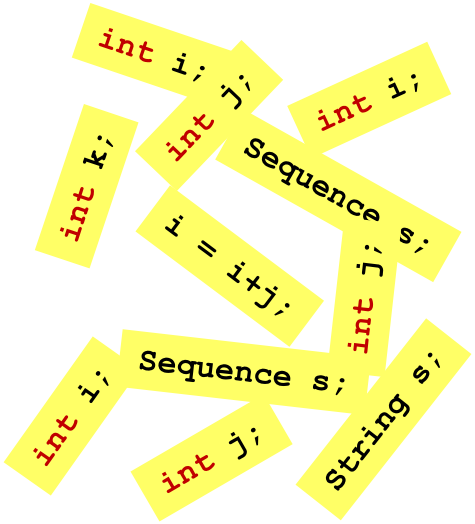
Forelæsning Uge 4 – Torsdag

- **Algoritmeskabeloner**
  - findOne, findAll, findNoOf, findSumOf (i mandags)
  - findBest
- **Levetid for variabler**
  - Hvor længe eksisterer de forskellige variabler?
  - Hvor længe har de en værdi?
- **Virkefeltsregler**
  - Hvor kan man bruge de forskellige variabler?



Algoritmeskabeloner

Lidt repetition fra i mandags

Køreprøven indeholder opgaver, som kan løses ved hjælp af algoritmeskabeloner

Returnerer ét element med den angivne egenskab

```
public TYPE findOne(PARAM) {
  for(TYPE elem : LIST) {
    if(TEST(elem,PARAM)) {
      return elem;
    }
  }
  return null;
}
```

Når vi finder et element, returneres elementet (og algoritmen terminerer)

Returnerer alle elementer med den angivne egenskab (i en arrayliste)

```
public ArrayList<TYPE> findAll(PARAM) {
  ArrayList<TYPE> result = new ArrayList<>();
  for(TYPE elem : LIST) {
    if(TEST(elem,PARAM)) {
      result.add(elem);
    }
  }
  return result;
}
```

Når vi finder et element, tilføjes det til den lokale variabel result, som er en arrayliste

Algoritmeskabeloner

Returnerer antallet af elementer med den angivne egenskab

```
public int findNoOf(PARAM) {
  int result = 0;
  for(TYPE elem : LIST) {
    if(TEST(elem,PARAM)) {
      result++;
    }
  }
  return result;
}
```

Når vi finder et element, tælles den lokale variabel result op

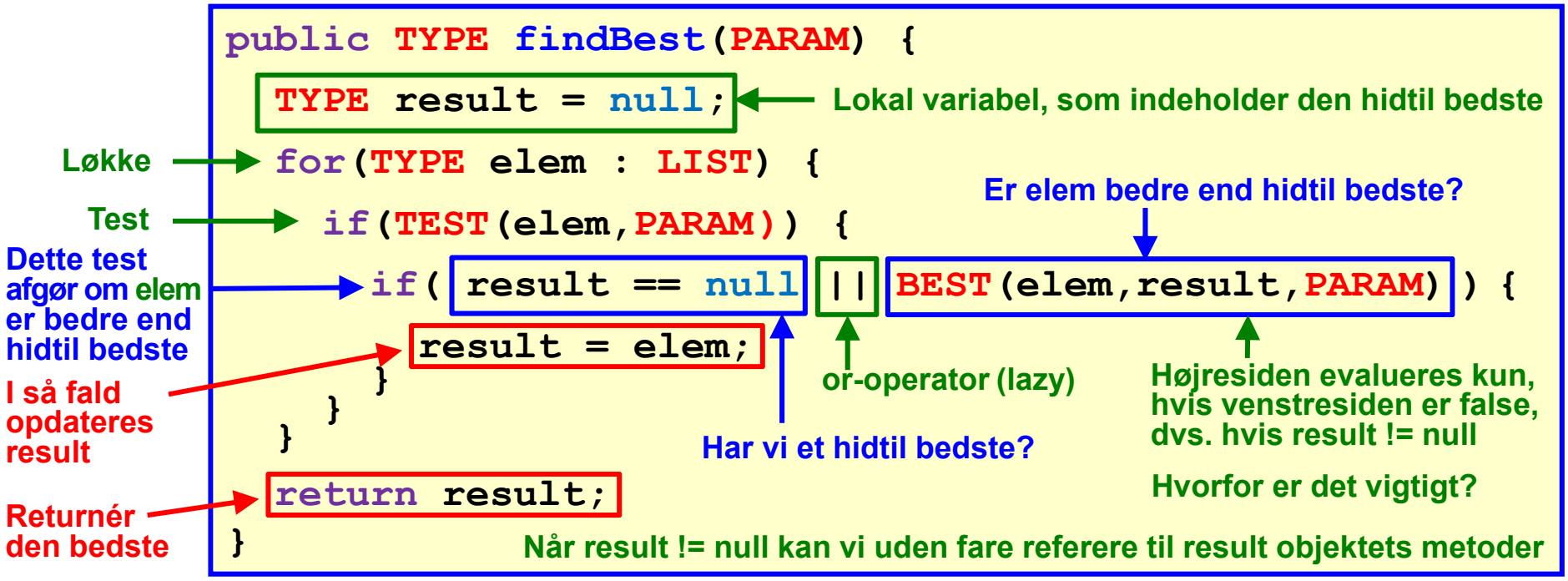
Returnerer summen af de elementer, der har den angivne egenskab

```
public int findSumOf(PARAM) {
  int result = 0;
  for(TYPE elem : LIST) {
    if(TEST(elem,PARAM)) {
      result += VALUE(elem,PARAM);
    }
  }
  return result;
}
```

Når vi finder et element, adderes værdien af elementet til den lokale variable result

Algoritmeskabelonen findBest

- **Gennemsøger arraylisten LIST med elementer af typen TYPE og returnerer det bedste af de elementer, der opfylder TEST**



- Hvis flere elementer er lige gode, returneres det først fundne
- Hvis ingen elementer opfylder **TEST**, returneres **null**
- Hvis man undlader **TEST** (og fjerner den yderste if sætning), finder man det **BEDSTE** af alle elementer i **LIST**

Gentag!

Eksempler på findBest

```
public Pixel findBrightestDarkPixel(int color) {
  Pixel result = null;
  for(Pixel p : pixels) {
    if(p.getColor() <= color) {
      if(result == null ||
            p.getColor() > result.getColor()) {
        result = p;
      }
    }
  }
  return result;
}
```

Finder den lyseste mørke pixel

```
public Person findOldestContaining(String q) {
  Person result = null;
  for (Person p : persons) {
    if(p.getName().contains(q)) {
      if(result == null ||
            p.getAge() > result.getAge()) {
        result = p;
      }
    }
  }
  return result;
}
```

- Husk at result == null skal stå til venstre for ||
- Ellers får vi en runtime fejl

Finder den ældste person, hvis navn indeholder den angivne tekststreng

Sammenligning af algoritmeskabelonerne

- **Alle skabeloner gennemsøger en arrayliste (eller en anden objektsamling)**
  - Hvert enkelt element i listen tjekkes op mod en **angiven betingelse**
  - Betingelsen involverer **kun det element** i listen, der pt. undersøges
- **Forskelle**

  - Skobutik med en arrayliste med sko
  - Betingelsen tester skoenes farve

  - findOne returnerer **ét** element, der opfylder den angivne betingelse (og stopper så snart et sådant element er fundet) ÉN RØD SKO
  - findAll returnerer en arrayliste med **alle** de elementer, der opfylder den angivne betingelse ALLE RØDE SKO
  - findNoOf returnerer **antallet** af elementer, der opfylder den angivne betingelse ANTALLET AF RØDE SKO
  - findSumOf returnerer **summen** af værdierne af de elementer, der opfylder den angivne betingelse SAMLEDE PRIS FOR ALLE RØDE SKO
  - findBest returnerer det **bedste** af de elementer, der opfylder den angivne betingelse (sammenligner elementer) BILLIGSTE RØDE SKO

Skabelon	Lokal variabel	Initialisering	Opdatering
`findAll`	`arrayliste`	`tom liste`	`add`
`findNoOf`	`heltal`	`0`	`+= 1`
`findSumOf`	`heltal`	`0`	`+= VALUE`
`findBest`	`element`	`null`	`hidtil bedste`

Køreprøven indeholder opgaver, som kan løses ved hjælp af algoritmeskabeloner

Quiz

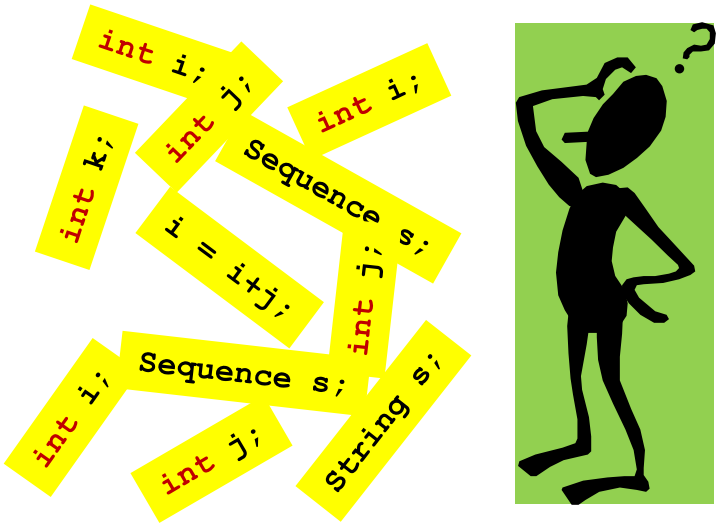
Levetid for variabler og parametre

- **Feltvariabler**
  - modellerer tilstand for objekter
  - levetiden er den samme som objektets
- **Klassevariabler**
  - modellerer tilstand for klasser
  - levetiden er hele programudførelsen
- **Lokale hjælpevariabler**
  - defineres i en metode/konstruktør
  - levetiden er varigheden af metode/konstruktør kaldet
- **Parametre**
  - lokale variabler hvor startværdien leveres af kalderen
  - levetiden er varigheden af metode/konstruktør kaldet
- **Kontrolvariabler**
  - variabler i ”hovedet” af en for eller for-each løkke
  - levetiden er varigheden af udførelsen af løkken

Virkefeltsregler (fortolkning af navne)

- **Et navn fortolkes i en kontekst, som er med til at definere navnets betydning**
  - Beskeden “Ring til Kirsten og sig at ...” kan betyde to helt forskellige ting på arbejde og hjemme (kollega / svigermor)
  - Sætningen i++ kan opdatere forskellige variabler – afhængig af, hvor i programmet den står

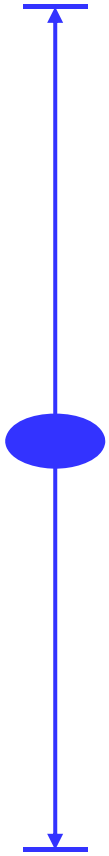
- **Java (og andre programmeringssprog) har præcise, utvetydige regler for fortolkning af navne**
  - Dem skal vi lære om nu



Erklæringer i Java

- **For at bruge et navn skal det erklæres eller importeres**
  - Navne fra java.lang pakken importeres dog automatisk
    - Klasserne String, Math og System
    - Wrapper klasserne for de primitive typer (Integer, Double, …)
    - En masse andet…(se i Java API)
- **Variabler kan erklæres på forskellig vis**
  - Feltvariabler og klassevariabler tilhører en klasse og kan bruges overalt i klassen
  - De kan kun bruges uden for klassen, hvis de er public, hvilket feltvariabler aldrig bør være (mens det kan være ok for klassevariabler)
  - Lokale hjælpevariabler og parametre tilhører en metode eller en konstruktør og kan **aldrig** bruges uden for denne
  - Kontrolvariabler tilhører en løkke og kan **aldrig** bruges uden for denne
- **Bemærk at private feltvariable også kan bruges/tilgås i et andet objekt fra samme klasse**
  - Hvis et **Person** objekt har en reference (f.eks. feltvariablen **farther**) til et andet **Person** objekt, kan vi skrive **farther.name** i stedet for **farther.getName()**

Feltvariabler og klassevariabler



```
public class Scope {
  public Scope() {
    i = 0;
  }

  public void addTwo() {
    addOne(); addOne();
  }

  private int i;

  public void addOne() {
    i = i + 1;
  }

  public int getValue() {
    return i;
  }
}
```

Style guide: Feltvariabler og klassevariabler bør erklæres i begyndelsen af klassen

- **Regel i Java Style Guide**
- **Oversætteren er ligeglad**

Alle feltvariabler/klassevariabler er tilgængelige overalt i klassen

Metoder og konstruktører

```
public class Scope {
  public Scope() {
    i = 0;
  }

  public void addTwo() {
    addOne(); addOne();
  }

  private int i;

  public void addOne() {
    i = i + 1;
  }

  public int getValue() {
    return i;
  }
}
```

Alle metoder/konstruktører er tilgængelige overalt i klassen

- Samme regel gælder for klassemetoder

Parametre

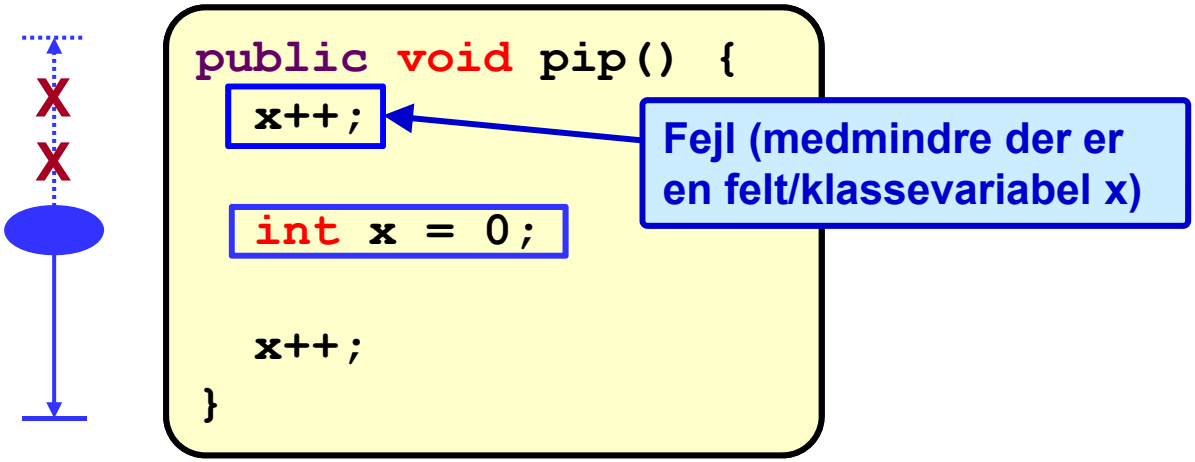
```
public void addDays( int d ) {

  for( int i = 0; i < d; i++ ) {
    setToNextDate();
  }
}
```

Parametre til en metode er tilgængelige overalt i metoden, men aldrig uden for metoden

- Samme regel gælder for parametre til en konstruktør

Andre lokale variabler



Lokale variabler erklæret i en blok er tilgængelige fra og med erklæringen og indtil blokkens afslutning

Pause

Indre blokke

En blok kan have indre blokke { ... { ... } ... }

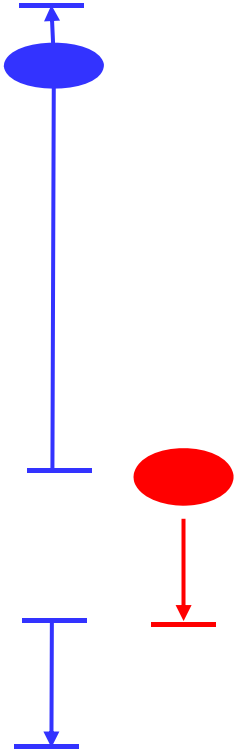
```
public void pip() {
   int i = 0;
   {
      i++;
      System.out.println(i);
      int x = 0;
      x = i;
      x++;
      i++;
      System.out.println(x);
   }
   System.out.println(i);
}
```

Samme regel: lokale variabler er tilgængelig fra og med erklæringen og indtil afslutningen af den blok, hvori de er erklæret

Et navn virker også inde i indre blokke

```
public void pip() {
   ...
   int x = 0;
   ...
   {
      i++;
      int j = 0;
      System.out.println(i);
      x++;
      System.out.println(x);
   }
   ...
   ...
}
```

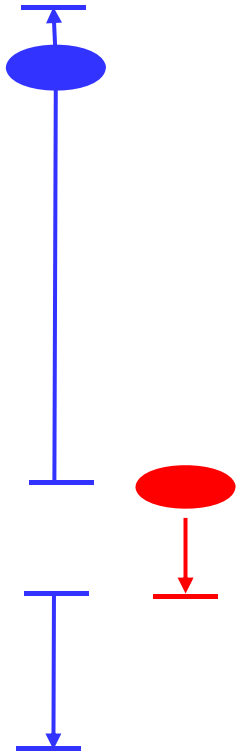
Et variabel kan “skygge” for et anden med samme navn



```
public class Scope {
  private int i;
   public Scope() {
     i = 0;
   }
   public void pip() {
     i++;
      {
        i++; System.out.println("a" + i);
        int i = 0;
         this.i++
        i++; System.out.println("b" + i);
      }
   }
}
```

- De to røde i'er referer til den lokale variabel (den røde erklæring)
- De fire blå i'er referer til feltvariablen (den blå erklæring)

Brug af this



```
public class Scope {
  private int i;
  public Scope() {
    i = 0;
  }
  public void pip() {
    i++;
    {
      i++; System.out.println("a" + i );
      int i = 0;
      this.i++
      i++; System.out.println("b" + i );
    }
  }
}
```

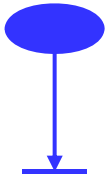
- this refererer til objektet selv
- this.i refererer altid til feltvariablen (selv om der er andre i'er, der skygger)

Bruges ofte i konstruktører

- Tillader at konstruktørens parametre har samme navne, som de feltvariabler de skal bruges til at initialisere
- Vi kan f.eks. skrive this.name = name; hvor venstresiden af assignmentet referer til en feltvariabel, mens højresiden referer til en parameter

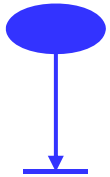
Kontrolvariabler i for / for-each løkke

for løkke



```
for(int j=0 ; j<4; j++) {
  System.out.println(j);
}
```

for-each løkke



```
for(Person p : persons) {
  System.out.println(p);
}
```

Kontrolvariabler erklæret i hovedet af en for / for-each løkke er tilgængelige overalt i løkken (inklusiv hovedet), men aldrig uden for løkken

Virkefeltsregler i Java (opsummering)

- **Vi har følgende virkefeltsregler**
  - En **feltvariabel/klassevariabel** (erklæret i en klasse) virker overalt i klassen med undtagelse af de steder, hvor den overskygges (af en parameter, lokal hjælpevariabel eller kontrolvariabel)
  - En **parameter** (erklæret i hovedet af en metode/konstruktør) virker overalt i metodens/konstruktørens krop med undtagelse af de steder, hvor den overskygges af en lokal hjælpevariabel eller kontrolvariabel
  - En **lokal variabel** (erklæret i en blok) virker fra erklæringsstedet og indtil blokkens afslutning med undtagelse af de steder, hvor den overskygges af en anden lokal hjælpevariabel eller kontrolvariabel
  - En **kontrolvariabel** (erklæret i hovedet af en for / for-each løkke) virker overalt i løkken (inklusiv hovedet) med undtagelse af de steder, hvor den overskygges af en lokal hjælpevariabel eller anden kontrolvariabel

Brug i andre klasser

- **Virkefeltsreglerne beskriver, hvor konstruktører, metoder og variabler kan bruges/tilgås inden for deres egen klasse**
  - Nogle af dem kan også bruges i andre klasser – det skal vi nu se på
- **Brug i andre klasser**
  - **Feltvariabler** kan aldrig tilgås fra andre klasser, idet de bør være private
  - **Konstruktører og klassevariabler/klassemetoder** kan tilgås fra andre klasser, hvis de er public
  - **Instansmetoder** kan tilgås fra andre klasser, hvis de er public og man har en reference til et objekt af den klasse, hvori metoden er erklæret
  - **Parametre, lokale hjælpevariabler og kontrolvariabler** kan aldrig tilgås fra andre klasser, hvorfor de ikke har en access modifier
- **Når en konstruktør, variabel eller metode kan tilgås i en anden klasse, kan den tilgås overalt i denne**

Hvilken variabel?

- **Antagelse:** Feltvariabler/klassevariabler er erklæret øverst i klassen (jvf. Style Guiden)
- **Regel:** Søg opad indtil en erklæring nås – gå ikke ind i de blokke, løkker, metoder og konstruktører, som du passerer undervejs

```
public class Scope {
  private int i = 0;
  public Scope ( int i ) {
    i = i + 1;
    this.i++
    System.out.println("a" + i );
  }
  public void pip() {
    System.out.println("b" + i );
    for( int i = 0; i < 3; i ++){
      System.out.println("c" + i );
    }
    System.out.println("d" + i );
  }
  ...
}
```

Quiz

Opsummering

- **Algoritmeskabeloner**
  - findOne, findAll, findNoOf, findSumOf (sidste mandag)
  - findBest (sammenligner på tværs af elementer)
- **Levetid for variabler**
  - Hvor længe eksisterer de forskellige variabler?
  - Hvor længe har de en værdi?
- **Virkefeltsregler**
  - Hvor kan man bruge de forskellige variabler?

Køreprøven indeholder opgaver, som kan løses ved hjælp af algoritmeskabeloner

Husk at se videoerne om køreprøvesættene Phone og Pirate før øvelserne i uge 5
- **Prøv dernæst selv at løse opgaverne (se videoerne igen, hvis det kniber)**
- **Opgave 10 bruger Comparable interfacet, som I først lærer om i næste uge (så dem kan I vente med at se)**

Husk at aflevere
- **Quiz 4 (alene)**
- **Skildpadde 2 (par)**
- **Billedredigering (par)**
- **Eventuelle genafleveringer fra tidligere uger**

Det var alt for nu….. … spørgsmål