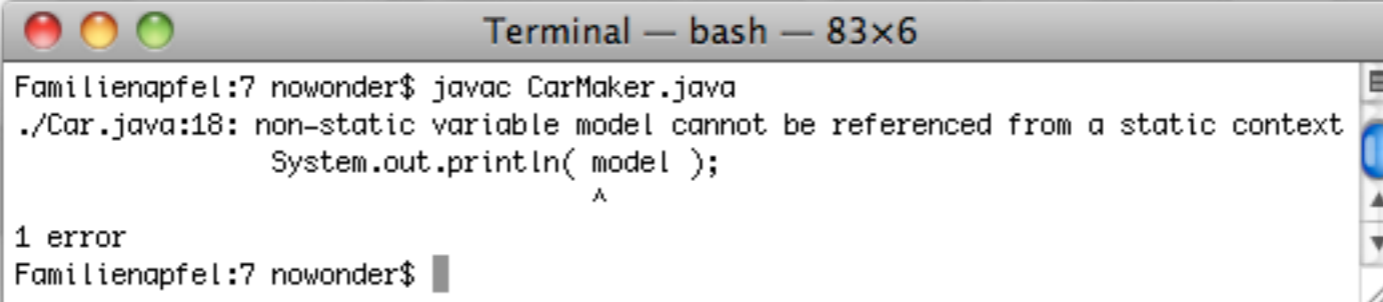


A terminal window titled "Terminal — bash — 83x6" showing the compilation of `CarMaker.java`. The command `javac CarMaker.java` is executed, resulting in a compilation error: `./Car.java:18: non-static variable model cannot be referenced from a static context`. The error points to the `System.out.println(model);` line in the code. The terminal also shows "1 error" and the prompt `Familienapfel:7 nowonder$`.

```
Terminal — bash — 83x6  
Familienapfel:7 nowonder$ javac CarMaker.java  
./Car.java:18: non-static variable model cannot be referenced from a static context  
    System.out.println( model );  
                        ^  
1 error  
Familienapfel:7 nowonder$
```

Static

- Fra metoden `getTotalCars` kan vi ikke tilgå variablene `model`, `year`, `color` og `mileage`
 - ```
public static int getTotalCars() {
 System.out.println(model);
 return Car.totalCars;
}
```

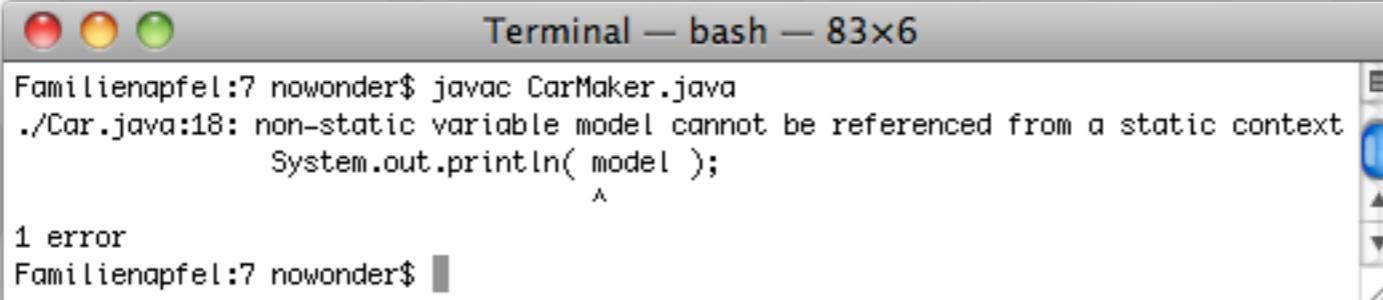
A screenshot of a terminal window titled "Terminal — bash — 83x6". The terminal shows the command `javac CarMaker.java` being executed. The output is an error message: `./Car.java:18: non-static variable model cannot be referenced from a static context`. Below the error message, the code `System.out.println( model );` is shown with a caret under the `model` variable. At the bottom, it says "1 error" and the prompt `Familienapfel:7 nowonder$` is visible.

```
Terminal — bash — 83x6
Familienapfel:7 nowonder$ javac CarMaker.java
./Car.java:18: non-static variable model cannot be referenced from a static context
 System.out.println(model);
 ^
1 error
Familienapfel:7 nowonder$
```

- De fire variable hører til et objekt (de har en bestemt værdi i hvert objekt)

# Static

- Fra metoden `getTotalCars` kan vi ikke tilgå variablene `model`, `year`, `color` og `mileage`
  - ```
public static int getTotalCars() {  
    System.out.println( model );  
    return Car.totalCars;  
}
```



```
Terminal — bash — 83x6  
Familienapfel:7 nowonder$ javac CarMaker.java  
./Car.java:18: non-static variable model cannot be referenced from a static context  
    System.out.println( model );  
                        ^  
1 error  
Familienapfel:7 nowonder$
```

- De fire variable hører til et objekt (de har en bestemt værdi i hvert objekt)
- I en statisk metode opererer vi ikke på et objekt, vi opererer på klassen

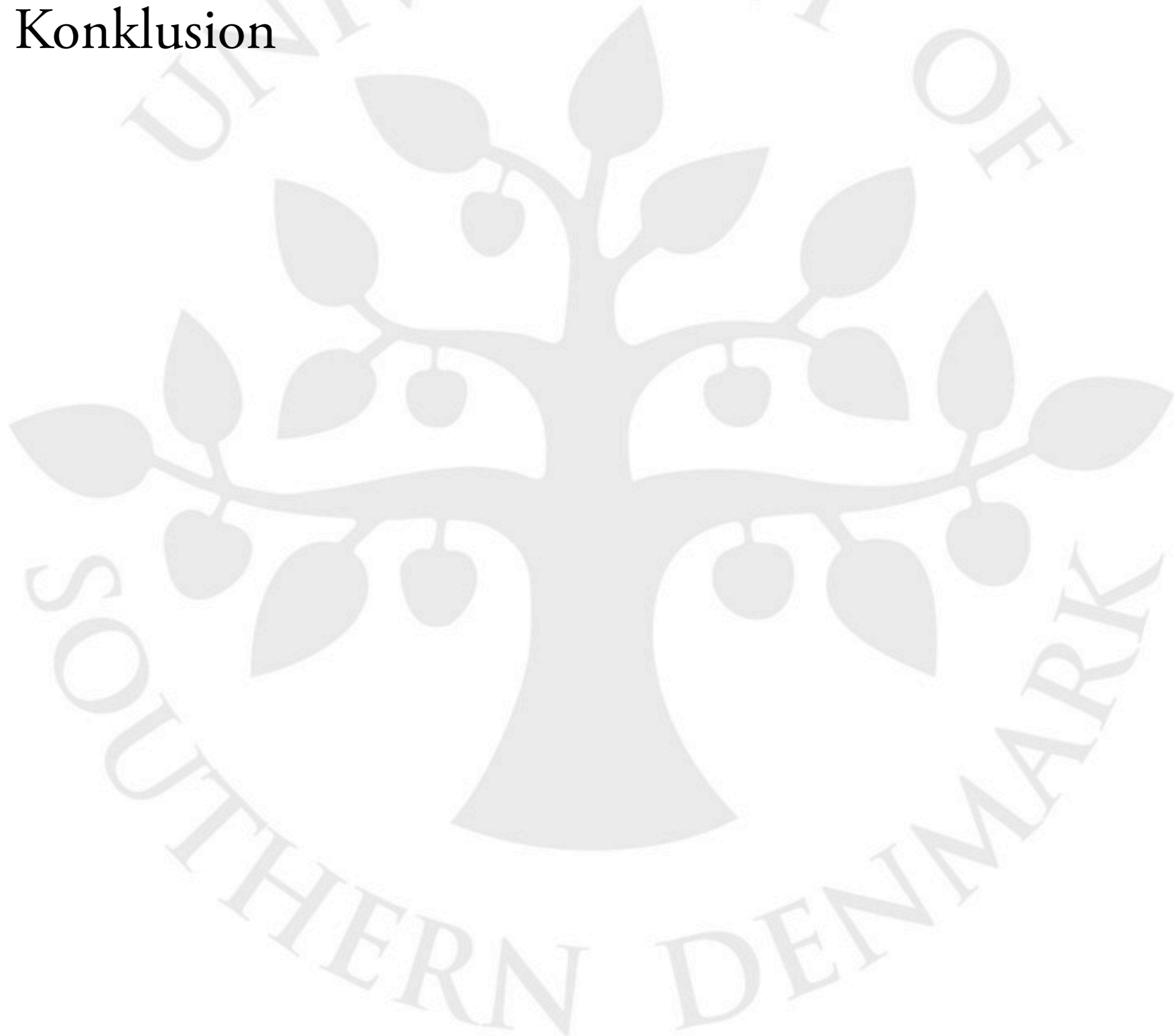


Static



Static

- Konklusion



Static

- Konklusion
 - Ikke-statiske metoder og variable
 - En udgave for hver instans/objekt
 - Kan tilgå statiske metoder og variable



Static

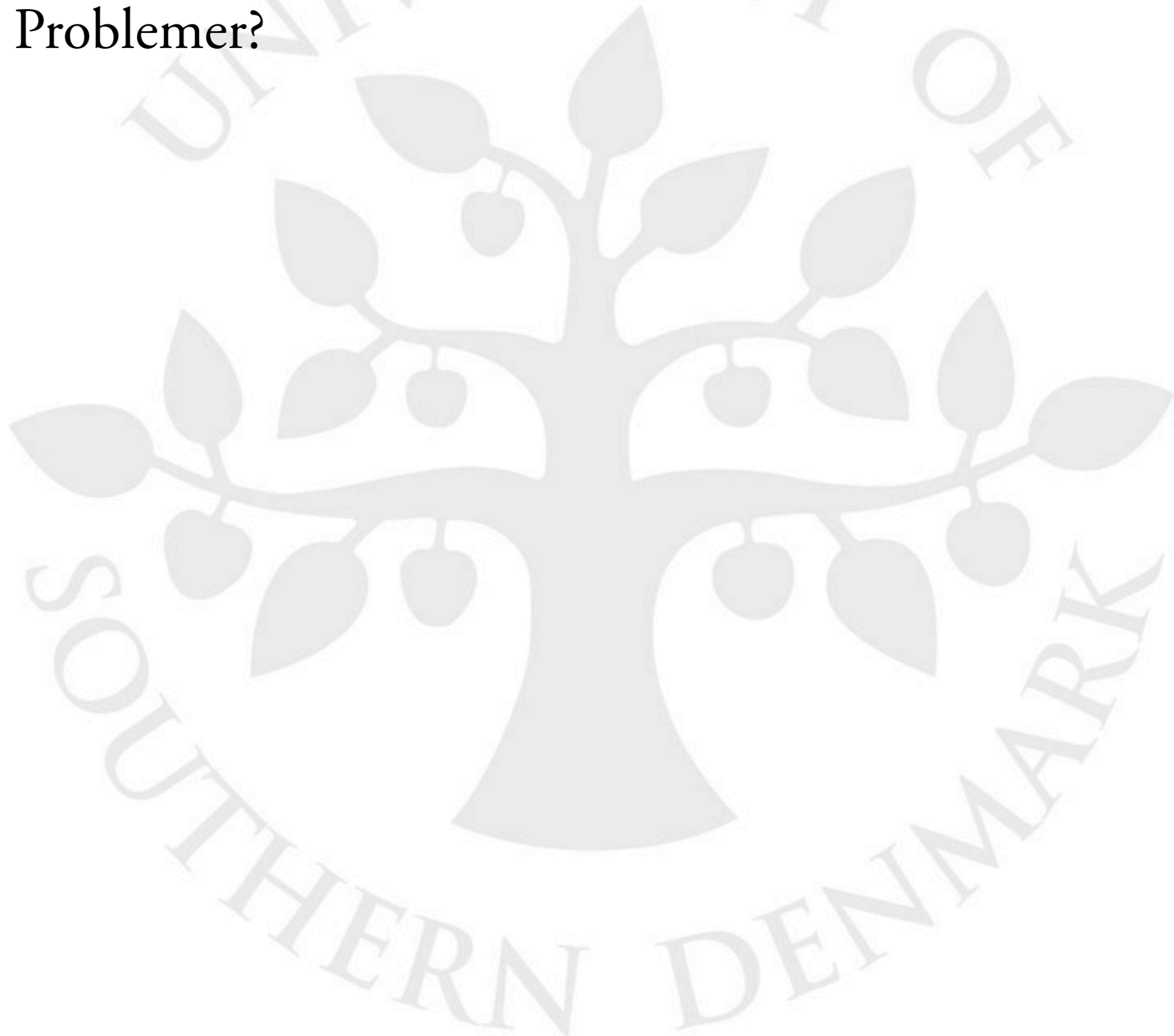
- Konklusion
 - Ikke-statiske metoder og variable
 - En udgave for hver instans/objekt
 - Kan tilgå statiske metoder og variable
 - Statiske metoder og variable
 - En udgave totalt set (uanset antallet af objekter)
 - Kan ikke tilgå ikke-statiske metoder og variable

Delprojekt 1



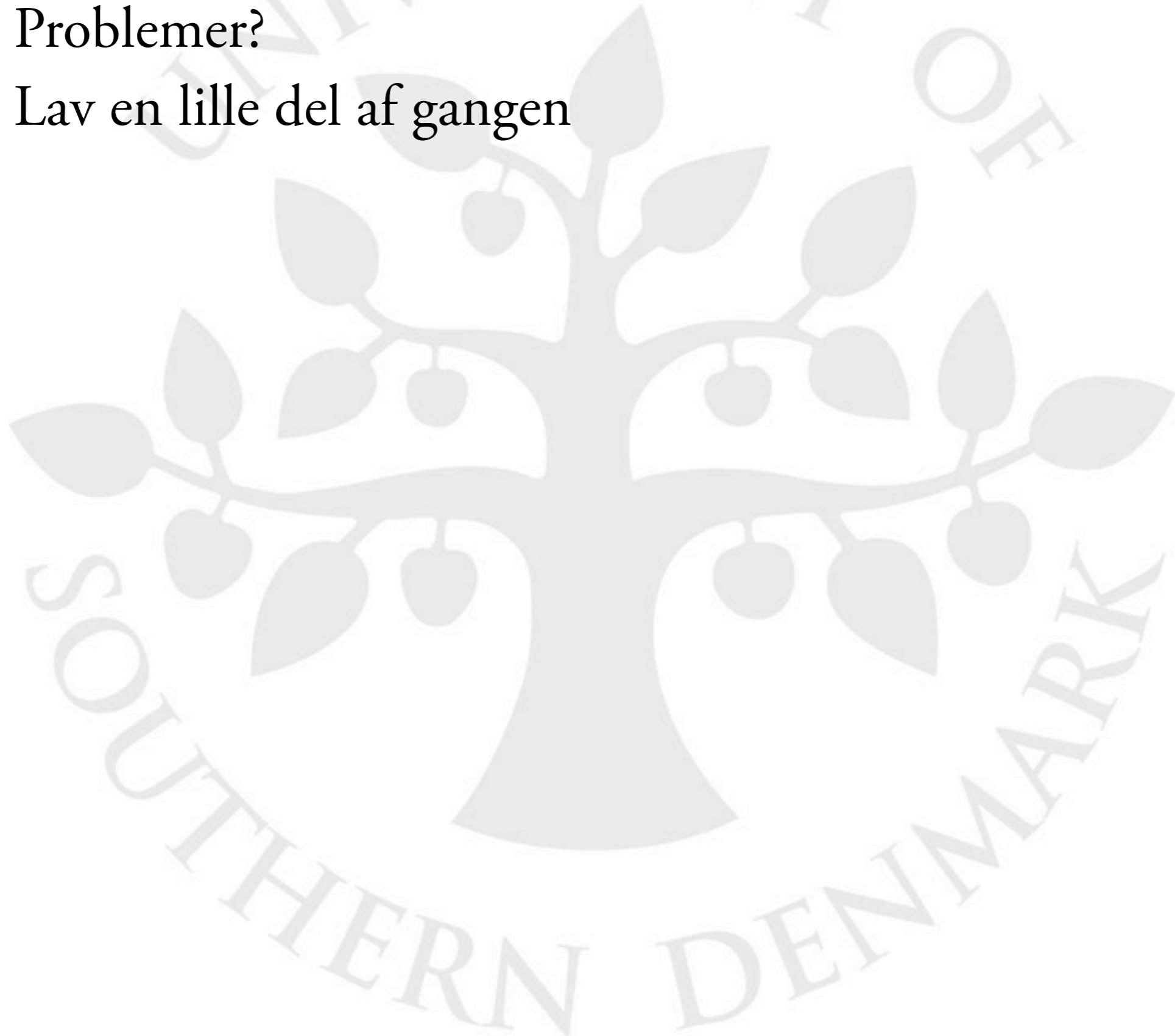
Delprojekt 1

- Problemer?



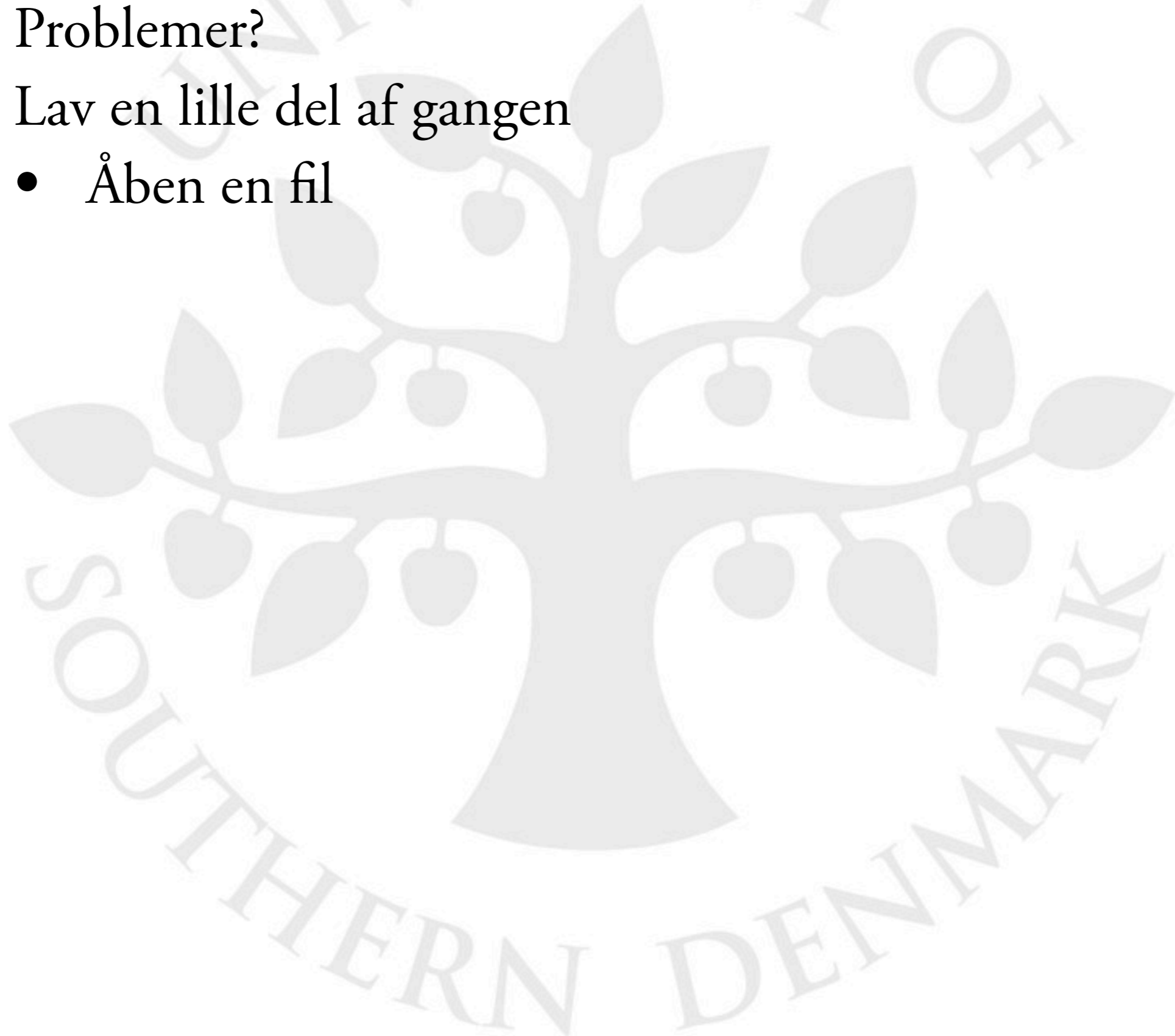
Delprojekt 1

- Problemer?
- Lav en lille del af gangen



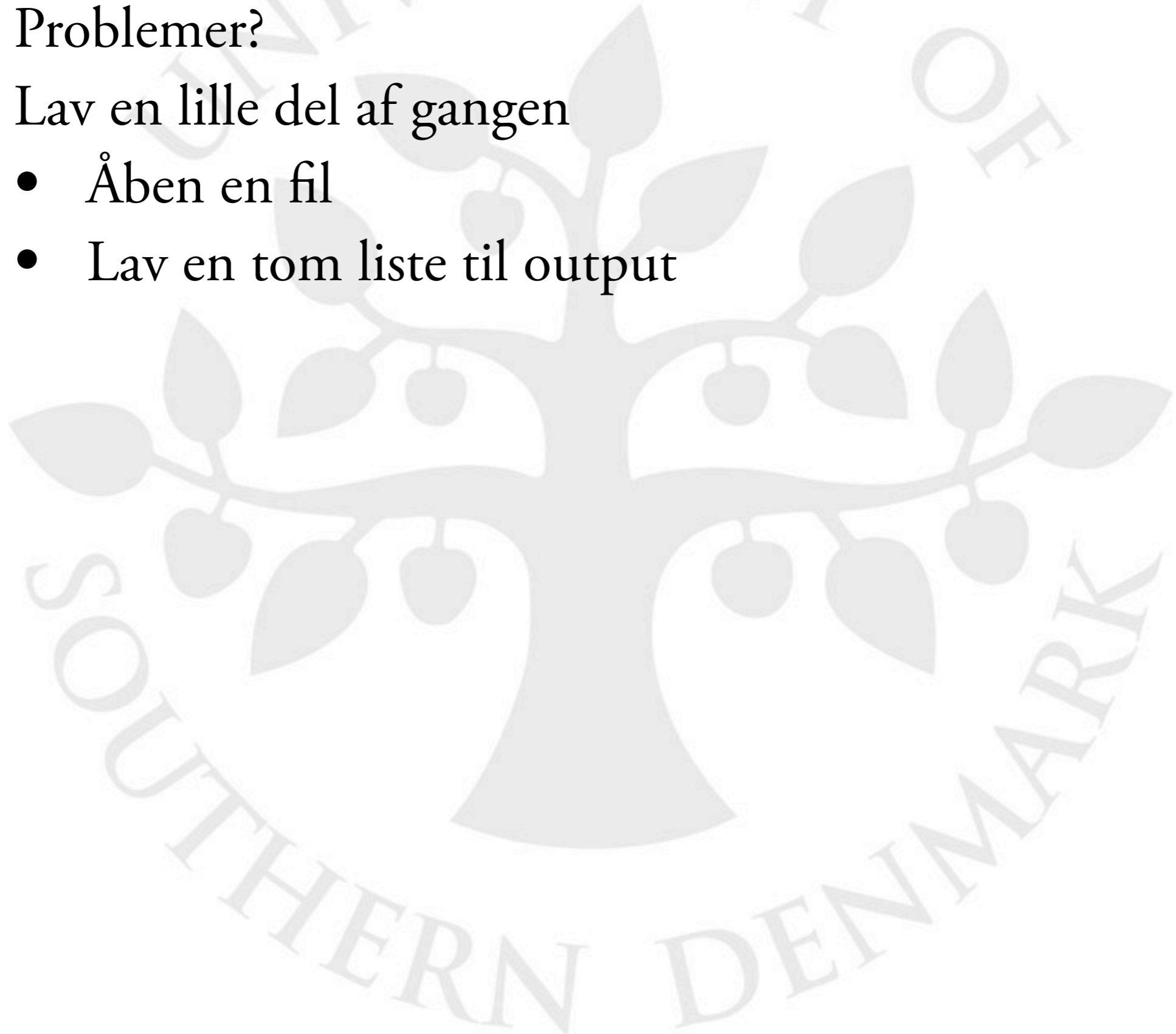
Delprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil



Delprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output



Delprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen



Delfprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen
 - **Krypter hver tegn i ordet**



Delprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen
 - **Krypter hver tegn i ordet**
 - **Tilføj hvert krypteret tegn til resultatordet**



Delprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen
 - **Krypter hver tegn i ordet**
 - **Tilføj hvert krypteret tegn til resultatordet**
 - Tilføj resultatordet til output-listen

Delfprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen
 - **Krypter hver tegn i ordet**
 - **Tilføj hvert krypteret tegn til resultatordet**
 - Tilføj resultatordet til output-listen
 - Udskriv output-listen

Delfprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen
 - **Krypter hver tegn i ordet**
 - **Tilføj hvert krypteret tegn til resultatordet**
 - Tilføj resultatordet til output-listen
 - Udskriv output-listen
- Husk at du kan udskrive efter hvert skridt

Delfprojekt 1

- Problemer?
- Lav en lille del af gangen
 - Åben en fil
 - Lav en tom liste til output
 - Læs hvert ord fra filen
 - **Krypter hver tegn i ordet**
 - **Tilføj hvert krypteret tegn til resultatordet**
 - Tilføj resultatordet til output-listen
 - Udskriv output-listen
- Husk at du kan udskrive efter hvert skridt
- Brug en debugger!

Eksempel



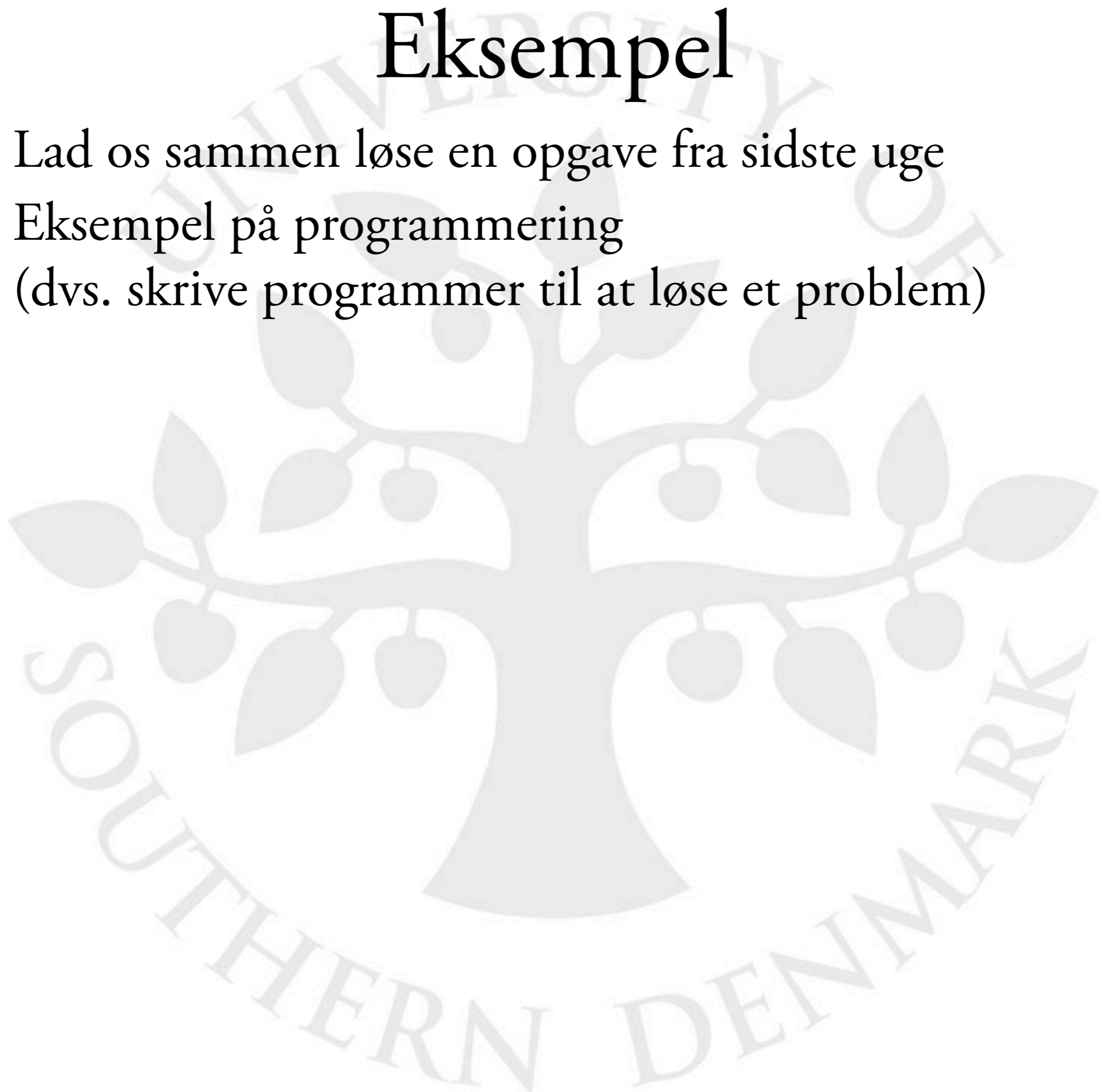
Eksempel

- Lad os sammen løse en opgave fra sidste uge



Eksempel

- Lad os sammen løse en opgave fra sidste uge
- Eksempel på programmering
(dvs. skrive programmer til at løse et problem)



Eksempel

- Lad os sammen løse en opgave fra sidste uge
- Eksempel på programmering
(dvs. skrive programmer til at løse et problem)
- Kræver lidt kreativ tænkning

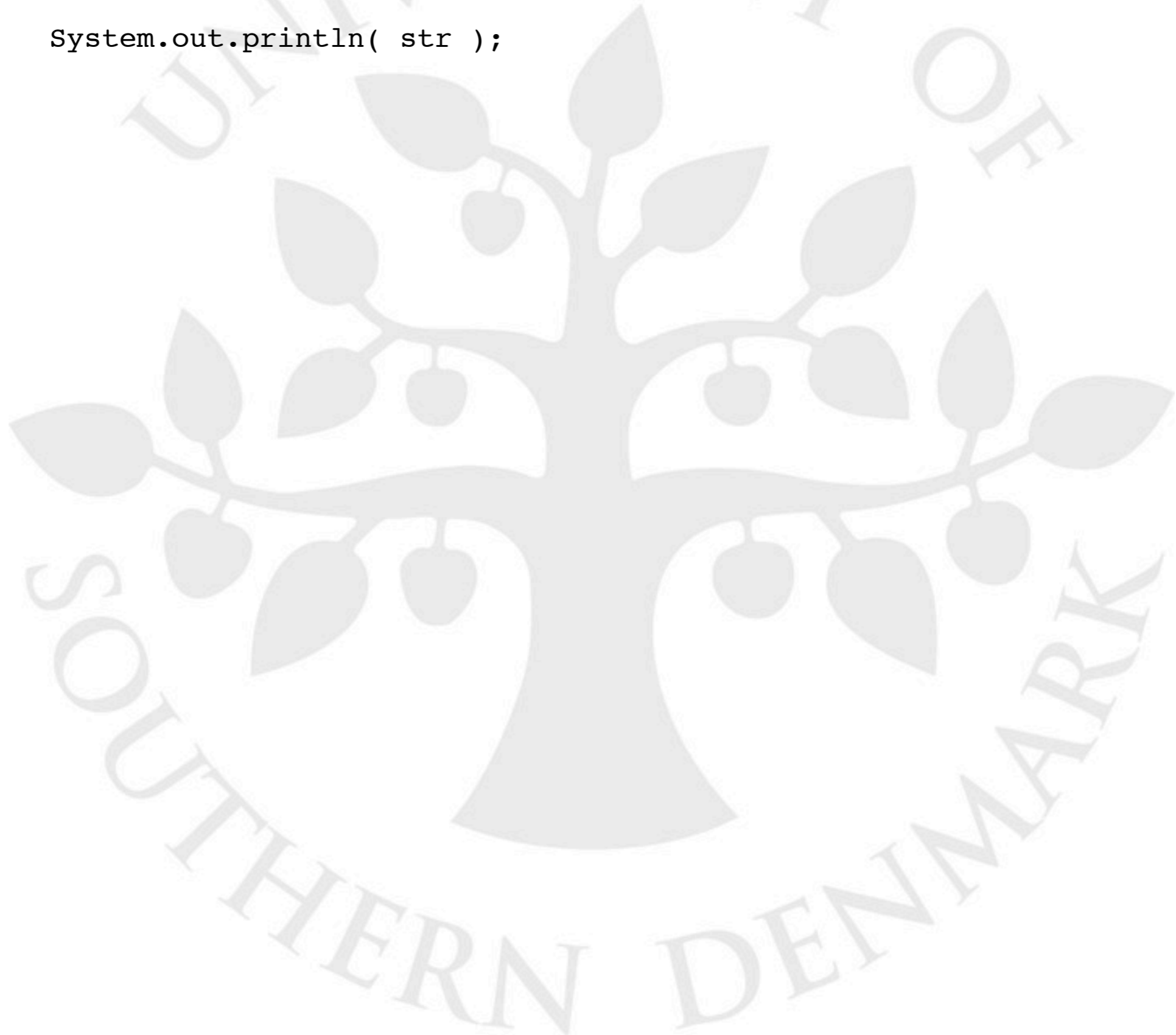


Eksempel

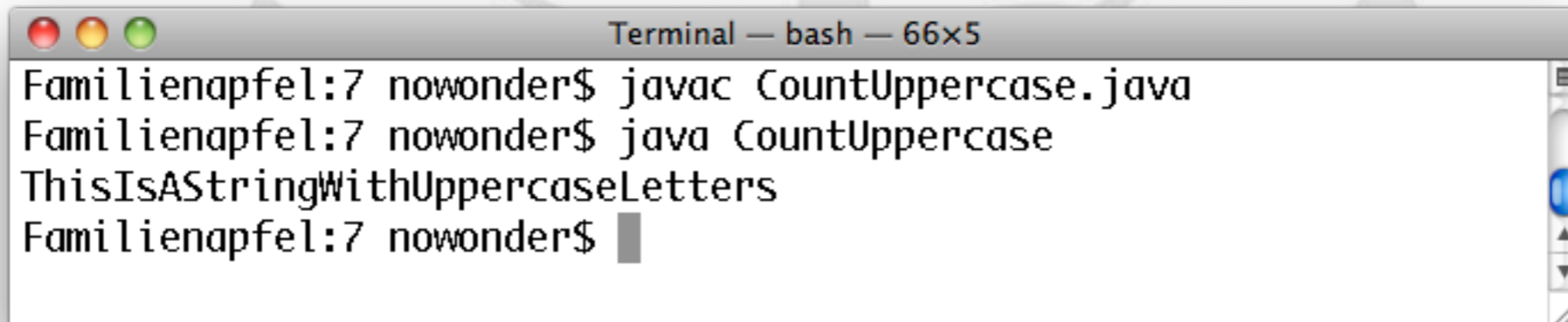
- Lad os sammen løse en opgave fra sidste uge
- Eksempel på programmering
(dvs. skrive programmer til at løse et problem)
 - Kræver lidt kreativ tænkning
- Problem: Tæl antallet af store bogstaver i en String



```
public class CountUppercase {  
    public static void main( String[] args ) {  
        String str = "ThisIsAStringWithUppercaseLetters";  
  
        System.out.println( str );  
    }  
}
```



```
public class CountUppercase {  
    public static void main( String[] args ) {  
        String str = "ThisIsAStringWithUppercaseLetters";  
  
        System.out.println( str );  
    }  
}
```

A terminal window titled "Terminal — bash — 66x5" is shown. The window contains the following text:

```
Familienapfel:7 nowonder$ javac CountUppercase.java  
Familienapfel:7 nowonder$ java CountUppercase  
ThisIsAStringWithUppercaseLetters  
Familienapfel:7 nowonder$ █
```

The terminal window has a standard macOS-style title bar with red, yellow, and green window control buttons on the left and a scroll bar on the right.

```
public class CountUppercase {
    public static void main( String[] args ) {
        String str = "ThisIsAStringWithUppercaseLetters";
        int i;

        for( i = 0; i < str.length(); i++ ) {
            System.out.println( str.charAt( i ) );
        }
    }
}
```





```
public  
pub  
}  
}
```

```
Terminal — bash — 66x36  
Familienapfel:7 nowonder$ javac CountUppercase.java  
Familienapfel:7 nowonder$ java CountUppercase  
T  
h  
i  
s  
i  
s  
s  
A  
S  
t  
r  
i  
n  
g  
W  
i  
t  
h  
U  
p  
p  
e  
r  
c  
a  
s  
e  
L  
e  
t  
t  
e  
r  
s  
Familienapfel:7 nowonder$
```



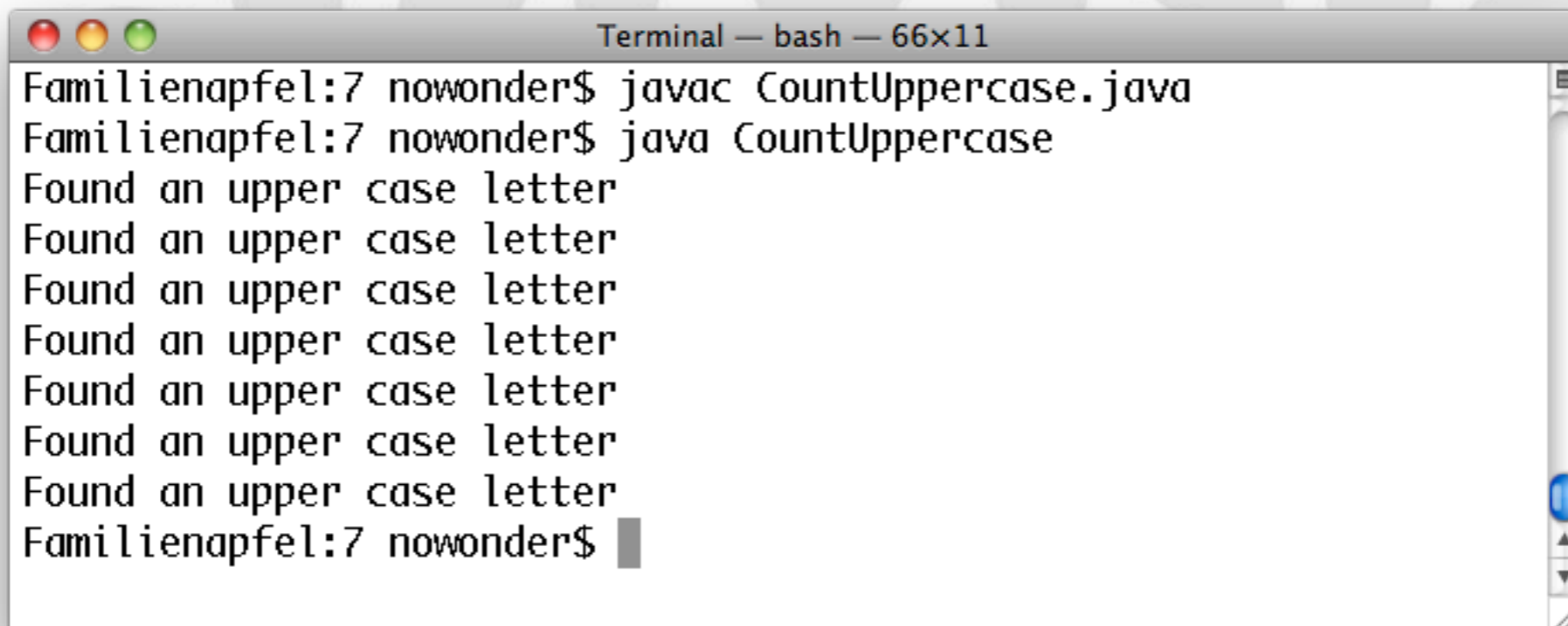
```
public class CountUppercase {
    public static void main( String[] args ) {
        String str = "ThisIsAStringWithUppercaseLetters";
        String STR = str.toUpperCase();
        int i;

        for( i = 0; i < str.length(); i++ ) {
            if( str.charAt( i ) == STR.charAt( i ) ) {
                System.out.println( "Found an upper case letter" );
            }
        }
    }
}
```



```
public class CountUppercase {
    public static void main( String[] args ) {
        String str = "ThisIsAStringWithUppercaseLetters";
        String STR = str.toUpperCase();
        int i;

        for( i = 0; i < str.length(); i++ ) {
            if( str.charAt( i ) == STR.charAt( i ) ) {
                System.out.println( "Found an upper case letter" );
            }
        }
    }
}
```

A terminal window titled "Terminal — bash — 66x11" showing the compilation and execution of the Java program. The user enters the commands to compile and run the program, and the output shows seven lines of "Found an upper case letter" corresponding to the uppercase letters in the string "ThisIsAStringWithUppercaseLetters".

```
Familienapfel:7 nowonder$ javac CountUppercase.java
Familienapfel:7 nowonder$ java CountUppercase
Found an upper case letter
Found an upper case letter
Found an upper case letter
Found an upper case letter
Found an upper case letter
Found an upper case letter
Found an upper case letter
Found an upper case letter
Familienapfel:7 nowonder$
```



```
public class CountUppercase {
    public static void main( String[] args ) {
        String str = "ThisIsAStringWithUppercaseLetters";
        String STR = str.toUpperCase();
        int i;
        int count = 0;

        for( i = 0; i < str.length(); i++ ) {
            if( str.charAt( i ) == STR.charAt( i ) ) {
                count++;
            }
        }

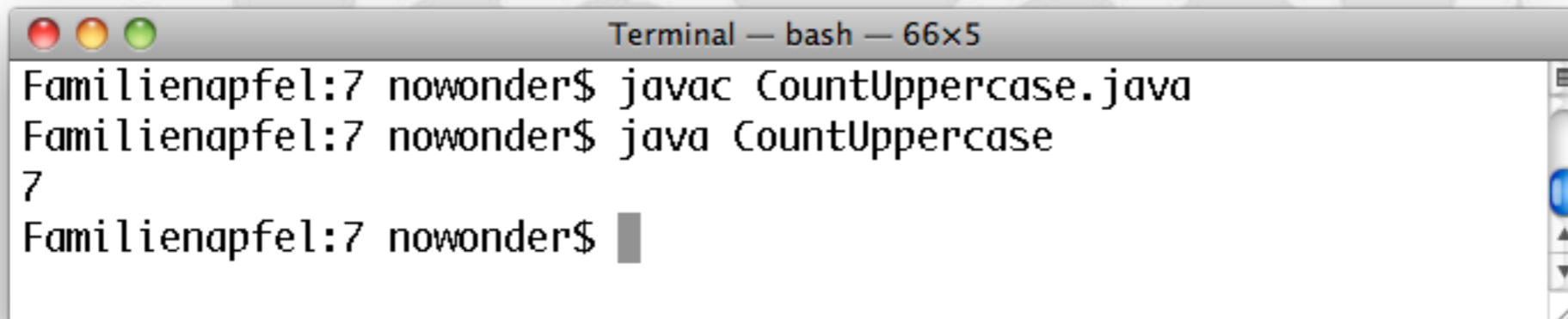
        System.out.println( count );
    }
}
```



```
public class CountUppercase {
    public static void main( String[] args ) {
        String str = "ThisIsAStringWithUppercaseLetters";
        String STR = str.toUpperCase();
        int i;
        int count = 0;

        for( i = 0; i < str.length(); i++ ) {
            if( str.charAt( i ) == STR.charAt( i ) ) {
                count++;
            }
        }

        System.out.println( count );
    }
}
```



```
Terminal — bash — 66x5
Familienapfel:7 nowonder$ javac CountUppercase.java
Familienapfel:7 nowonder$ java CountUppercase
7
Familienapfel:7 nowonder$
```