University of Southern Denmark • Syddansk Universitet

DM502

Forelæsning 8

Indhold

- Information
- Interfaces
  - Comparable
  - Iterator / Iterable
- Try-catch (igen)

Information

- Der findes (groft sagt) to slags DM502-studerende
  - Dem er synes kurset er meget svært
  - Dem er synes kurset er meget let
- Ídeen med DM502 er at bringe alle op på ca. samme niveau
- Til dem der synes kurset er meget svært
  - Vi gør hvad vi kan for at hjælpe jer til at lære det
  - Tempoet er lagt an på jer
- Til dem der synes kurset er meget let
  - Det bliver bedre i DM503
  - Glæd dig over at du er god til noget ;-)
  - Udvid programmerne (særlig i projektet)
  - Brug tiden på at lære andre ting (diskret matematik)

Motivation

Motivation

- Problem:
  - Vi vil gerne skrive et program der kan sortere lister af elementer

Motivation

- Problem:
  - Vi vil gerne skrive et program der kan sortere lister af elementer
- Hvordan sorterer man? (algoritme)

Motivation

- Problem:
  - Vi vil gerne skrive et program der kan sortere lister af elementer
- Hvordan sorterer man? (algoritme)
  - Find det mindste element og flyt til plads 0
  - Find det næst-mindste element og flyt til plads 1
  - ...

Motivation

- Problem:
  - Vi vil gerne skrive et program der kan sortere lister af elementer
- Hvordan sorterer man? (algoritme)
  - Find det mindste element og flyt til plads 0
  - Find det næst-mindste element og flyt til plads 1
  - ...
- Hvilke elementer kan sorteres?
  - Hvad skal vi kræve af elementerne for at de kan sorteres?

Motivation

- Elementerne skal kunne sammenlignes (total ordning)
- Vi gider ikke lave unødigt meget arbejde
- Algoritmen for sortering er den sammen, om man sorterer
  - Heltal
  - Personer efter højde
  - Biler efter kilometertal
  - Osv...
- Det ville være "træls" at skrive et nyt sorteringsprogram for hvert tilfælde
- Vil gerne nøjes med et sorteringsprogram, som kan sortere elementer der kan sammenlignes

Motivation

- Hvordan kan man sammenligne elementer?
- Vil gerne sammenligne to elementer `e1` og `e2`
  - Instanser af en klasse som kan sammenlignes (`Person`, `Car`, `int`, ...)
- Det kunne gøres igennem en metode
  - Fx "`e1 < e2`" hvis og kun hvis "`e1.compareTo(e2) < 0`"
- Hvordan skriver vi et program der kan sortere, hvis blot vi (programmet) ved at elementerne har en sammenligningsmetode der overholder ovenstående?

Interfaces

- Java og interfaces to the rescue!
- Et interface (grænseflade) er en beskrivelse af hvilke metoder en klasse som minimum skal have
  - Giver en veldefineret måde at bruge klasser der er forskellige men ser ens ud på nogle områder
    - Fx har `Dog` og `Cat` det til fælles at de begge har en højde (`public double getHeight()`)
  - Fx har `Car` og `Bicycle` det til fælles at de begge har en farve (`public String getColor()`)
  - Osv...

Interfaces

- Interfaces i Java
  - ```
    public interface Printable {
        public void print();
        // evt flere metode-erklæringer
    }
    ```
  - Blot en liste af metoder, uden implementation
- Minder meget om klasser
- Skal ligge i en fil ved navn Printable.java
- Hvis en klasse implementere interfacet `Printable` skal klasse have en metode `public void print()`
    ```

Printable.java

```
public interface Printable {
    public void print();
}
```

Car.java

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

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

    public int getMileage() {
        return mileage;
    }

    public void print() {
        System.out.println( color + " " + model + " fra " + year +
                            " der har kørt " + mileage + " kilometer." );
    }

     ...
}
```

Car.java

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

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

    public int getMileage() {
        return mileage;
    }

    public void print() {
        System.out.println( color + " " + model + " fra " + year +
                            " der har kørt " + mileage + " kilometer." );
    }

     ...
}
```

Interfaces

- “Hvorfor skulle det nu være så smart?!?”

Bicycle.java

```
public class Bicycle implements Printable {
    private String model;
    private String color;
    private int gears;

    public Bicycle( String m, String c, int g ) {
        model = m;
        color = c;
        gears = g;
    }

    public void print() {
        System.out.println( color + " " + model + " med " +
                            gears + " gear." );
    }
}
```

Bicycle.java

```
public class Bicycle implements Printable {
    private String model;
    private String color;
    private int gears;

    public Bicycle( String m, String c, int g ) {
        model = m;
        color = c;
        gears = g;
    }

    public void print() {
        System.out.println( color + " " + model + " med " +
                            gears + " gear." );
    }
}
```

MainProgram.java

```
public class MainProgram {
    public static void main( String[] args ) {
        Car car = new Car( "Audi A4", "sølvgrå" );
        Bicycle bike = new Bicycle( "Kildemoes", "blå", 7 );

        car.drive( 50 );
        car.drive( 15 );

        printTransportation( bike );
        printTransportation( car );
    }

    public static void printTransportation( Printable o ) {
        o.print();
    }
}
```

Interfaces

- "Hvorfor skulle det nu være så smart?!?"
  - Fordi metoden `printTransportation` ikke behøver vide andet end at dets input (argument) kan printes
  - Det eneste `printTransportation` ved, er at dets argument har en metode `public void print();`

Interfaces

- Tilbage til vores sorterings-eksempel
- Vi vil gerne lave et interface der angiver at objekter fra en klasse der implementerer interfacet kan sammenlignes
- ```
  public interface Comparable {
      public int compareTo( Object o );
  }
  ```
  - Ligger i `java.lang.Comparable` (skal altså ikke importeres)
  - `Object` er et generelt objekt, dvs. en vilkårlig klasse (for mere information om nedarvning og generics følg DM503 :-)
  - "Kontrakt" ved brug af `Comparable`:
    - `a.compareTo( b ) < 0` hvis og kun hvis `a < b`


  ```

Comparable.java

```
public interface Comparable {
    public int compareTo( Object o );
}
```

MyInteger.java

```
public class MyInteger implements Comparable {
```

MyInteger.java

```
public class MyInteger implements Comparable {
    private int val;
```

MyInteger.java

```
public class MyInteger implements Comparable {
     private int val;

     public MyInteger( int v ) {
          val = v;
     }
```

MyInteger.java

```
public class MyInteger implements Comparable {
    private int val;

    public MyInteger( int v ) {
        val = v;
    }

    public int getInt() {
        return val;
    }
```

MyInteger.java

```
public class MyInteger implements Comparable {
    private int val;

    public MyInteger( int v ) {
        val = v;
    }

    public int getInt() {
        return val;
    }

    public int compareTo( Object o ) {
        MyInteger i = (MyInteger) o;
        return val - i.getInt();
    }
}
```

MyInteger.java

```
public class MyInteger implements Comparable {
    private int val;

    public MyInteger( int v ) {
        val = v;
    }

    public int getInt() {
        return val;
    }

    public int compareTo( Object o ) {
        MyInteger i = (MyInteger) o;
        return val - i.getInt();
    }
}
```

Bemærk:
a.compareTo(b) < 0
hvis og kun hvis a < b

MySorter.java

```
public class MySorter {
    public static void main( String[] args ) {
        MyInteger[] list = new MyInteger[5];
        MyInteger i;

        list[0] = new MyInteger( 42 );
        list[1] = new MyInteger( 5 );
        list[2] = new MyInteger( 13 );
        list[3] = new MyInteger( 9 );
        list[4] = new MyInteger( 1 );

        sort( list );

        for( int j = 0; j < list.length; j++ ) {
            System.out.println( list[j].getInt() );
        }
    }
```

MySorter.java

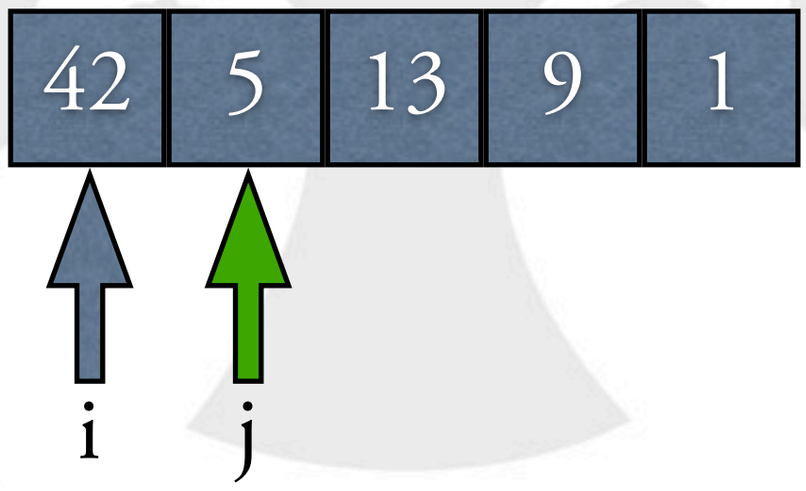
```
    public static void sort( Comparable[] list ) {
        int i, j;

        for( i = 0; i < list.length; i++ ) {
            for( j = i+1; j < list.length; j++ ) {
                if( list[j].compareTo( list[i] ) < 0 ) {
                    Comparable x = list[i];
                    list[i] = list[j];
                    list[j] = x;
                }
            }
        }
    }
}
```

MySorter.java

```
    public static void sort( Comparable[] list ) {
        int i, j;

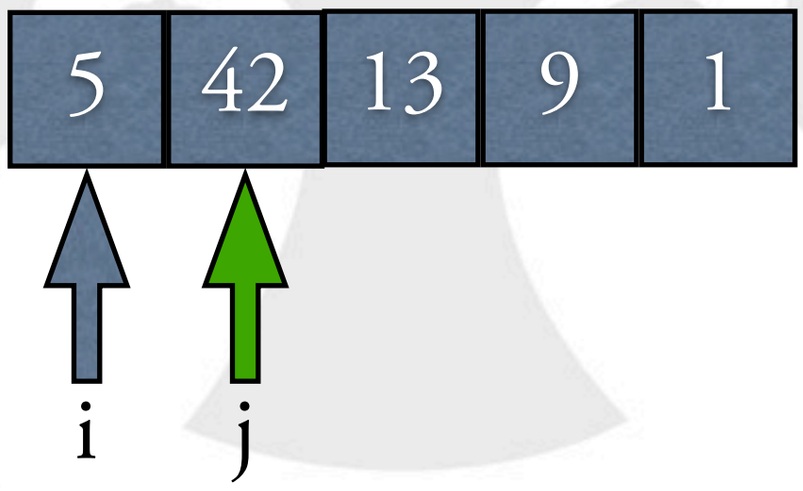
        for( i = 0; i < list.length; i++ ) {
            for( j = i+1; j < list.length; j++ ) {
                if( list[j].compareTo( list[i] ) < 0 ) {
                    Comparable x = list[i];
                    list[i] = list[j];
                    list[j] = x;
                }
            }
        }
    }
}
```



MySorter.java

```
        public static void sort( Comparable[] list ) {
            int i, j;

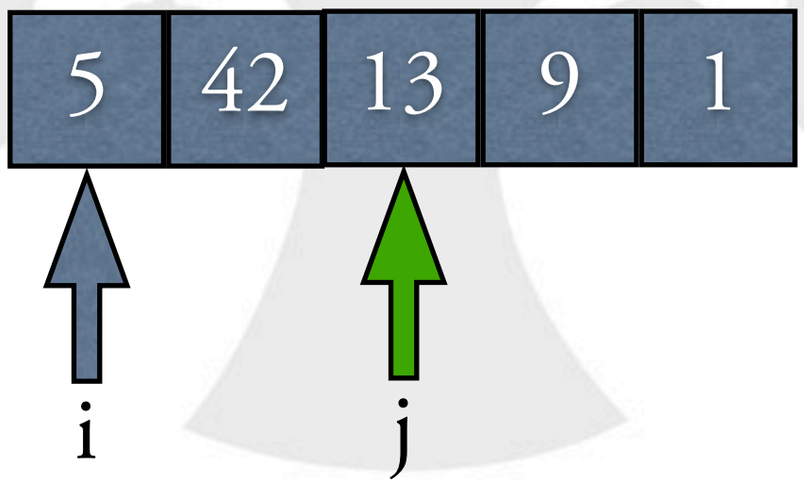
            for( i = 0; i < list.length; i++ ) {
                for( j = i+1; j < list.length; j++ ) {
                    if( list[j].compareTo( list[i] ) < 0 ) {
                        Comparable x = list[i];
                        list[i] = list[j];
                        list[j] = x;
                    }
                }
            }
        }
    }
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

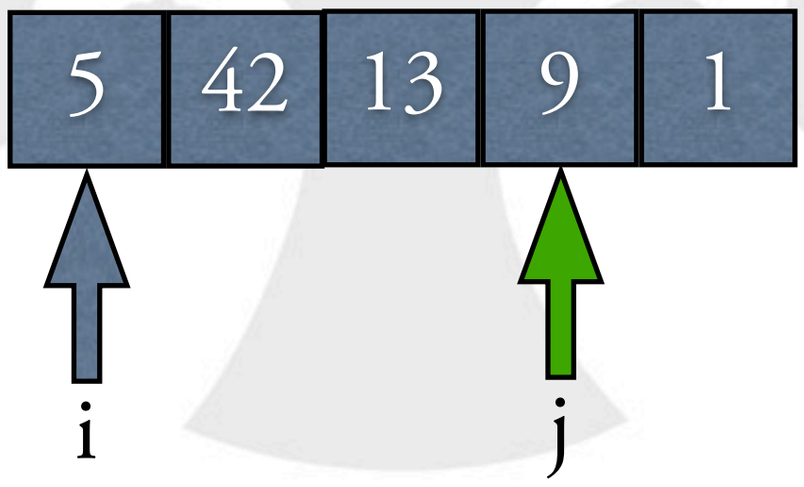
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

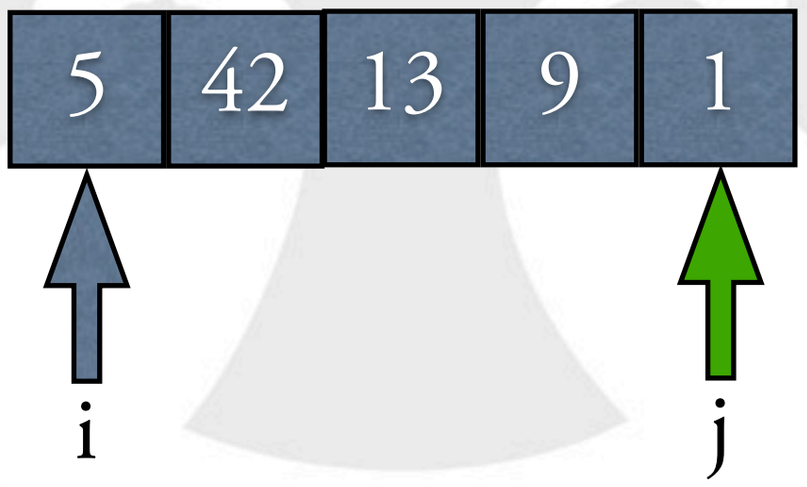
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
}
```



MySorter.java

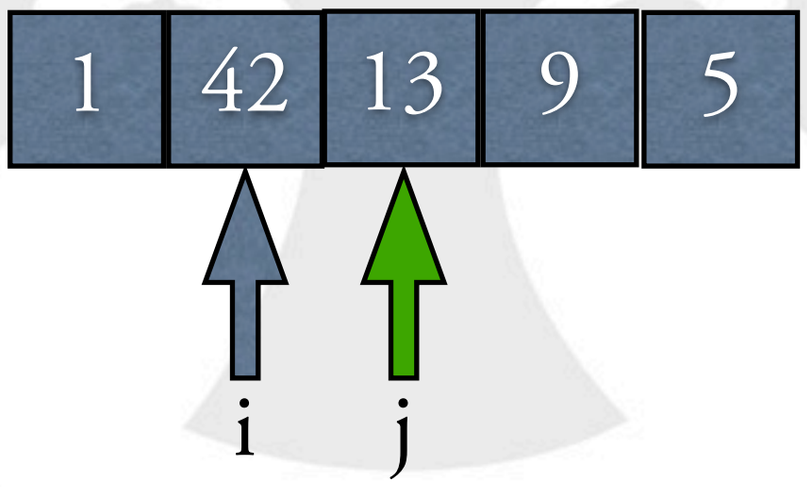
```
public static void sort( Comparable[] list ) {
    int i, j;

    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
}
```

MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

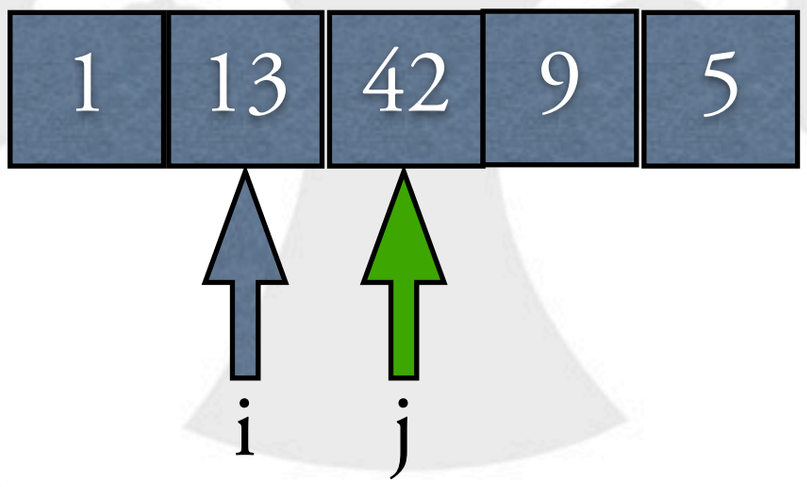
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

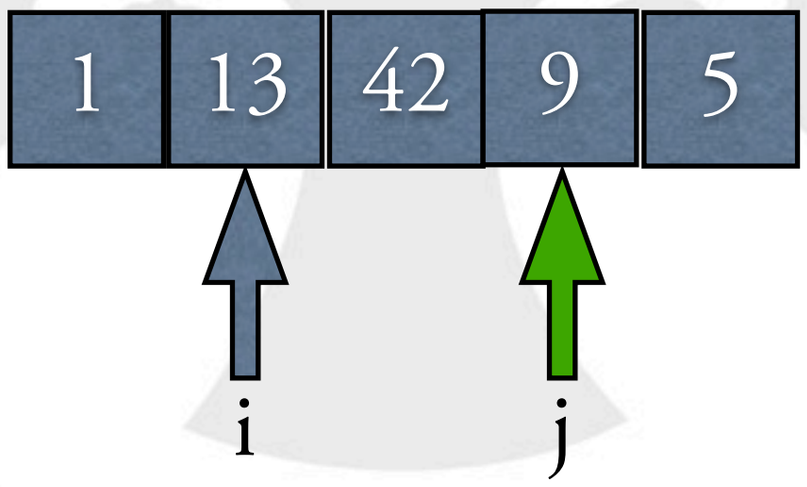
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

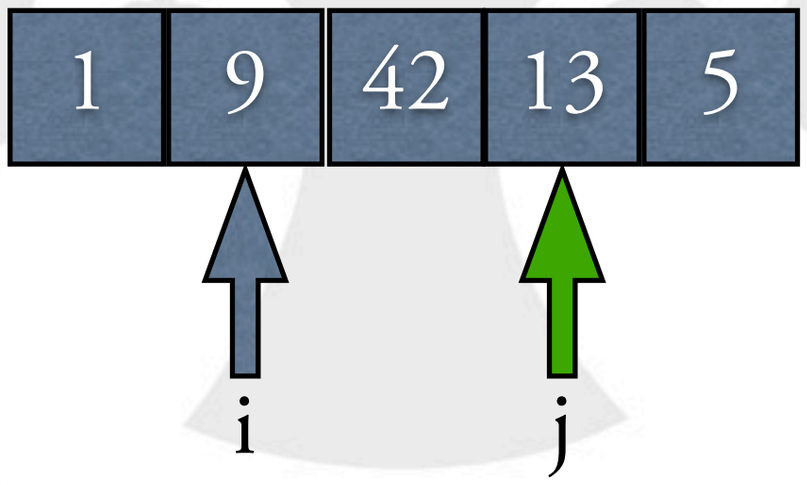
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

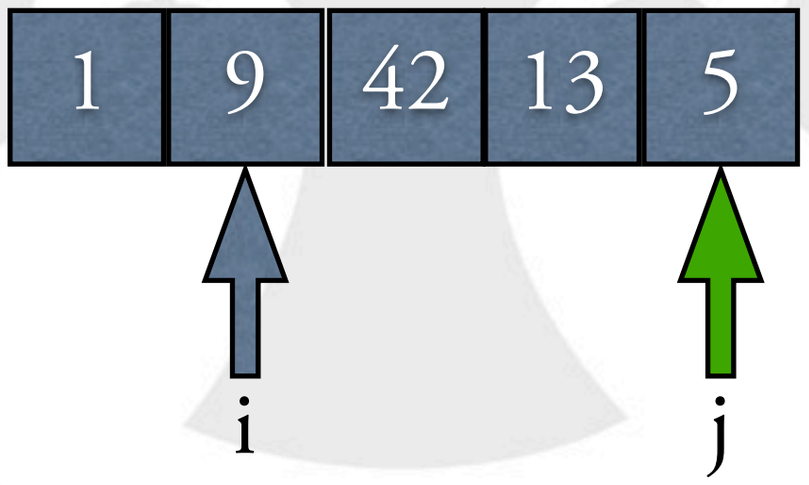
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

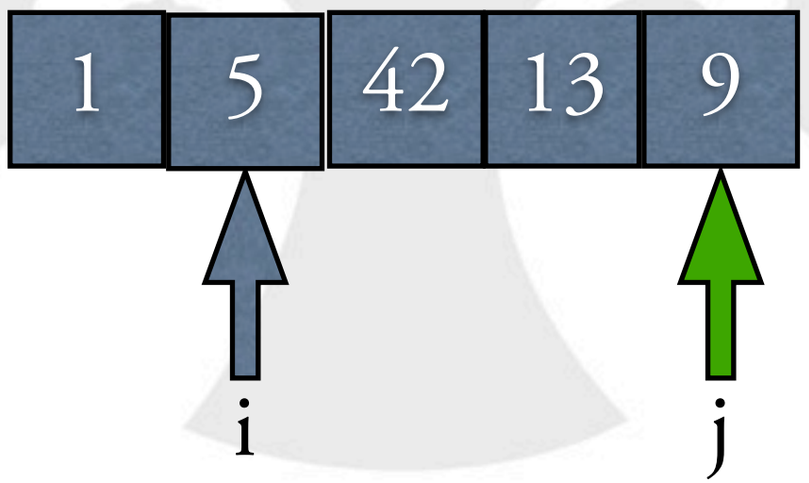
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

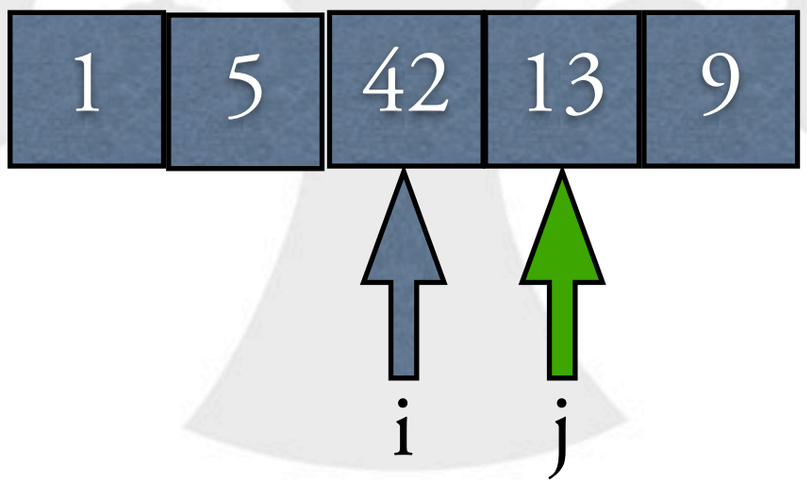
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

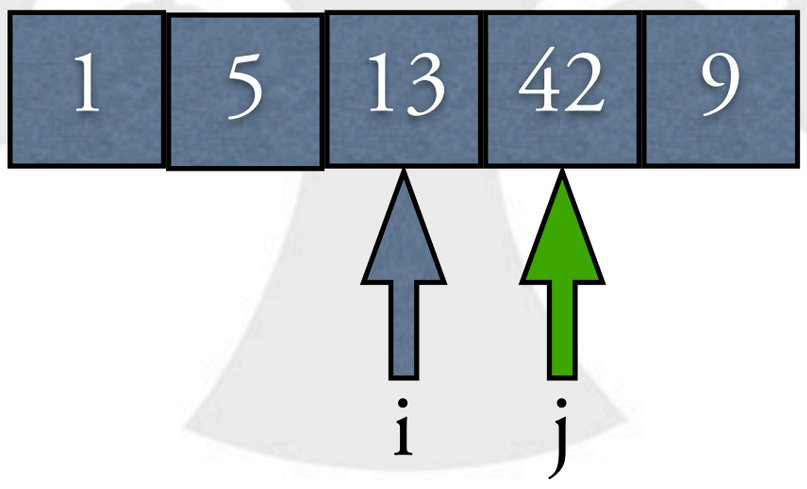
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

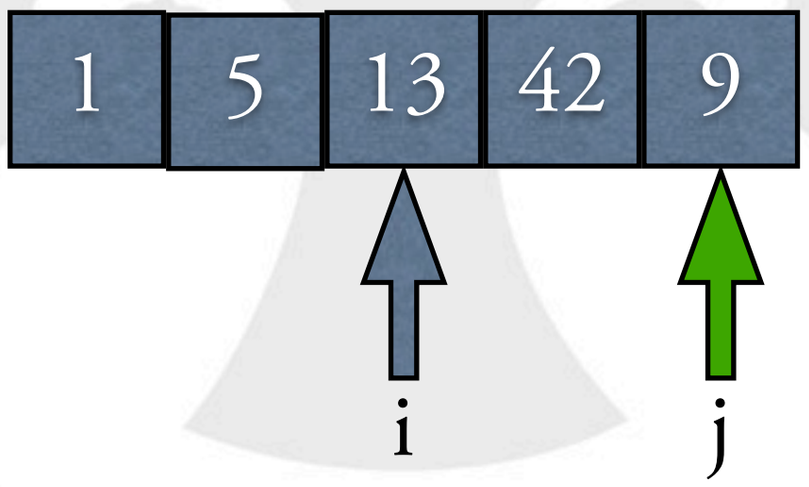
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

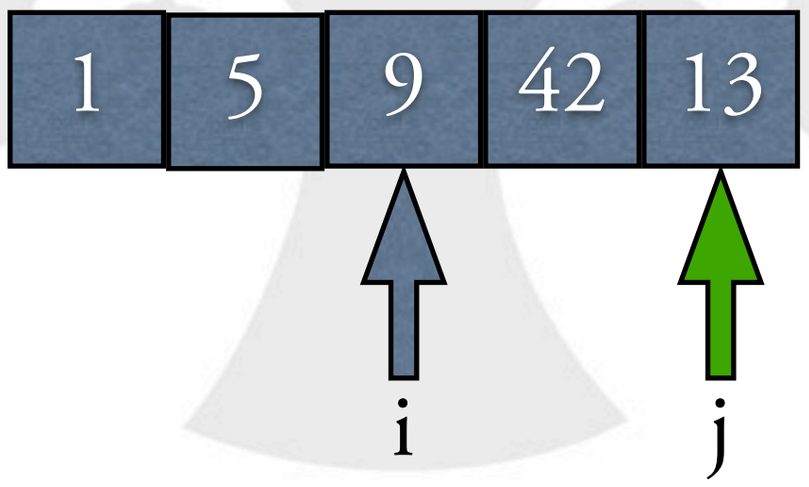
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
}
```



MySorter.java

```
    public static void sort( Comparable[] list ) {
        int i, j;

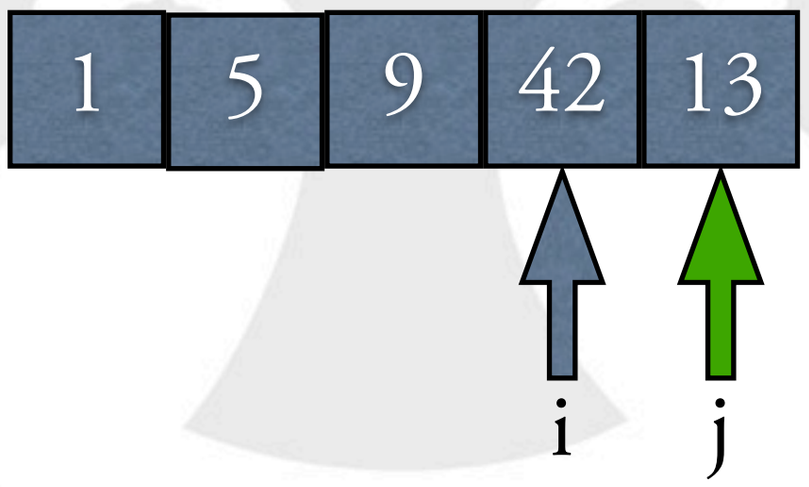
        for( i = 0; i < list.length; i++ ) {
            for( j = i+1; j < list.length; j++ ) {
                if( list[j].compareTo( list[i] ) < 0 ) {
                    Comparable x = list[i];
                    list[i] = list[j];
                    list[j] = x;
                }
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

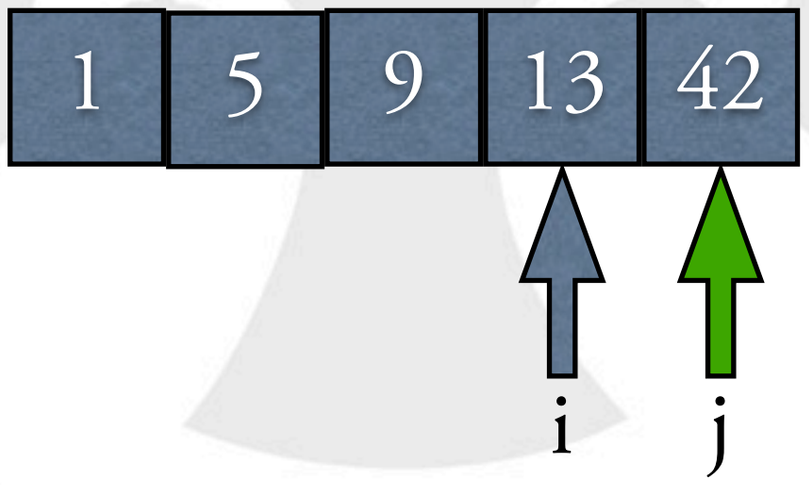
    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```



MySorter.java

```
public static void sort( Comparable[] list ) {
    int i, j;

    for( i = 0; i < list.length; i++ ) {
        for( j = i+1; j < list.length; j++ ) {
            if( list[j].compareTo( list[i] ) < 0 ) {
                Comparable x = list[i];
                list[i] = list[j];
                list[j] = x;
            }
        }
    }
}
```

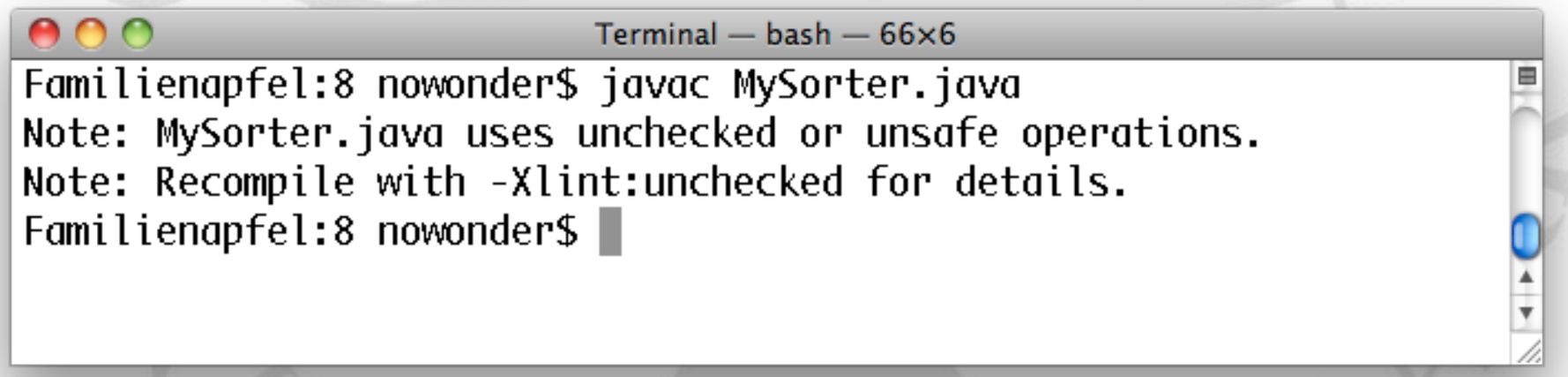


NB

- Bemærk at metoden

```
public static void sort( Comparable[] list )
```

ikke behøver at vide andet end at elementerne kan sammenlignes
- Kan sortere fx `MyDouble`, `Person`, `Car`, osv... hvis blot de implementerer interfacet `Comparable`.
- Ignorer



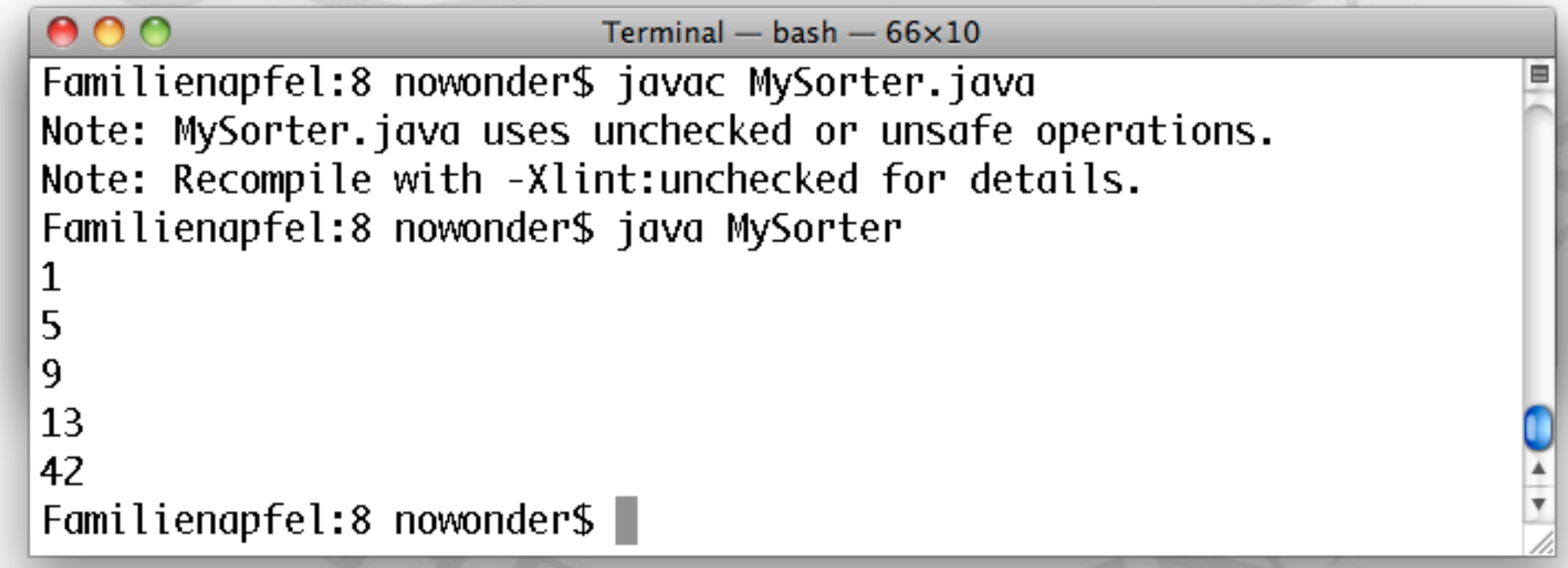
- Mere om det i DM503

NB

- Bemærk at metoden

  ```
  public static void sort( Comparable[] list )
  ```

  ikke behøver at vide andet end at elementerne kan sammenlignes
- Kan sortere fx `MyDouble`, `Person`, `Car`, osv... hvis blot de implementerer interfacet `Comparable`.
- Ignorer



Iterable & Iterator

- To interfaces i `java.util`
- Abstraherer over typen Liste
  - `hasNext()` - Er der et næste element
  - `next()` - Henter det næste element
  - `remove()` - Fjerner det sidst returnerede element
- Implementeres af bl.a. arrays og ArrayList
- Muliggør kode på følgende form

```
public class IterableExample {
    public static void main( String[] args ) {
        int[] list = {1,2,3,4,5,6};
        for (int i : list)
            System.out.println(i);
    }
}
```

Iterable & Iterator

- To interfaces i `java.util`
- Abstrahere over typen Liste
  - `hasNext()` - Er der et næste element
  - `next()` - Henter det næste element
  - `remove()` - Fjerner det sidst returnerede element
- Implementeres af bl.a. arrays og ArrayList
- Muliggør kode på følgende form

```
public class IterableExample {
    public static void main( String[] args ) {
        int[] list = {1,2,3,4,5,6};
        for (int i : list)
            System.out.println(i);
    }
}
```

Iterable & Iterator

- `for (int i : list)`
  - `i` tager nu hver værdi i arrayet `list`
  - Tænk på det som:

```
for (int j = 0; j < list.length; j++) {
     i = list[j];
     ...
}
```

And now for something completely different...

Exceptions

- Vi har kigget på try-catch til a fange (runtime) fejl
  - ```
    try {
        average = sum / count;
    } catch( ArithmeticException ae ) {
        System.out.println("Fejl!");
    }
    ```
  - Hvis division med 0, skrives “Fejl!” men programmet går ikke ned
  - Hvad sker hvis vi ikke fanger fejlen (udover at programmet går ned)?
    - Fejlen “back-tracker” gennem programmet
    - Fejlen kan fanges hvor som helst på den sti igennem programmet


    ```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

ArithmeticException

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Linje 19

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Linje 15

Linje 19

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Linje 11

Linje 15

Linje 19

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Linje 7

Linje 11

Linje 15

Linje 19

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        func3();
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Linje 3

Linje 7

Linje 11

Linje 15

Linje 19

```
Terminal — bash — 67×10
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionPropagation.func4(ExceptionPropagation.java:19)
        at ExceptionPropagation.func3(ExceptionPropagation.java:15)
        at ExceptionPropagation.func2(ExceptionPropagation.java:11)
        at ExceptionPropagation.func1(ExceptionPropagation.java:7)
        at ExceptionPropagation.main(ExceptionPropagation.java:3)
Familienapfel:8 nowonder$
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        try {
            func3();
        } catch( ArithmeticException ae ) {
            System.out.println( "Fanget i func2" );
        }
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        try {
            func3();
        } catch( ArithmeticException ae ) {
            System.out.println( "Fanget i func2" );
        }
    }

    public static void func3() {
        func4();
    }

    public static void func4() {
        int val = 32 / 0;
    }
}
```

Exceptions

```
public class ExceptionPropagation {
    public static void main( String[] args ) {
        func1();
    }

    public static void func1() {
        func2();
    }

    public static void func2() {
        try {
            func3();
        } catch( ArithmeticException ae ) {
            System.out.println( "Fanget i func2" );
        }
    }

    pu

    }

    pu

    }
}
```

```
Terminal — bash — 66×6
Familienapfel:8 nowonder$ javac ExceptionPropagation.java
Familienapfel:8 nowonder$ java ExceptionPropagation
Fanget i func2
Familienapfel:8 nowonder$
```