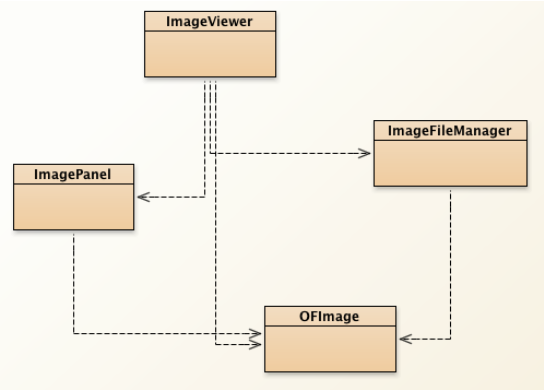
Grafiske Brugergrænseflader

--- GUI (Graphical User Interface) ---

<u>GRPRO</u>: "Grundlæggende Programmering"

A G E N D A

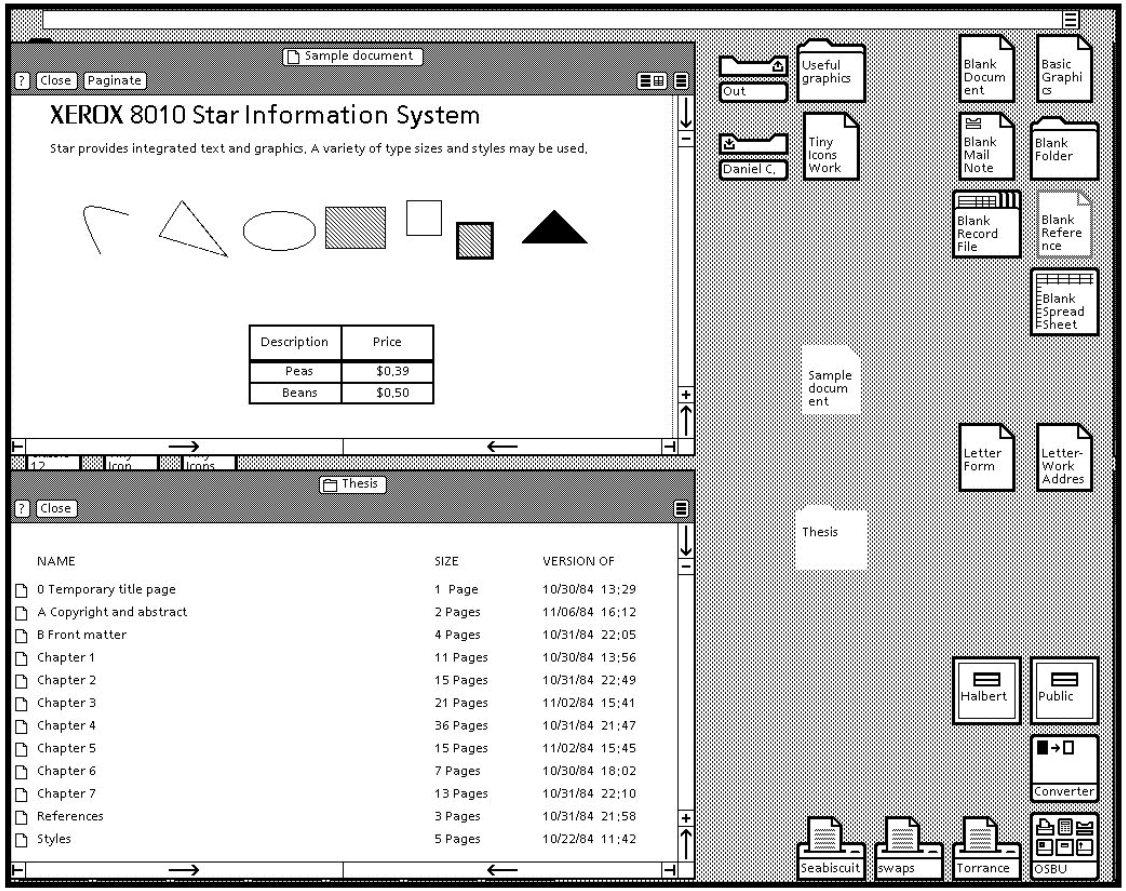
- Grafiske brugergrænseflader
- Components
- Event Handling
  - Anonymous inner classes
- Example: "ImageViewer"



- Graphics i Java

GUI Historie

- Doug Engelbart, SRI (1960s)
  - Computermus
- Xerox Palo Alto Research Center (1970s)
  - Vinduer, menuer, knapper, ikoner, ...
- Xerox Star (1981)
- Apple Macintosh (1984)
- Unix X Windows (1985)
- MS Windows 3.1 (1990)



Grafiske brugergrænseflader

- **Komponenter:**
  - Knapper, tekstfelter, checkboxes, radio buttons, scrollbarer, menuer og menupunkter, ...
- **Containere:**
  - Frames (vinduer), paneler, ...
- **Layout:**
  - Styrer hvor komponenter vises på den omgivende container
- **Event handlers:**
  - Håndterer hændelser: det som brugeren gør ved komponenter: tryk på knap, vælg menupunkt, ...

Vigtige pakker

`java.awt`:

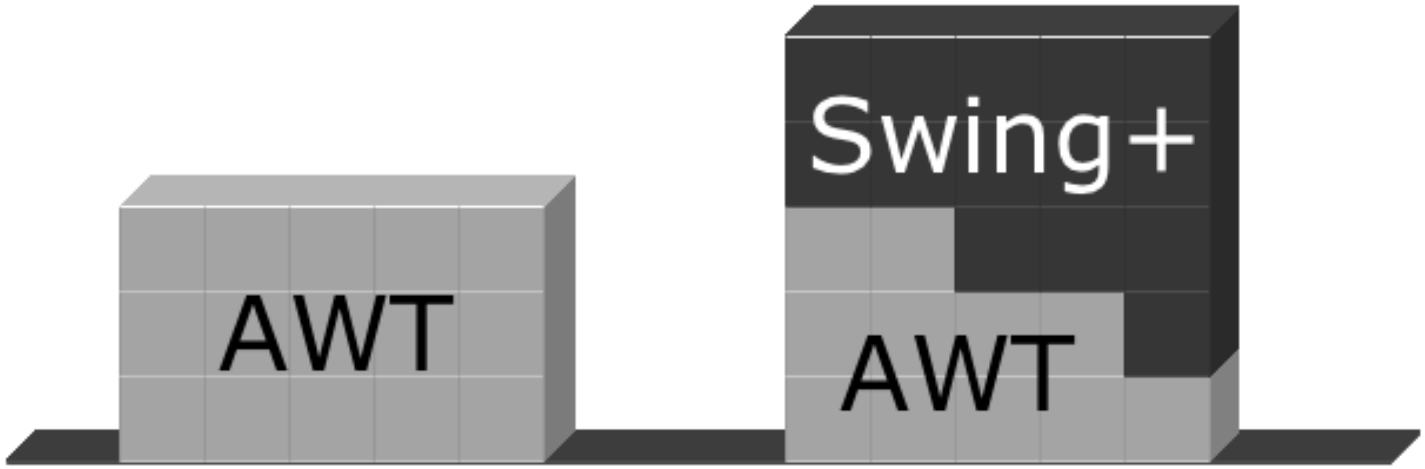
- AWT (Abstract Window Toolkit)

`java.awt.event`:

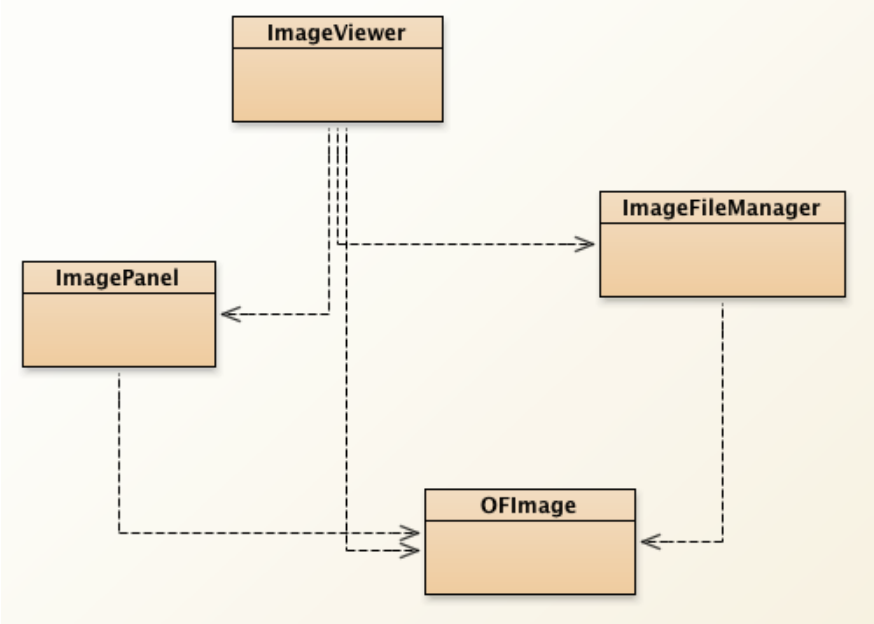
- Event handling

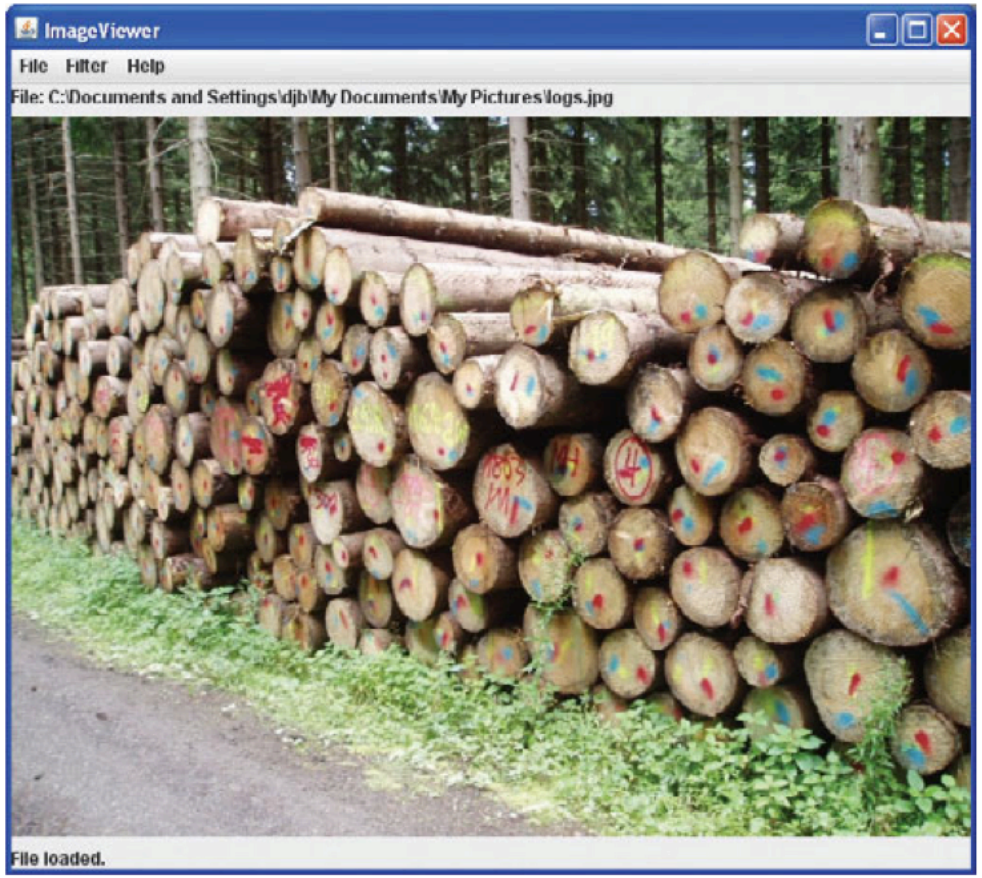
`javax.swing`:

- Swing (bygger på AWT)



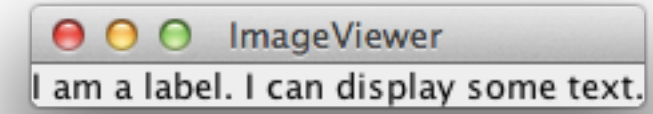
ImageViewer v 1.0 [B&K, kap 11]





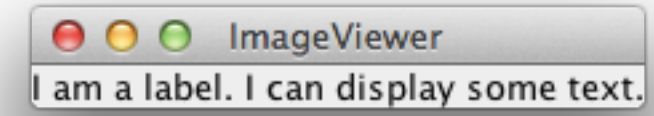
- **ImageViewer:**
  - Open and display images
  - Simple transformations (lighten, darken, resize, ...)

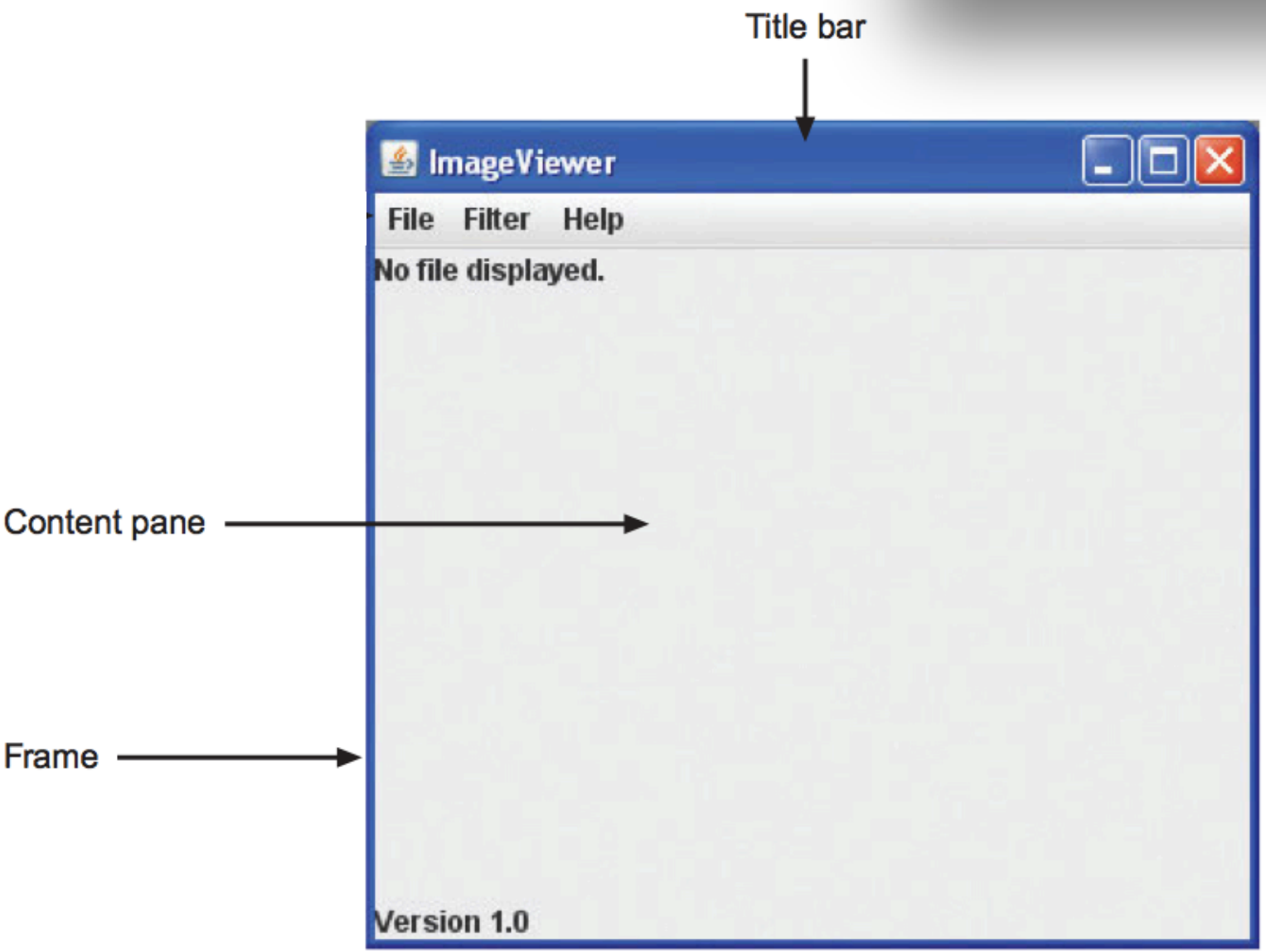
ImageViewer v 0.1 [B&K, kap 11]



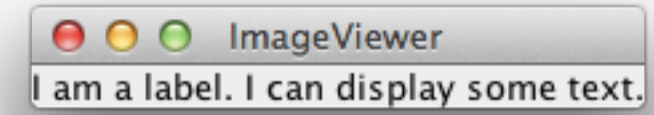
- **ImageViewer:**
  - Just display a simple frame (vindue)
  - No transformations

Bestanddele af en frame





ImageViewer 0.1



- "Hello World" ... for images:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ImageViewer {
   private JFrame frame;

   public ImageViewer() {
      makeFrame();
   }

   private void makeFrame() {
      frame = new JFrame("ImageViewer");
      Container contentPane = frame.getContentPane();

      JLabel label = new JLabel("I am a label. I can display some text.");
      contentPane.add(label);

      frame.pack();
      frame.setVisible(true);
    }
}
```

Standard GUI imports

Opsætning af frame (vindue)

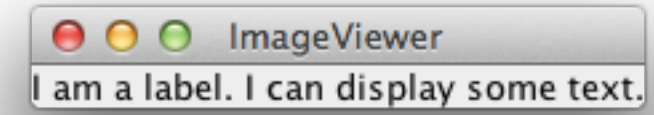
Frame med titel

Container for Frames indhold

Lav Label for text (or img) og sæt på container

Lav layout på frame og gør den synlig

ImageViewer 0.1a

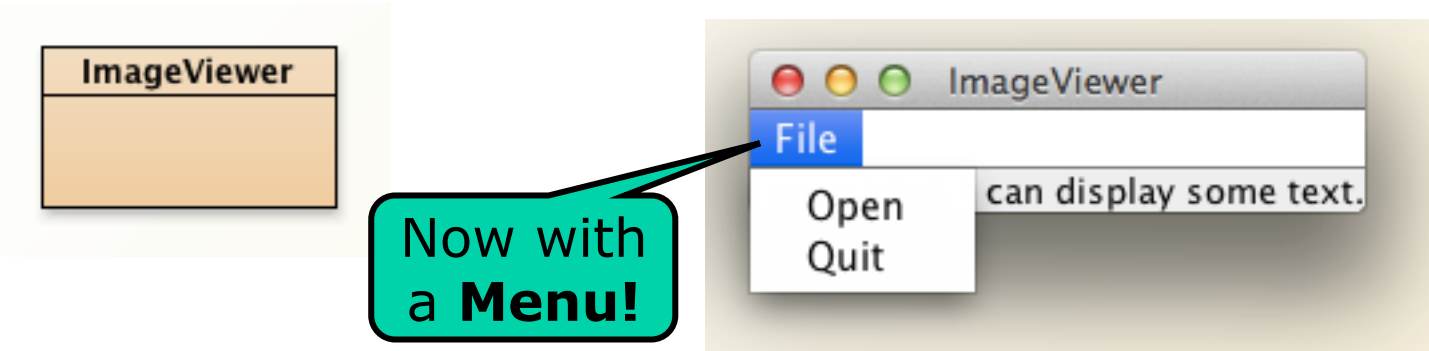


- Alternative version (subclassing JFrame):

Does same thing, but has different structure

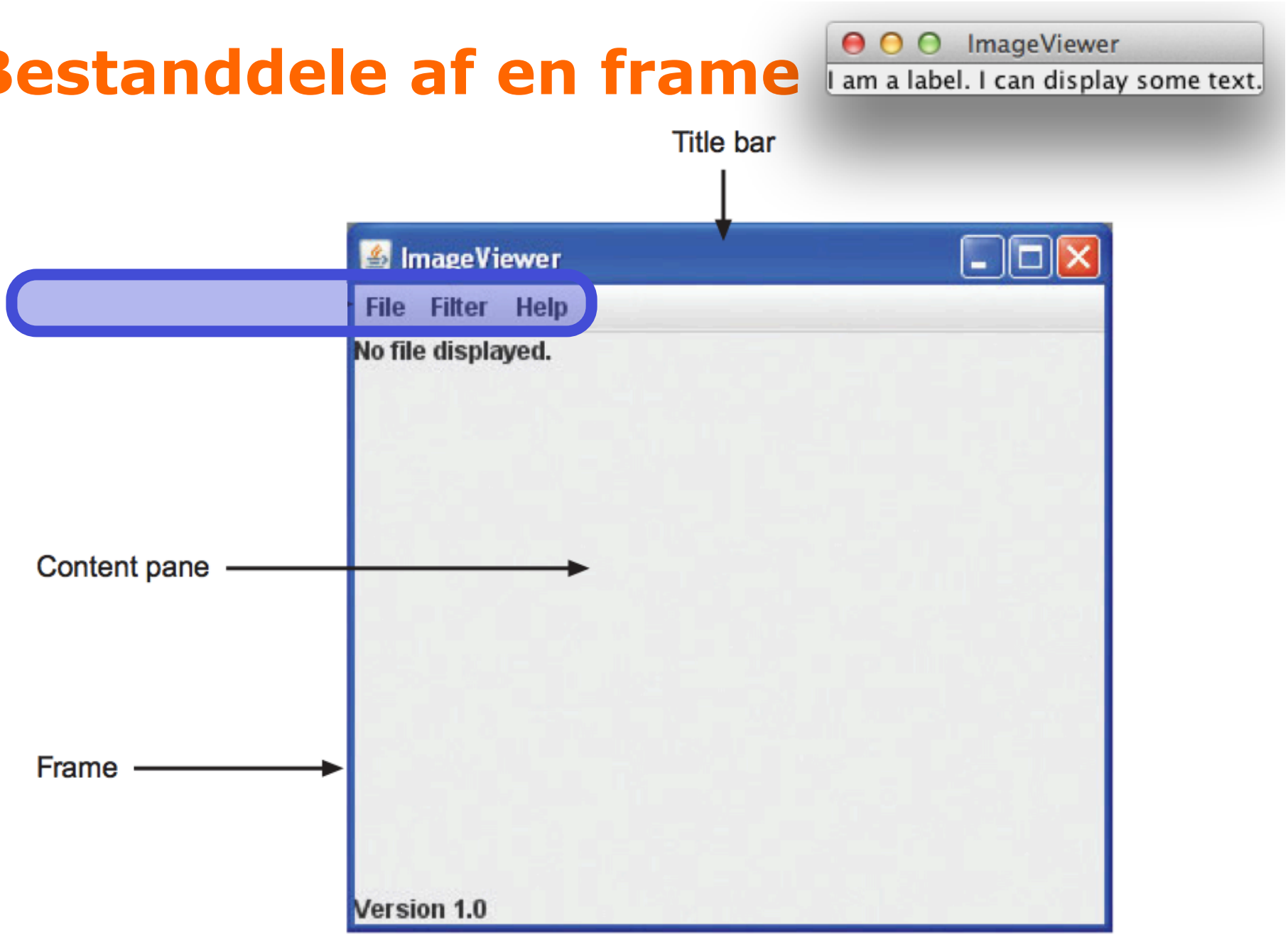
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ImageViewer extends JFrame {
   public ImageViewer() {
      super("ImageViewer");
      makeFrame();
   }

   // ---- swing stuff to build the frame and all its components ----
   private void makeFrame() {
      Container contentPane = getContentPane();

      JLabel label = new JLabel("I am a label. I can display some text.");
      contentPane.add(label);

      pack();
      setVisible(true);
   }
}
```

ImageViewer v 0.2

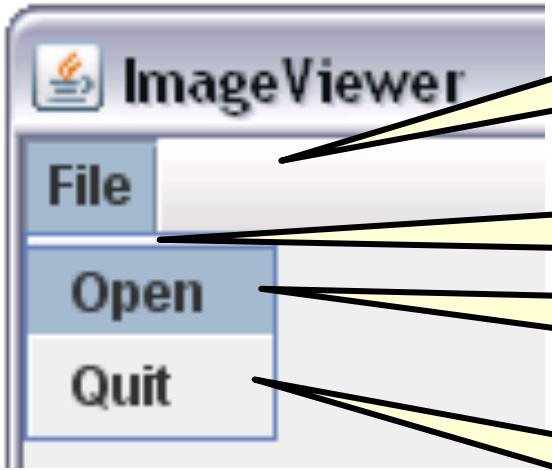


- **ImageViewer:**
  - Just display a simple frame – ...with a **menu!**
  - No transformations

Bestanddele af en frame



ImageViewer v 0.2

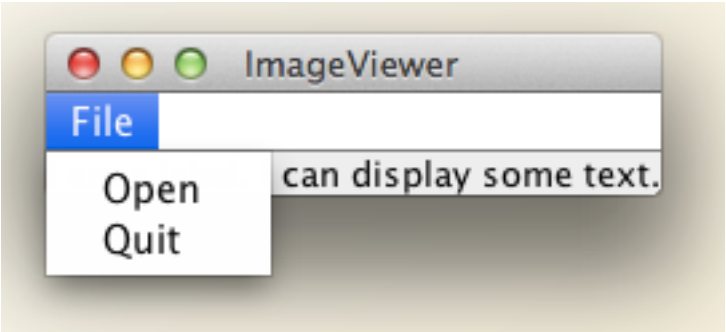



```
JMenuBar menubar = new JMenuBar();
frame.setJMenuBar(menubar);
```

```
JMenu fileMenu = new JMenu("File");
menubar.add(fileMenu);
```

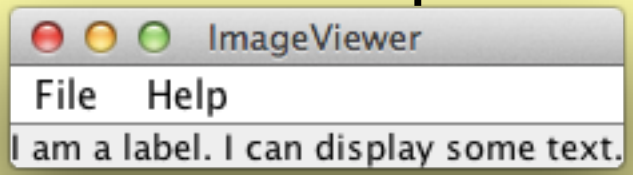
```
JMenuItem openItem = new JMenuItem("Open");
fileMenu.add(openItem);
```

```
JMenuItem quitItem = new JMenuItem("Quit");
fileMenu.add(quitItem);
```



EXERCISE:

- add a "Help" menu



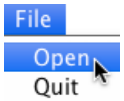
Event Handling

(Hændelsehåndtering)

Event Handling

- **En event** (på dansk: "hændelse") svarer til brugers ***interaktion*** med en GUI-komponent

- Forskellige eventtyper for forskellige komponenttyper:
  - Manipulation af en **`Frame`** giver en **`WindowEvent`**
  - Valg i en **`Menu`** giver en **`ActionEvent`**
  - Klik på musen giver en **`MouseEvent`**



- En event aktiverer en metode i et såkaldt ***"Event Listener"*** objekt, aka:
  - *"event handler"*
  - *"action listener"*
  - *"lytter"* *(på dansk)*

The "ActionListener" Interface

ActionListener (Java Platfor ×

docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html

Overview Package Class Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.awt.event

Interface ActionListener

⋮

```
public interface ActionListener
extends EventListener
```

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's `addActionListener` method. When the action event occurs, that object's `actionPerformed` method is invoked.

Since:

1.1

See Also:

`ActionEvent`, Tutorial: Java 1.1 Event Model

Method Summary

Methods

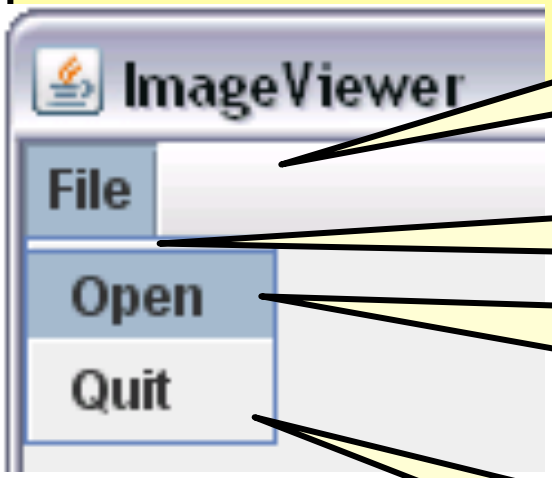
Modifier and Type	Method and Description
void	`actionPerformed(ActionEvent e)` Invoked when an action occurs.

ImageViewer v 0.2

Centralized Event Handling:
ImageViewer lytter efter *alle* events

```
public class ImageViewer implements ActionListener {
    ...
    public void actionPerformed(ActionEvent event) {
        System.out.println("Menu item: " + event.getActionCommand());
    }
```

Lytter-metoden




```
JMenuBar menubar = new JMenuBar();
frame.setJMenuBar(menubar);
```

```
JMenu fileMenu = new JMenu("File");
menubar.add(fileMenu);
```

```
JMenuItem openItem = new JMenuItem("Open");
openItem.addActionListener(this);
fileMenu.add(openItem);
```

```
JMenuItem quitItem = new JMenuItem("Quit");
quitItem.addActionListener(this);
fileMenu.add(quitItem);
```

NB: event occurs --event-notification--> action listener **!**

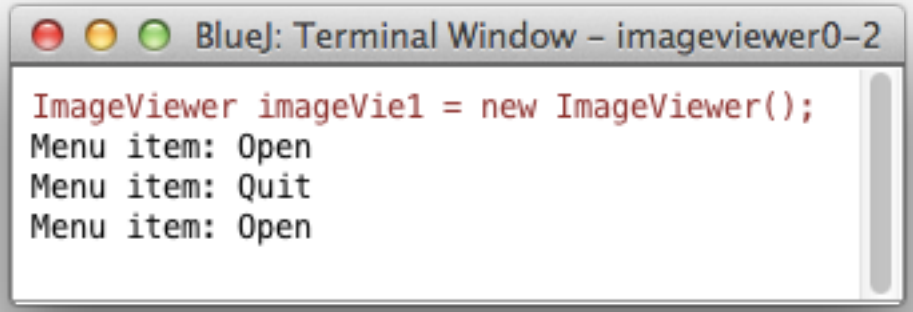
...som kalder `actionPerformed(e)`

Centralized Event Handling

- ImageViewer håndterer *alle* egne events:
  - Skal implementere interface: `ActionListener`
  - Skal definere en metode: `actionPerformed(e)`

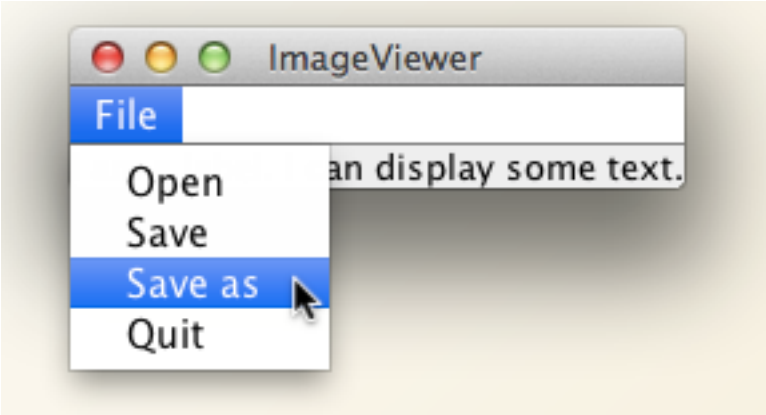
- ImageViewer objektet tilknyttes som event listener til *alle* relevante komponenter:
  - `item.addActionListener(this)`

- Når `actionPerformed(e)` kaldes, så undersøger den hvilken komponent hændelsen `e` kommer fra



EXERCISE: Event Handling

- **Extend V 0.2 program med '`Save`' & '`Save as`' samt Event Handling:**



- **`Open`:** *Print "**`åbn`**"* (på konsollen)
- **`Save`:** *Print "**`gem`**"* (på konsollen)
- **`Save as`:** *Print "**`gem som`**"* (på konsollen)
- **`Quit`:** *Stop programmet*

- **<u>Hint</u>:**

```
System.exit(1); // terminate execution!
```

Problemer med denne løsning (Centralized Event Handling)

- Det virker - og det er nemt for ***små*** systemer
  (...og det bliver brugt, så man bør kende det)

Ulemper!:

- Det er ufleksibelt at genkende komponenter på navn:

```
if (name.equals("Open")) { ...handle Open action... }
else if (name.equals("Quit")) { ...handle Quit action... }
else if (name.equals("Help")) { ...handle Help action... }
else { ...handle 999 other!?! actions... }
```

  - Hvad hvis menusproget nu skal være dansk eller fransk?
  - Det bliver noget rod for større systemer:
    - » readability, maintainability, extensibility :-(

- **Der er en bedre måde...:**
  - Lav en separat lytter til hver event
  - Lav lytteren som instans af en ***indre klasse...***

```
class EnclosingClass {
    ...
    class InnerClass {
        ...
    }
}
```

Recap: Inner Classes

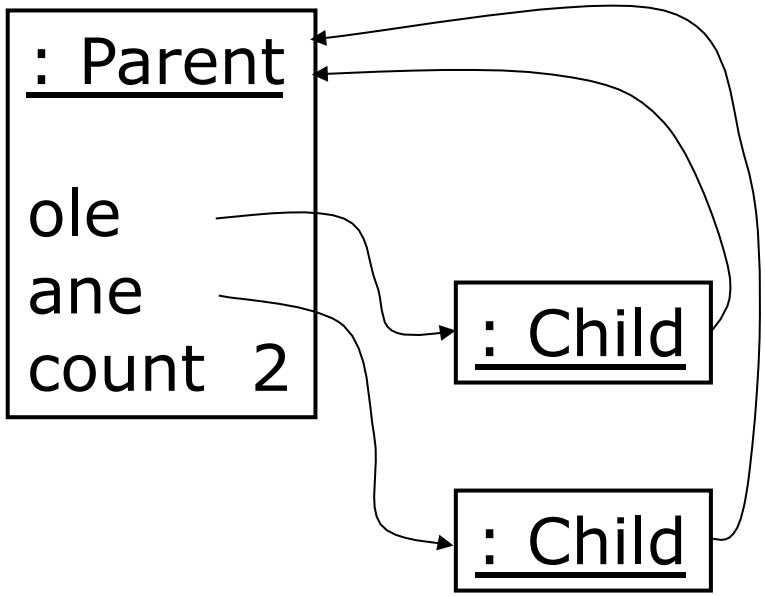
(Indre klasser)

Recap: Inner Classes

- En klasse kan erklæres inde i en anden
- Den indre kan bruge den ydres klasses felter
- Hver indre instans er knyttet til en ydre

```
class Parent {
  private int count = 0;
  Child ole = new Child();
  Child ane = new Child();

  class Child {
    public Child() {
      count++;
    }
  }
}
```

```
Parent eva = new Parent();
```



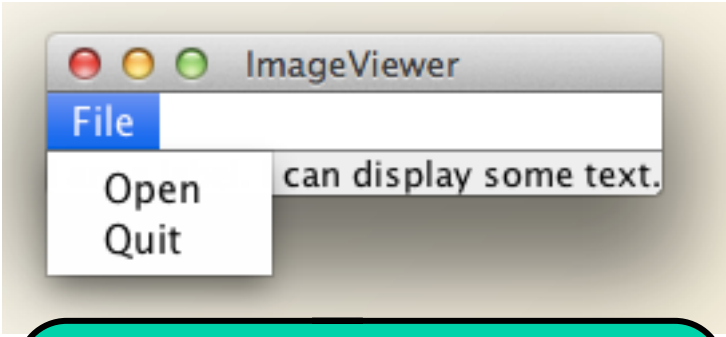
Anonymous Inner Classes

- Kan bruge omgivende klasses felter og metoder
- Bruges typisk når man kun skal have et objekt ...og klassenavnet er ligegyldigt
  - E.g., til ***event listeners***
- En anonym klasse skal navngive en ***supertype:***
  - Enten en ***superklasse*** [her "`Child`"]
  - ...eller et ***interface*** [se senere]

```
class Parent {

  Child ole = new Child() {
                  public string getName() {
                      return "Ole";
                  }
              }                          ;
}
```

Udtryk hvis *værdi* er: et *objekt* af ***anonym subklasse*** af `Child`

ImageViewer v 0.3



As before, but with improved structure
(without centralized event handler)

- **`ImageViewer`:**
  - Just display a simple frame – ...with a **menu!**
  - No transformations

ImageViewer v 0.3

```
JMenuItem openItem = new JMenuItem("Open");
openItem.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            open();
        }
    }
);
fileMenu.add(openItem);

JMenuItem quitItem = new JMenuItem("Quit");
quitItem.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            quit();
        }
    }
);
fileMenu.add(quitItem);
```

Særlig lytter til Open

Særlig lytter til Quit

ImageViewer v 0.3

```
JMenuItem openItem = new JMenuItem("Open");
openItem.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            open();
        }
    }
);
fileMenu.add(openItem);

JMenuItem quitItem = new JMenuItem("Quit");
quitItem.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            quit();
        }
    }
);
fileMenu.add(quitItem);
```

Uinteressant "boilerplate"

Interessant kode

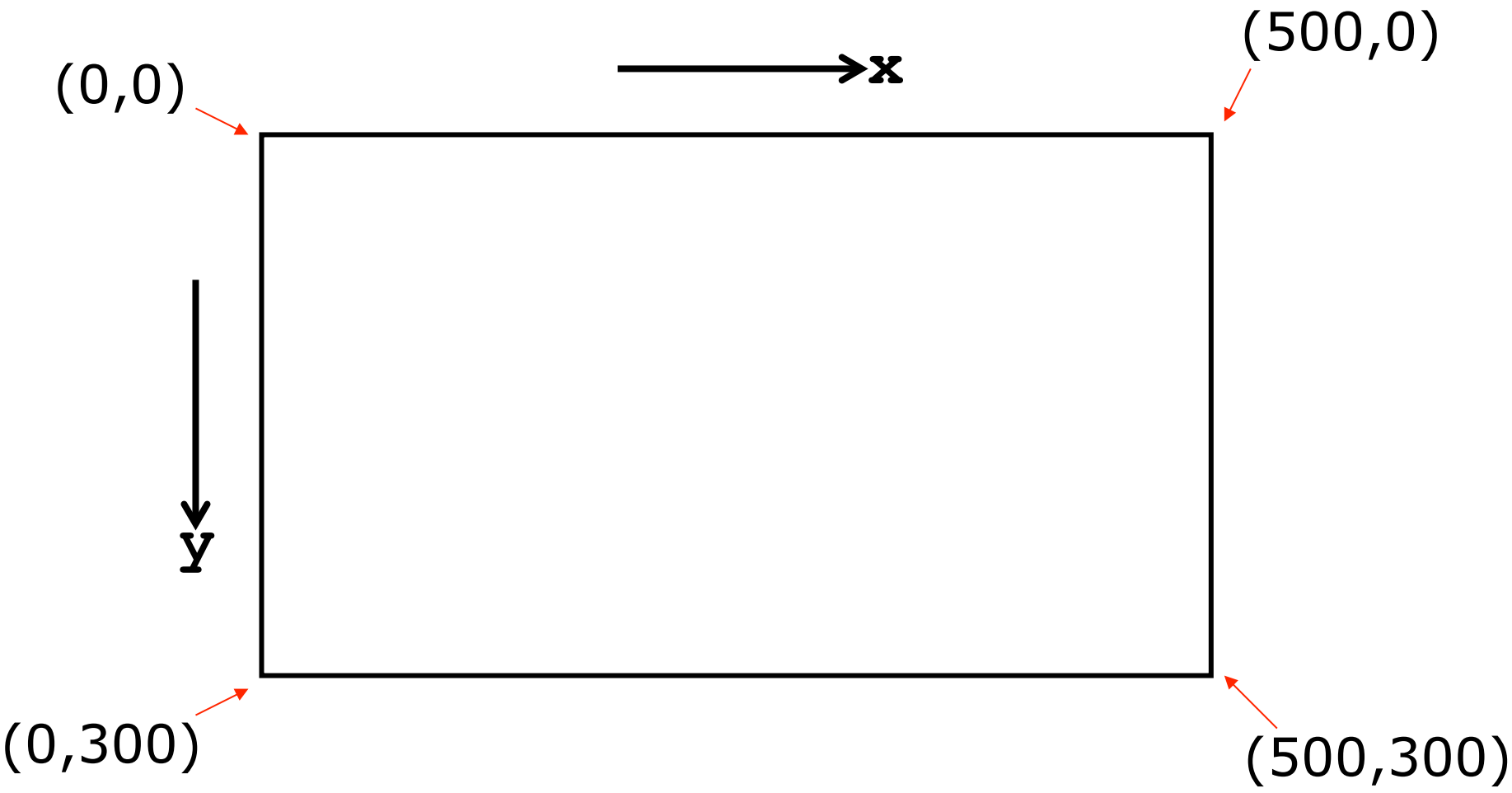
Uinteressant "boilerplate"

Interessant kode

Graphics i Java

Pixelbaseret tegning i Java

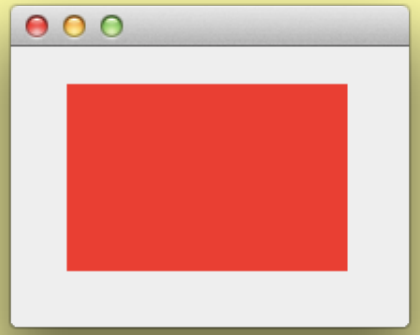
- **Koordinatsystem** (med y-axis nedad)**:**



Hello World - Java Graphics version

- Man kan tegne på **`java.awt.Component`**
- ...og dermed på alle dens ***subklasser:***
  - **`Container`, `Panel`, `JComponent`, `Jpanel` , ...**
- Ofte laver man en subklasse af **`JComponent`**:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TegneTegne extends JComponent {
   public void paint(Graphics g) {
      g.setColor(Color.RED);
      g.fillRect(30, 20, 150, 100);
   }

   public static void main(String[] args) {
      JFrame window = new JFrame();
      window.setSize(640,480);
      window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      window.getContentPane().add(new TegneTegne());
      window.setVisible(true);
   }
}
```



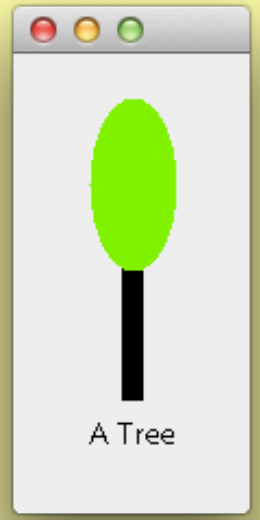
Her tegnes!

Metoden `paint(Graphics g)`

- Metode **`paint(Graphics g)`** kaldes når komponenten skal tegnes

- Graphics **`g`** har metoder til at tegne
  - streger, ovaler, cirkler, rektangler, tekst, ...

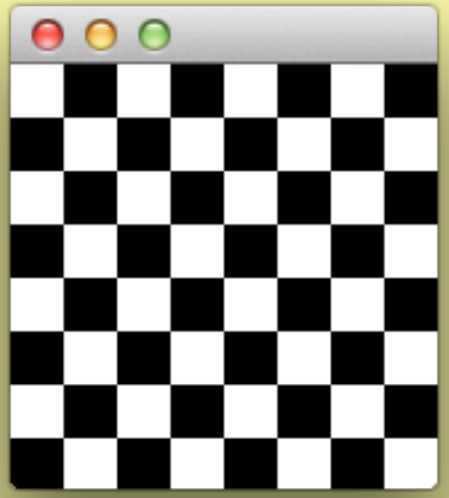
```
public class TegneTegne extends JComponent {
    ...
    public void paint(Graphics g) {
        g.setColor(Color.BLACK);
        g.fillRect(50, 60, 10, 100);
        g.setColor(Color.GREEN);
        g.fillOval(35, 20, 40, 80);
        g.setColor(Color.BLACK);
        g.drawString("A Tree", 35, 180);
    }
}
```



Tegn et skakbræt

- Dobbelt for-løkke
- Boolsk variabel styrer skiftende farver

```
private void chessboard(Graphics g) {
   boolean white = true;
   for (int i=0; i<8; i++) {
      for (int j=0; j<8; j++) {
         if (white) {
            g.setColor(Color.WHITE);
         } else {
            g.setColor(Color.BLACK);
         }
         white = !white;
         g.fillRect(i*20, j*20, 20, 20);
      }
      white = !white;
   }
}
```



Tilfældig-farvet skakbræt

- `java.awt.Color` er en såkaldt "RGB-kode": (**rød**, **grøn**, **blå**) // hver 0-255

```
private static final Random rnd = new Random();

Color randomColor() {
   return new Color(rnd.nextInt(256),
                    rnd.nextInt(256),
                    rnd.nextInt(256));
}

private void randomChessboard(Graphics g) {
   for (int i=0; i<8; i++) {
      for (int j=0; j<8; j++) {
         g.setColor(randomColor());
         g.fillRect(i*20, j*20, 20, 20);
      }
   }
}
```

Turtle Graphics

Turtle Graphics

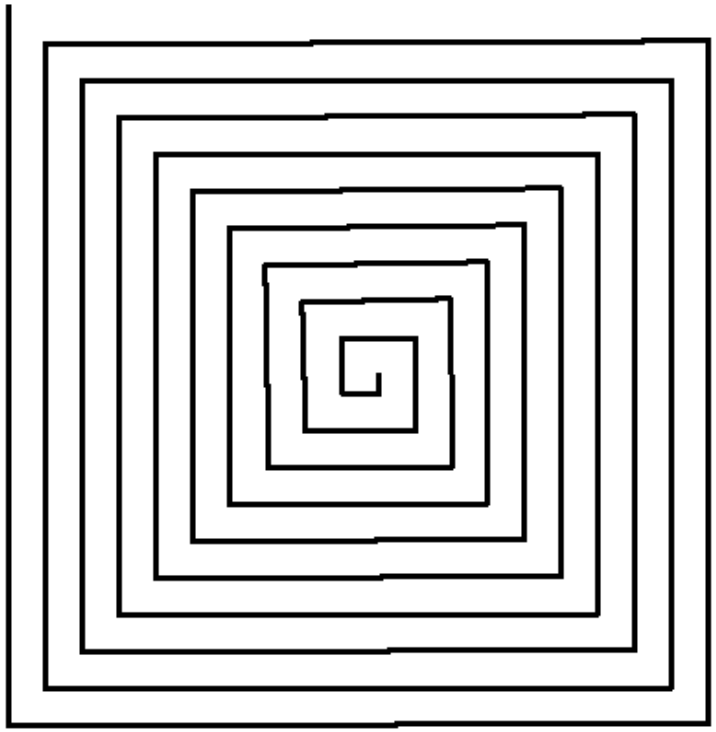
```
// imports

public class Turtle extends JComponent {
   protected double angle = 0, x = 100, y = 100;

   public Turtle() {
      JFrame window = new JFrame();
      window.setSize(640,480);
      window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      window.getContentPane().add(this);
      window.setVisible(true);
   }

   protected void init(int x, int y) { angle = 0; this.x = x; this.y = y; }

   public void turn(double degrees) { angle = angle + degrees; }

   public void turnto(double degrees) { angle = degrees; }

   public void move(Graphics g, int distance) {
      double new_x = x + distance * Math.cos(Math.toRadians(angle));
      double new_y = y + distance * Math.sin(Math.toRadians(angle));
      g.drawLine((int) x, (int) y, (int) new_x, (int) new_y);
      x = new_x;
      y = new_y;
   }
}
```

Up-Down Turtle

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class ExtendedTurtle extends Turtle {
   protected boolean draw = true;

   public ExtendedTurtle() {
      super();
   }

   public void up() { draw = false; }

   public void down() { draw = true; }

   public void move(Graphics g, int distance) {
      if (draw) super.move(g, distance);
      else {
         g.setColor(Color.WHITE);
         super.move(g, distance);
         g.setColor(Color.BLACK);
      }
   }
}
```

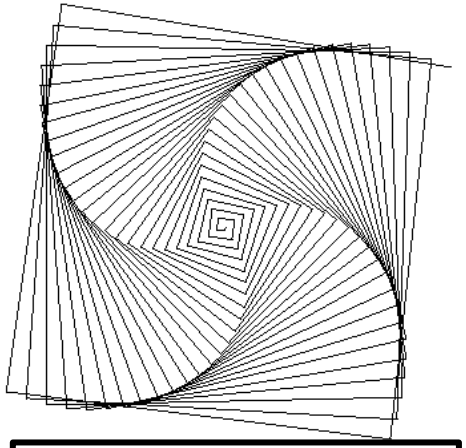
Squaaare

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

class MyTurtle extends Turtle {
   public MyTurtle() {
      super();
   }

   public void paint(Graphics g) {
      init(300,250);
      for (int i=0; i<400; i=i+10) {
         move(g, i);
         turn(90);
      }
   }
}
```

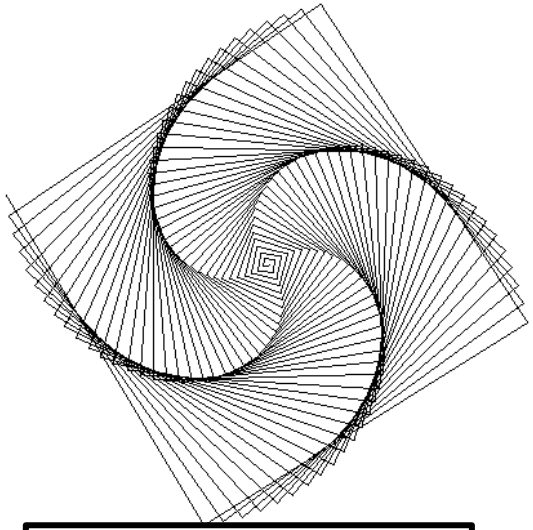


Spirals

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

class MyTurtle extends Turtle {
   public MyTurtle() {
      super();
   }

   public void spiral(Graphics g, int size, int step) {
      init(300,250);
      for (int i=0; i<size; i=i+step) {
         move(g, i);
         turn(91);
      }
   }

   public void paint(Graphics g) {
      spiral(g, 300, 2);
   }
}
```

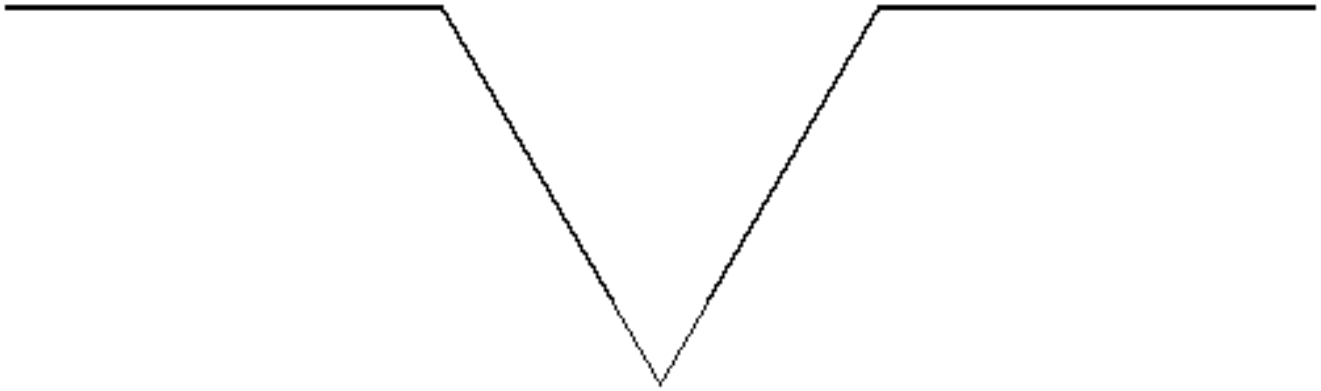


```
spiral(g, 300, 3);
```



```
spiral(g, 300, 2);
```

Kock Fraktaler



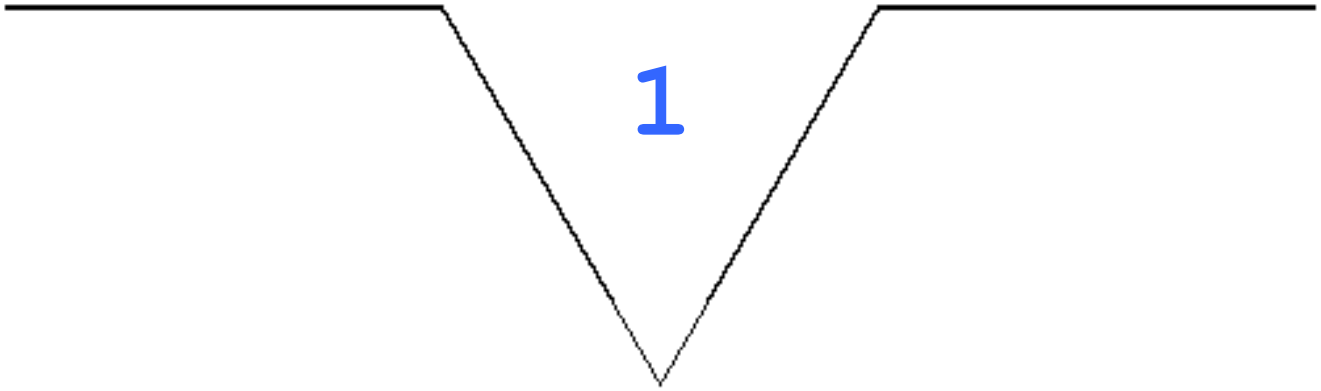
```
public class Fractal extends Turtle {
   void kock1(Graphics g, int dir, int length) {
      move(g, length/3);
      turn(dir+60);
      move(g, length/3);
      turn(dir-120);
      move(g, length/3);
      turn(dir+60);
      move(g, length/3);
   }

   public void paint(Graphics g) {
      init(75,175);
      kock1(g, 0, 500);
   }
}
```

Recursive Kock Fraktaler

0

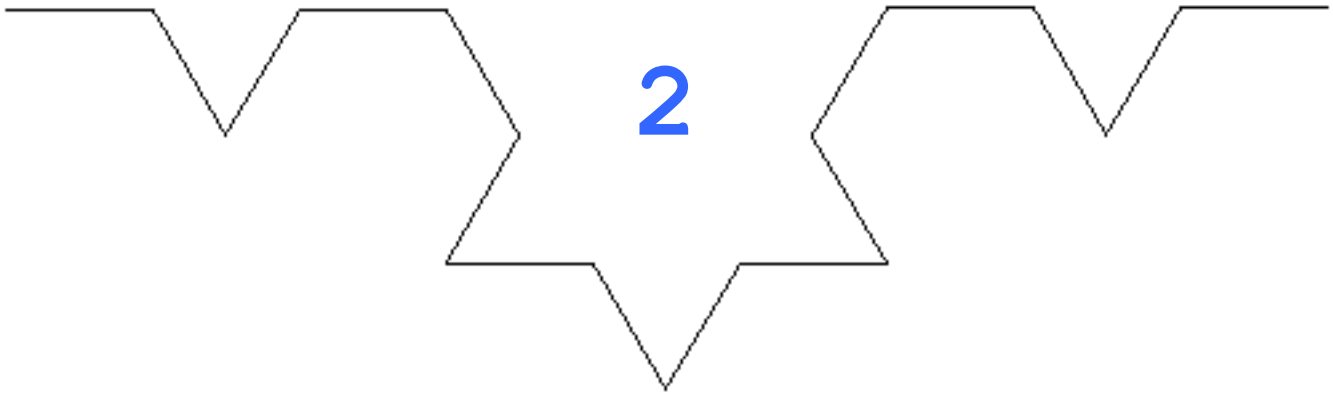
```
public class Fractal extends Turtle {
   void kock(Graphics g, int n, int dir, int length) {
      if (n==0) { turnto(dir); move(g, length); }
      else {
         kock(g, n-1, dir,    length/3);
         kock(g, n-1, dir+60, length/3);
         kock(g, n-1, dir-60, length/3);
         kock(g, n-1, dir,    length/3);
      }
   }

   public void paint(Graphics g) {
      init(75,175);
      kock(g, 0, 0, 500);
   }
}
```

Recursive Kock Fraktaler



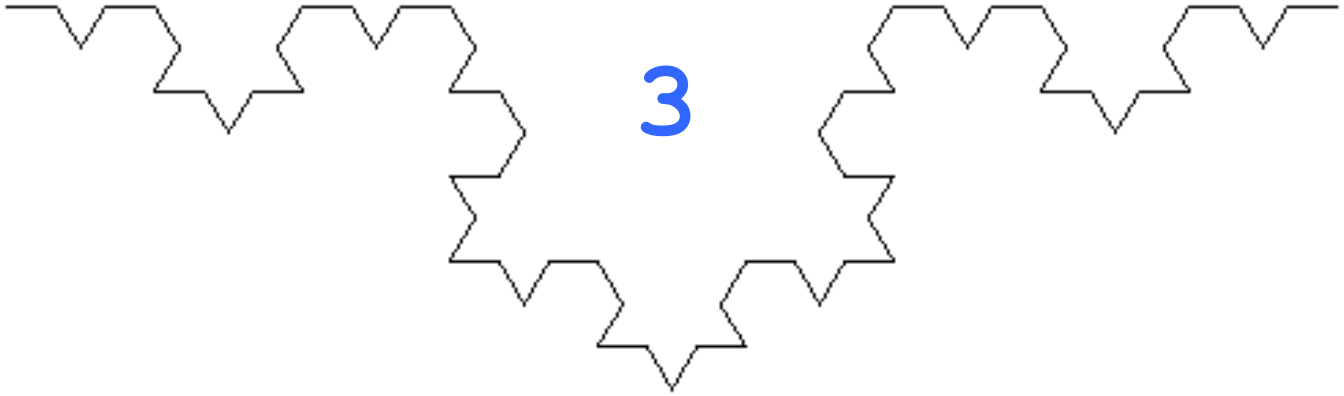

```
public class Fractal extends Turtle {
   void kock(Graphics g, int n, int dir, int length) {
      if (n==0) { turnto(dir); move(g, length); }
      else {
         kock(g, n-1, dir,    length/3);
         kock(g, n-1, dir+60, length/3);
         kock(g, n-1, dir-60, length/3);
         kock(g, n-1, dir,    length/3);
      }
   }

   public void paint(Graphics g) {
      init(75,175);
      kock(g, 1, 0, 500);
   }
}
```

Recursive Kock Fraktaler



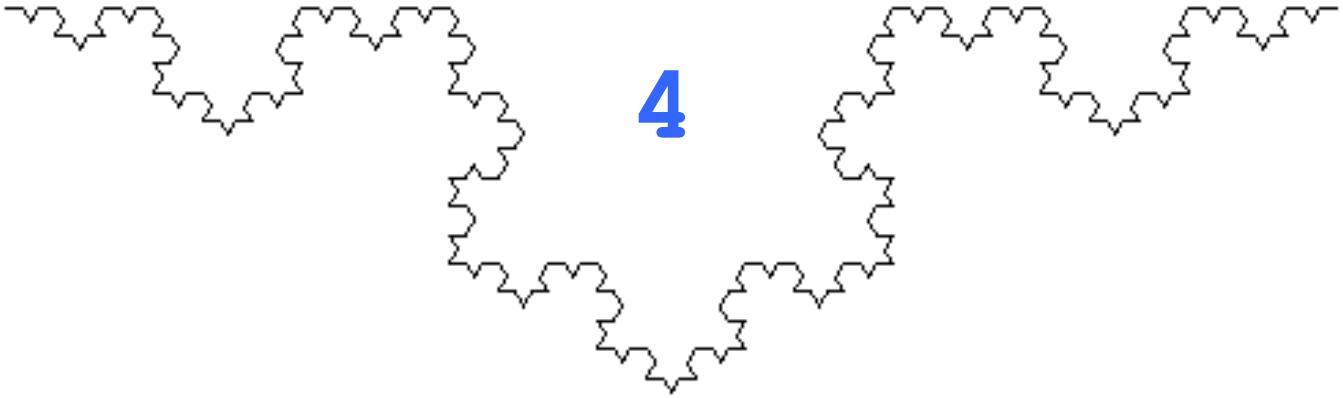

```
public class Fractal extends Turtle {
   void kock(Graphics g, int n, int dir, int length) {
      if (n==0) { turnto(dir); move(g, length); }
      else {
         kock(g, n-1, dir,    length/3);
         kock(g, n-1, dir+60, length/3);
         kock(g, n-1, dir-60, length/3);
         kock(g, n-1, dir,    length/3);
      }
   }

   public void paint(Graphics g) {
      init(75,175);
      kock(g, 2, 0, 500);
   }
}
```

Recursive Kock Fraktaler



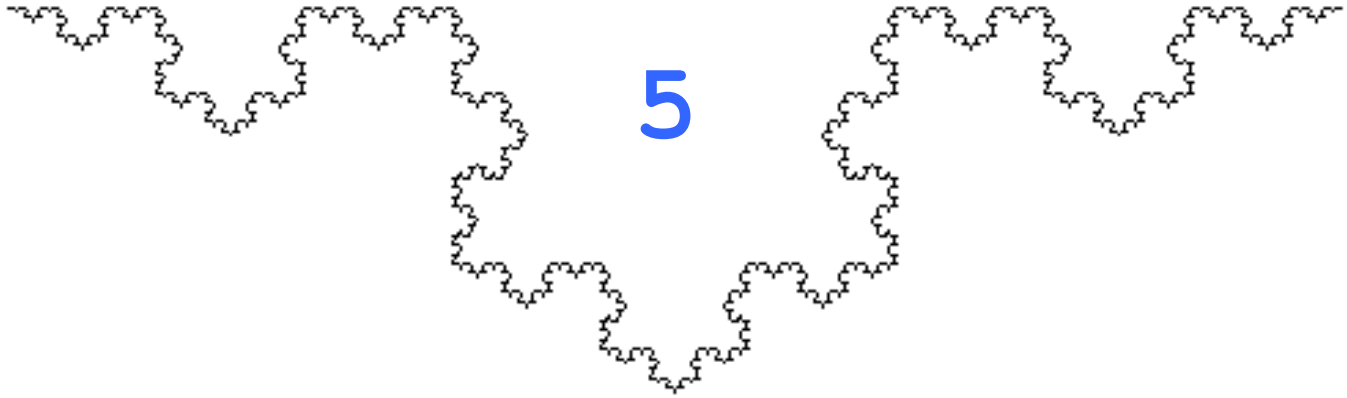

```
public class Fractal extends Turtle {
   void kock(Graphics g, int n, int dir, int length) {
      if (n==0) { turnto(dir); move(g, length); }
      else {
         kock(g, n-1, dir,    length/3);
         kock(g, n-1, dir+60, length/3);
         kock(g, n-1, dir-60, length/3);
         kock(g, n-1, dir,    length/3);
      }
   }

   public void paint(Graphics g) {
      init(75,175);
      kock(g, 3, 0, 500);
   }
}
```

Recursive Kock Fraktaler




```
public class Fractal extends Turtle {
   void kock(Graphics g, int n, int dir, int length) {
      if (n==0) { turnto(dir); move(g, length); }
      else {
         kock(g, n-1, dir,    length/3);
         kock(g, n-1, dir+60, length/3);
         kock(g, n-1, dir-60, length/3);
         kock(g, n-1, dir,    length/3);
      }
   }

   public void paint(Graphics g) {
      init(75,175);
      kock(g, 4, 0, 500);
   }
}
```

Recursive Kock Fraktaler




```
public class Fractal extends Turtle {
   void kock(Graphics g, int n, int dir, int length) {
      if (n==0) { turnto(dir); move(g, length); }
      else {
         kock(g, n-1, dir,    length/3);
         kock(g, n-1, dir+60, length/3);
         kock(g, n-1, dir-60, length/3);
         kock(g, n-1, dir,    length/3);
      }
   }

   public void paint(Graphics g) {
      init(75,175);
      kock(g, 5, 0, 500);
   }
}
```

Øvelser & Næste gang

- **<u>Øvelser</u>:**
  Prøv at bruge dette til at ***lave et simpelt spil***
  (der er helt frit valg hvilket spil I ønsker at lave, blot I forsøger at bruge så meget som muligt af det I har lært)

- **<u>Torsdag</u>:** Introduktion til databaser

(Møde med klasse-repæsentanter nu)