# DM502

## Forelæsning 6

# Indhold

- Klasser og objekter
  - Introduktion
- `Math`-klassen

# Klasser vs. objekter

- Først et tænkt eksempel: Vi vil lave en bil i Java
- 1. spørgsmål: Hvad karakteriserer en bil?
  - Model
  - År
  - Farve...
- Hver enkelt bil har udfyldte værdier for ovenstående, f.eks.
  - En sølvgrå Audi A4 fra 2001
  - En rød Skoda Fabia fra 2003
- Klasse = Abstrakt beskrivelse
- Objekt = Konkret instans af en klasse
- Bemærk: Ovenstående svarer til variable

# Klasser vs. objekter

- 2. spørgsmål: Hvordan interagerer vi med et objekt (fx en sølvgrå Audi)?
  - Hvordan farver vi bilen?
  - Hvordan kører vi en tur?
  - Hvordan ...

- Vi vil gerne kunne ændre på variable
  - At farve bilen ændrer på farve-variablen
  - At køre en tur ændrer på kilometertallet

- Det gøres gennem metoder
  - Det har vi også allerede set!

# Klasser vs. objekter

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

# Klasser vs. objekter

```
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;
    }
```

# Klasser vs. objekter

```
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;
    }
```

# Klasser vs. objekter

```
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;
    }
```

# Klasser vs. objekter

```
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;
    }
}
```

# Klasser vs. objekter

- Bemærk at vi stadig ikke har noget færdigt program
  - Vi har kun en abstrakt beskrivelse (klasse) af en bil
  - Vi har ingen konkrete biler (objekter)
- 3. spørgsmål: Hvordan bruger vi Car-klassen?
  - Hvordan laver vi konkrete biler, Car-objekter?
- Vi laver et hovedprogram (main-metode) der anvender Car-klassen

# Klasser og objekter

```
public class CarMaker {
    public static void main( String[] args ) {
        Car audi;
        Car skoda;
        int mileage;
```

# Klasser og objekter

```
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" );
```

# Klasser og objekter

```
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 );
```

# Klasser og objekter

```
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 );
    }
}
```

# Lidt terminologi

- `Car` er en klasse, dvs. en beskrivelse, i dette tilfælde af en bil
- `skoda` og `audi` er to instanser af klassen `Car`
  - `skoda` og `audi` er objekter
  - Objekter er instanser af en klasse
- Bemærk at der ingen instanser er af klassen `CarMaker`!
  - `CarMaker` indeholder kun hovedprogrammet (main-metoden)
- `getMileage`, `drive` og `setColor` er metoder på klassen `Car`
  - Lidt misvisende, da de faktisk er metoder der virker på objekter

# Lidt filosofi

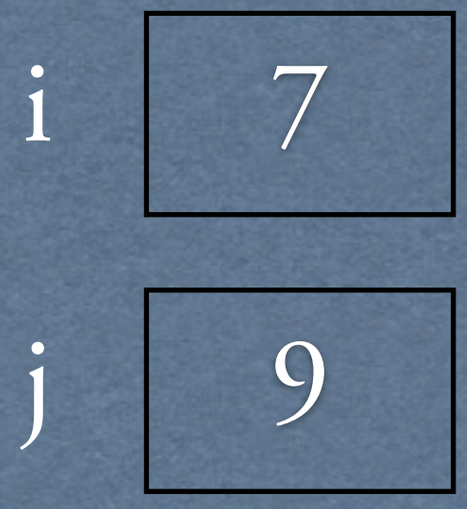
- Klasser er en måde at gruppere data (variable) og metoder på data sammen
- Man bør kun sætte logiske ting sammen
  - Fx giver en klasse med de to variable `bilFarve` og `cykelFarve` næppe mening
  - Lav hellere en klasse `Transportmiddel`

# Advarsel!

# Advarsel!

- Quiz

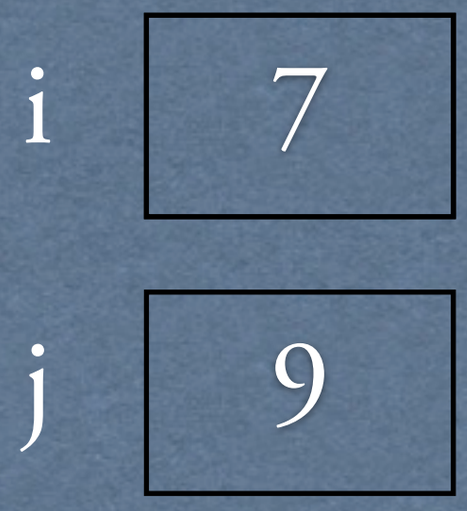
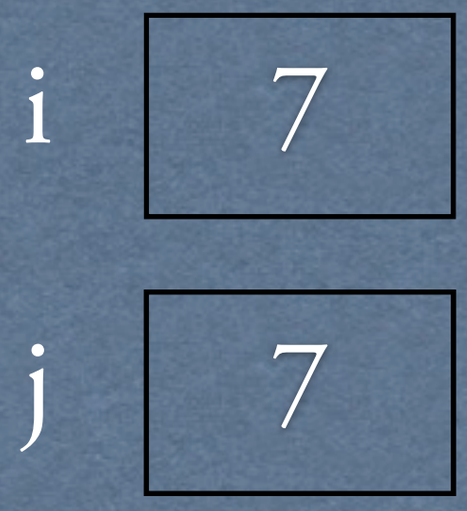
# Advarsel!

- Quiz
  - ```
    int i = 7;
    int j = 9;
    j = i;
    i = 13;
    System.out.println( j ); // Output?
    ```
    ```

# Advarsel!

- Quiz
  - ```
    int i = 7;
    int j = 9;
    j = i;
    i = 13;
    System.out.println( j ); // Output?
    ```
  - ```
    Car bil1 = new Car( “Skoda Fabia”, “rød” );
    Car bil2 = new Car( “Audi A4”, “sølvgrå” );
    bil2 = bil1;
    bil1.drive( 30 );
    System.out.println( bil2.getMileage() ); // Output?
    ```

# Advarsel!

- I det første tilfælde udskrives 7
  - De to variable `i` og `j` er "uafhænige"
- I det andet tilfælde udskrives 30
  - `bil1` og `bil2` afhænger af hinanden
  - Når `bil1` kører, så kører `bil2` også

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- **`int i = 7;`**

```
int j = 9;
j = i;
i = 13;
System.out.println( j ); // Output?
```

## Hukommelse

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  int i = 7;
  int j = 9;
  j = i;
  i = 13;
  System.out.println( j ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  int i = 7;
  int j = 9;
  j = i;
  i = 13;
  System.out.println( j ); // Output?
  ```

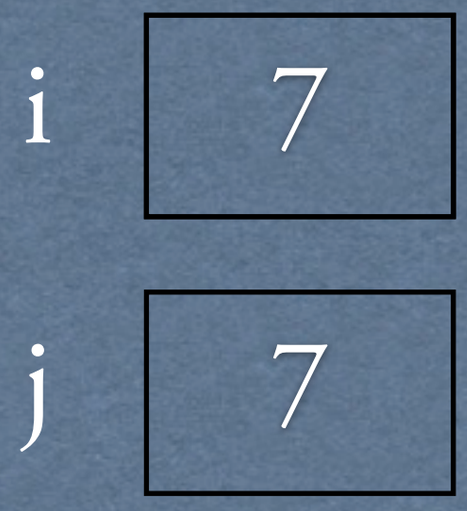
## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  int i = 7;
  int j = 9;
  j = i;
  i = 13;
  System.out.println( j ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  int i = 7;
  int j = 9;
  j = i;
  i = 13;
  System.out.println( j ); // Output?
  ```

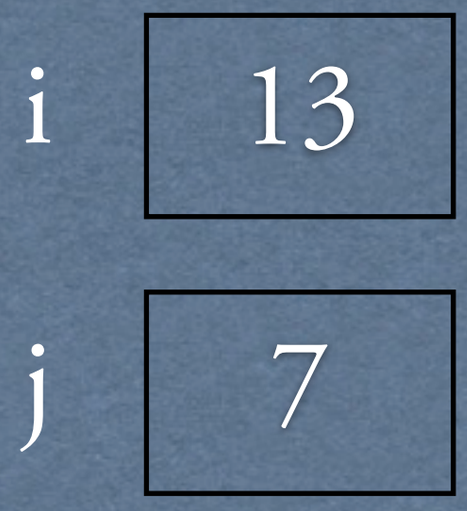
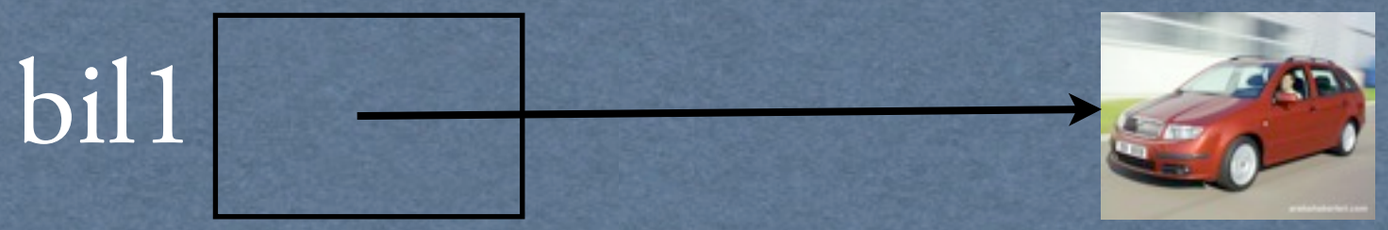
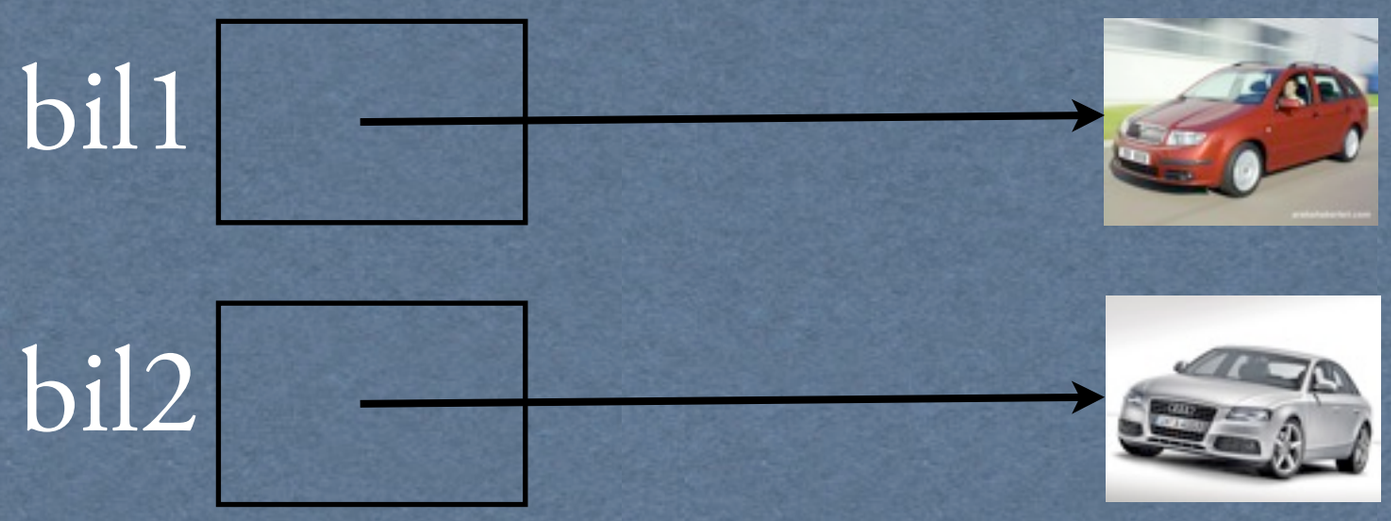
## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  int i = 7;
  int j = 9;
  j = i;
  i = 13;
  System.out.println( j ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  int i = 7;
  int j = 9;
  j = i;
  i = 13;
  ```
  **`System.out.println( j ); // Output?`**

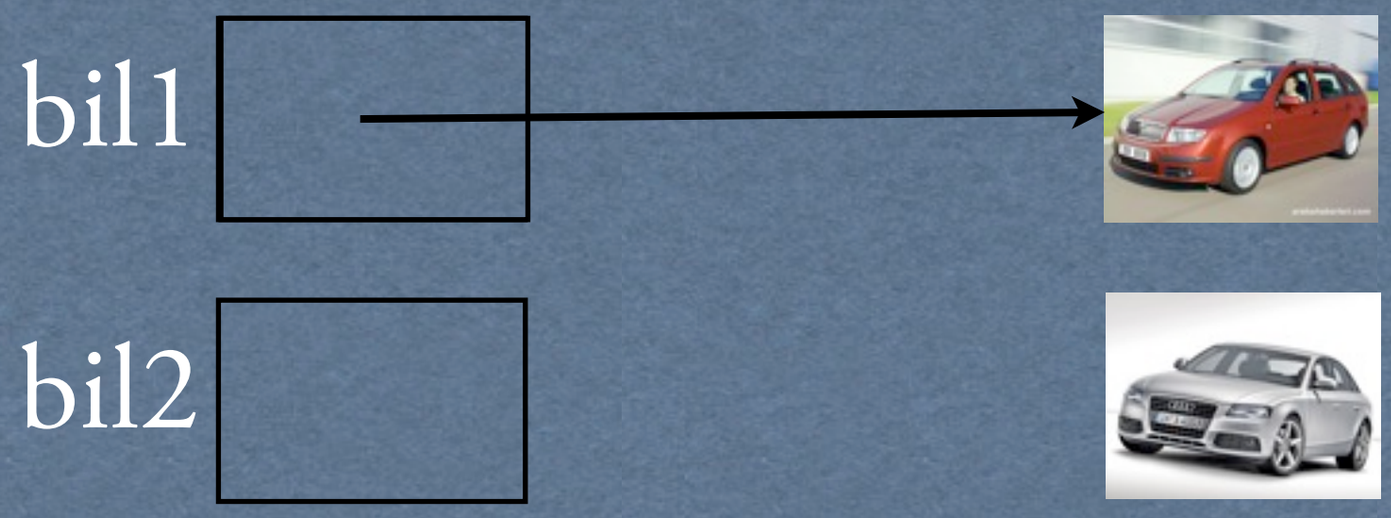
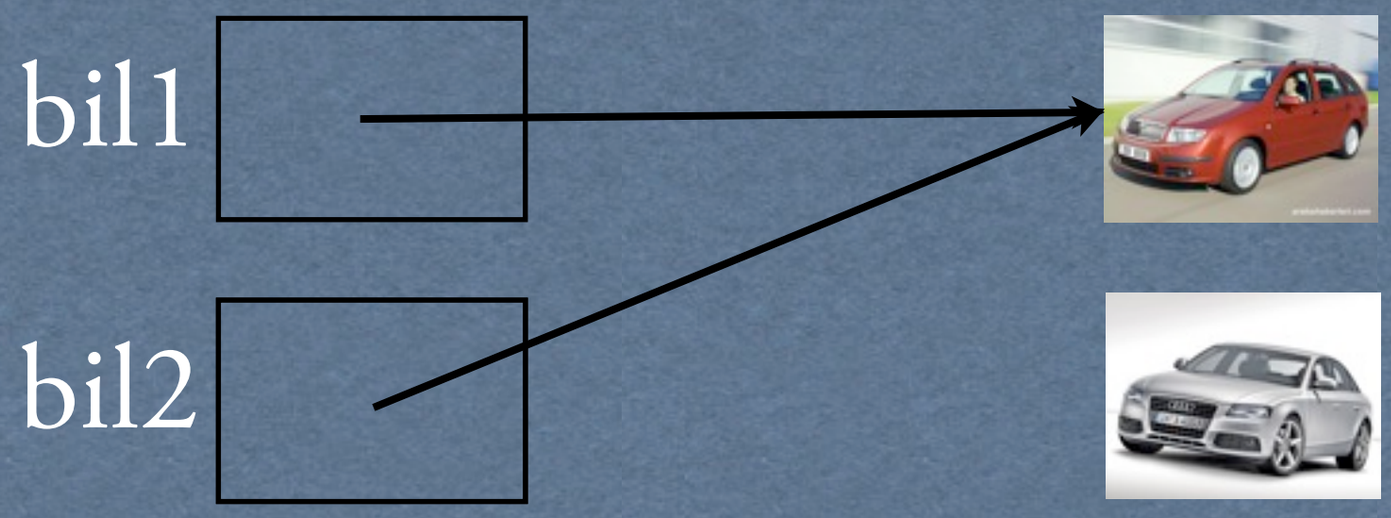
## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- **`Car bil1 = new Car( “Skoda Fabia”, “rød” );`**

```
Car bil2 = new Car( “Audi A4”, “sølvgrå” );
bil2 = bil1;
bil1.drive( 30 );
System.out.println( bil2.getMileage() ); // Output?
```

## Hukommelse

bil1

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- **`Car bil1 = new Car( “Skoda Fabia”, “rød” );`**

```
Car bil2 = new Car( “Audi A4”, “sølvgrå” );
bil2 = bil1;
bil1.drive( 30 );
System.out.println( bil2.getMileage() ); // Output?
```

## Hukommelse



# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

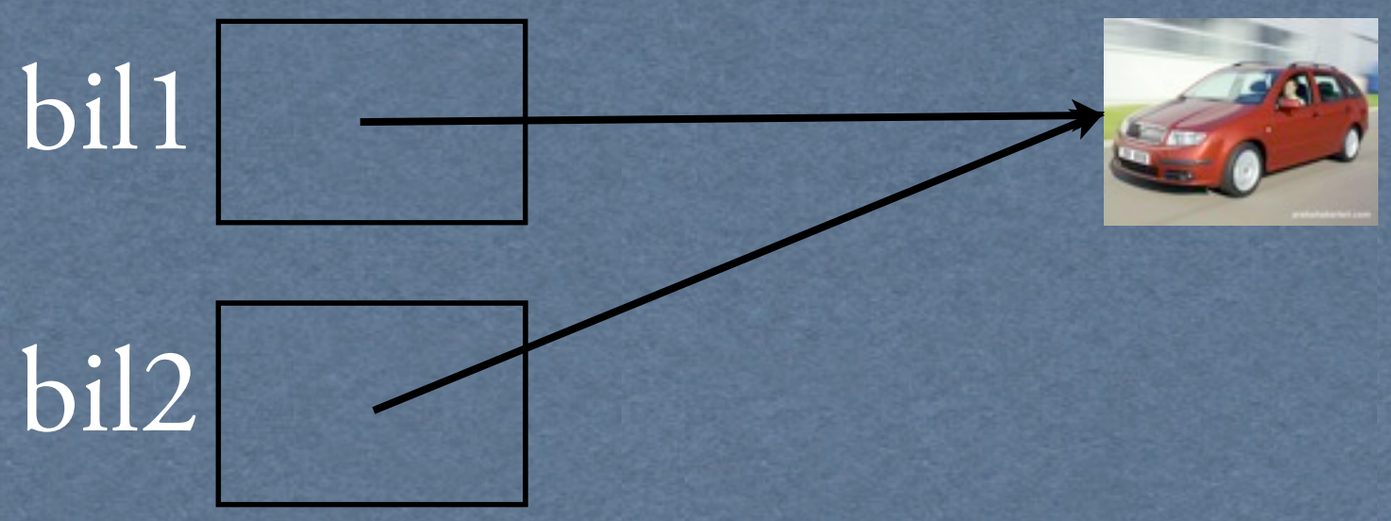
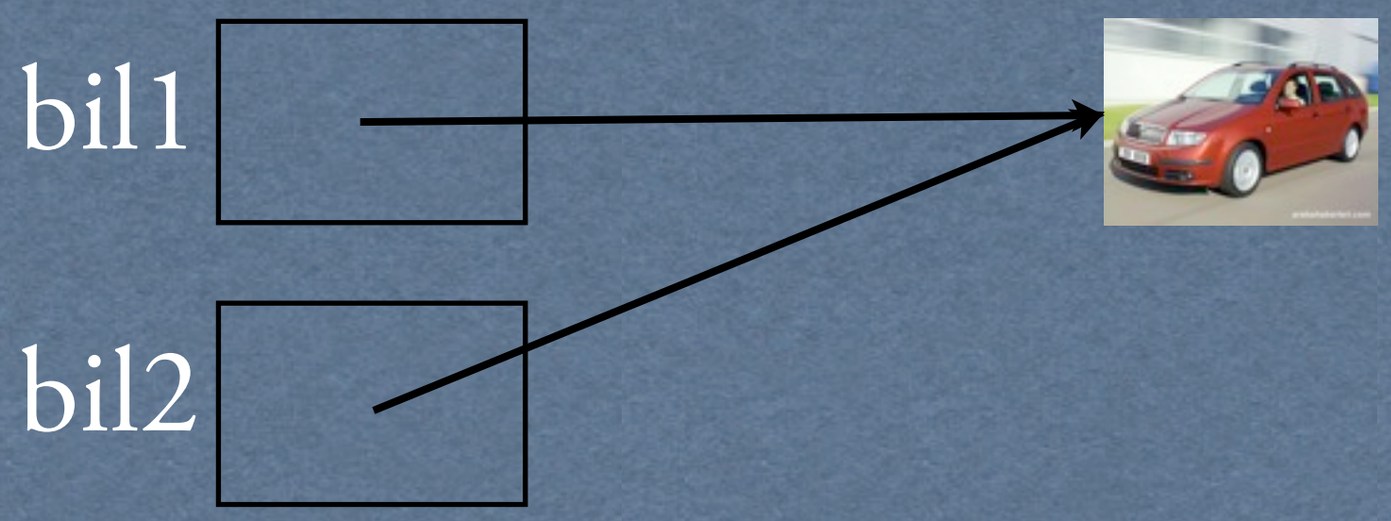
## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

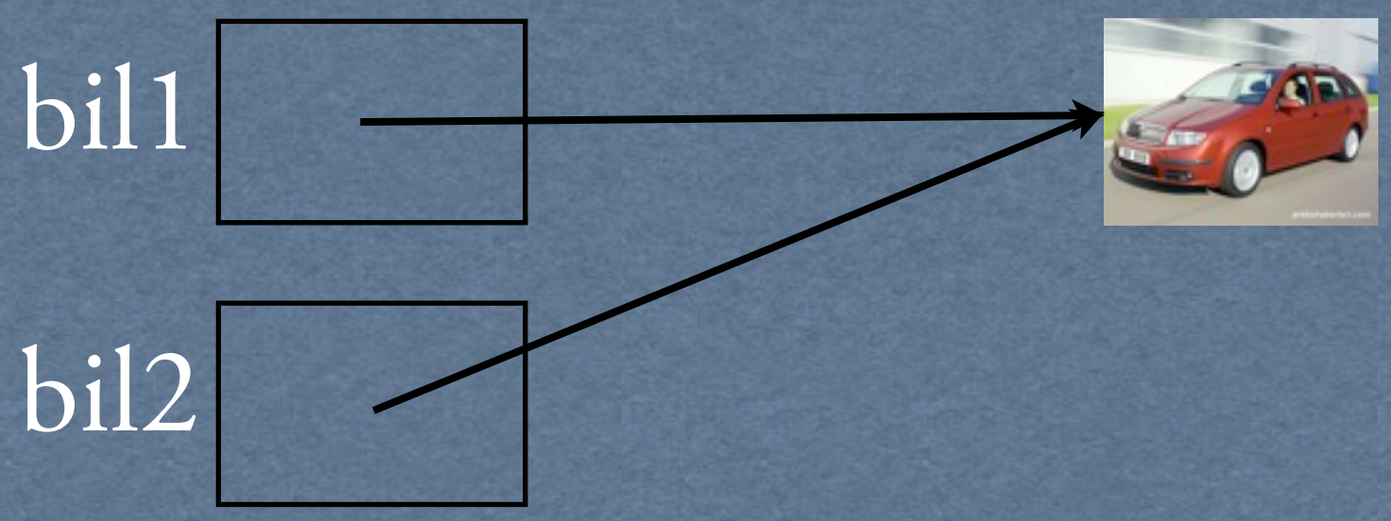
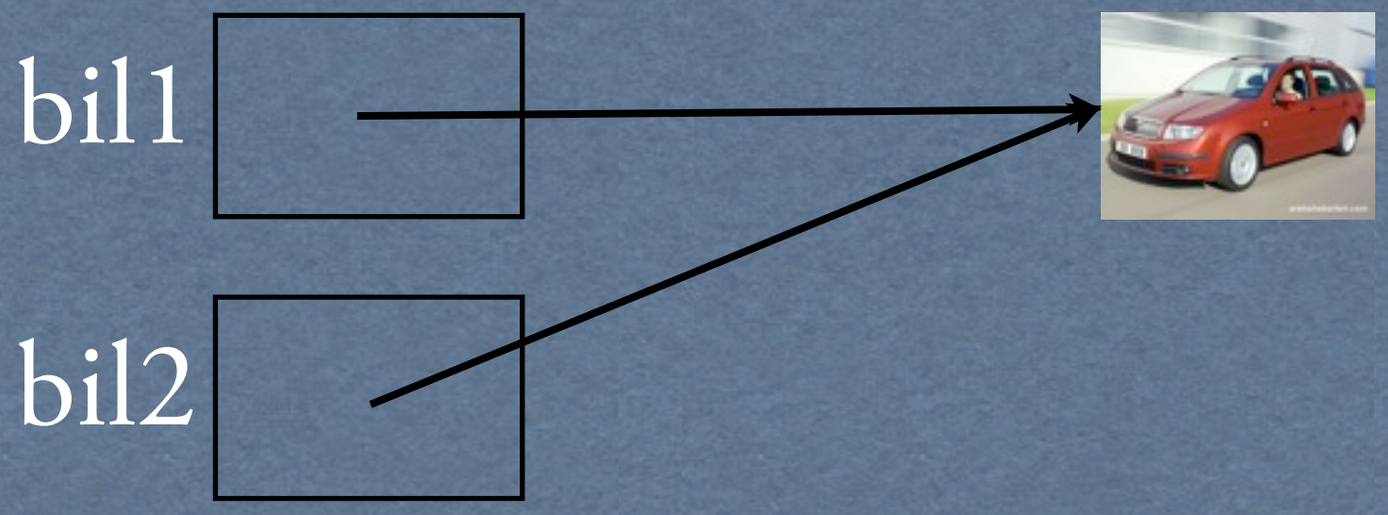
## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

## Hukommelse


  ```

# Advarsel!

- Fundamentale typer og objekter er gemt forskelligt
- ```
  Car bil1 = new Car( "Skoda Fabia", "rød" );
  Car bil2 = new Car( "Audi A4", "sølvgrå" );
  bil2 = bil1;
  bil1.drive( 30 );
  System.out.println( bil2.getMileage() ); // Output?
  ```

## Hukommelse


  ```

# Math-klassen

- Matematiske operationer `+, -, *, /, %, ...`
- Andre operationer i klassen `java.lang.Math;`
- Indeholder metoder til matematiske operationer
- Mest brugte
  - `Math.E` og `Math.PI` - konstanter
  - `Math.exp( x )` - $e^x$
  - `Math.log( x )` - $\ln(x)$
  - `Math.sin(x), Math.cos(x), Math.tan(x)`
  - Osv...