

# DM502

Peter Schneider-Kamp (petersk@imada.sdu.dk)

<http://imada.sdu.dk/~petersk/DM502/>



# DM502

- Bog, ugesedler og noter
  - ”De første øvelser”
- Let for nogen, svært for andre
  - Kom til øvelserne!
  - Lav opgaverne!
- Problemer?
  - Konto på IMADAs system
  - Kursets hjemmeside
  - Holdfordelingsproblemer

# Kursets form

- Forelæsninger
  - Ca. 2 gange 2 timer per uge.
  - Præsenterer og demonstrerer nyt stof
  - Vil ofte være korte
- Øvelser
  - Ca. 2 gange 2 timer per uge
  - I løser og præsenterer opgaver
  - I arbejder med opgaver
- Projekt-eksamen i 2 dele



# De første uger

- De primære mål er:
  - Alle kommer i gang med at skrive, køre og fejlfinde programmer
  - Alle lærer den basale Java-syntaks
  - Alle bliver klar til at lave den første projekt-opgave

# Brug af IMADAs system

- Det er ikke et krav at I sidder i terminalrummet og arbejder når I løser opgaverne, men
  - Det er et krav at de programmer I afleverer kan køre på IMADAs system
- Dvs
  - Enten arbejder I i terminalrummet
  - Ellers tester I i terminalrummet

# Programmeringsprocessen

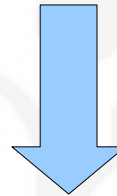
- Hvad er et program?
  - Instruktioner til computeren (den er dum :-)
  - Input → Beregning → Output

# Programmering

- Hvad er programmering?
  - Problemløsning på computer
- Eksempler på problemer man (vi) kan løse:
  - SuDoKu
  - Ordforråd i en tekst (Lix)
  - Korteste vej fra A til B ([maps.google.dk](https://maps.google.dk))
  - Pakning af en flyttebil
  - Osv...
- Fælles: Model, abstraktion og struktur

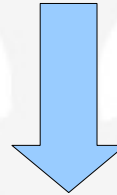
# Programmering

Opgave / kundens problem



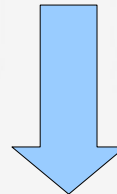
Problemanalyse

Program-specifikation (hvad?)



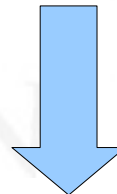
Datastrukturer + algoritmer

Program-design (hvordan?)



Implementering (i fx Java)

Implementation



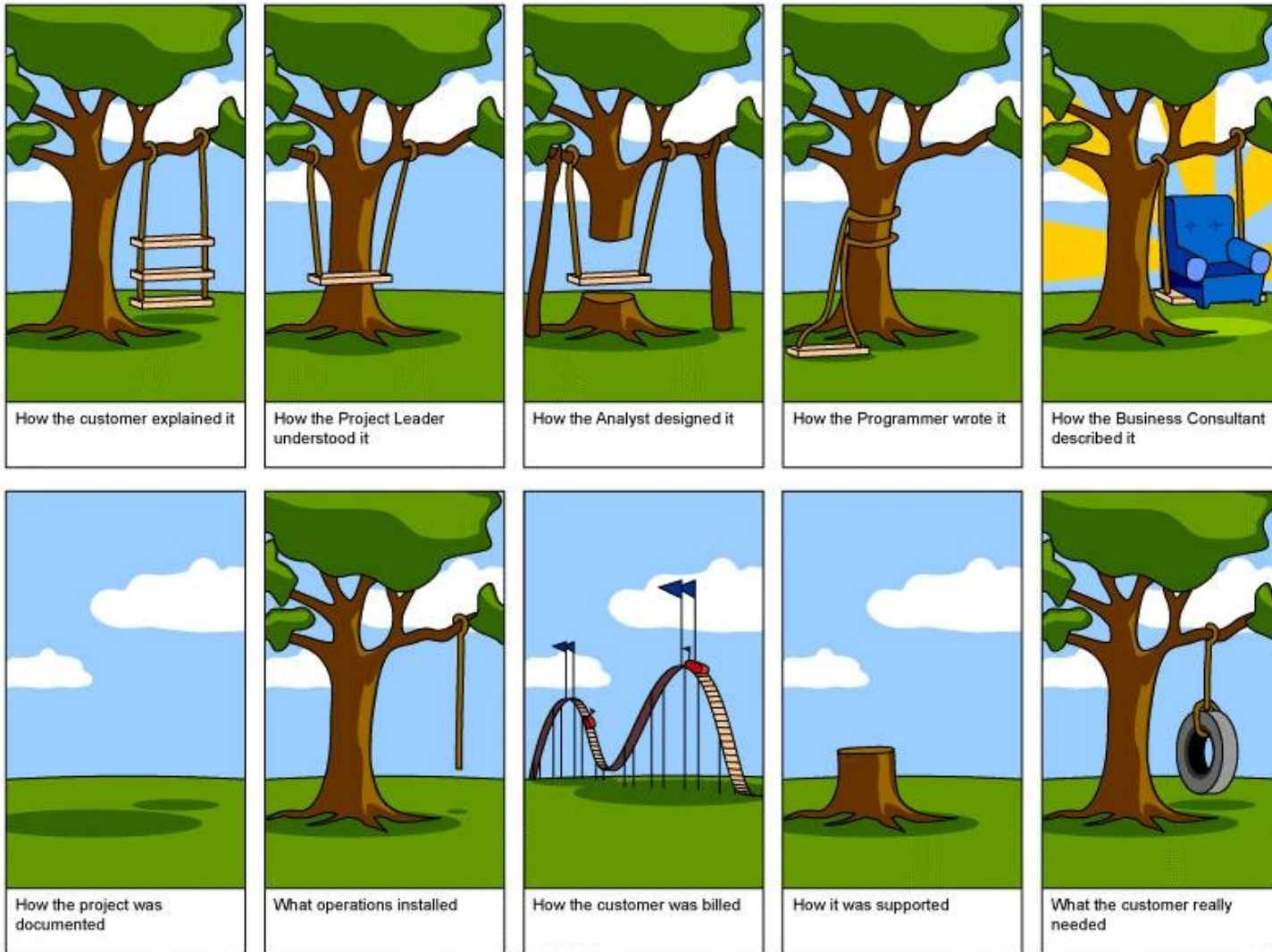
Afprøvning og indkøring

Program / produkt





# "Programming"



# Programmering i DM502

- Problemorienteret
- Bruger emner fra hele bogen straks
- Interaktiv
  - Online modellering og udvikling
  - Afprøvning vha. debugger (Jswat)
- Forklarer for lidt og for meget
- Programmerne er små, men tænk på:
  - Programmering for andre
  - Programmering med andre

# Programmering er SIMPELT

## Simple instruktioner, simpelt data

- Sekvens: I1, I2, I3, ...
- Gentagelse af sekvens: Gentag  
I1, I2, I3, ...  
Indtil "betingelse"
- Betinget sekvens: Hvis "betingelse" da  
I1, I2, I3, ...  
Ellers  
J1, J2, J3
- Delprogrammer (funktioner): Resultat = funktion(input)
- Data: Skalarer, Klasser / objekter

# HelloWorld.java

```
// HelloWorld.java
// Author: Peter Schneider-Kamp (+ millions)

public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

# HelloWorld.java

```
// HelloWorld.java
// Author: Peter Schneider-Kamp (+ millions)

public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

# HelloWorld.java

```
// HelloWorld.java
// Author: Peter Schneider-Kamp (+ millions)

public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

# HelloWorld.java

```
// HelloWorld.java
// Author: Peter Schneider-Kamp (+ millions)

public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

# HelloWorld.java

```
// HelloWorld.java
// Author: Peter Schneider-Kamp (+ millions)

public class HelloWorld {

    public static void main(String[] args) {

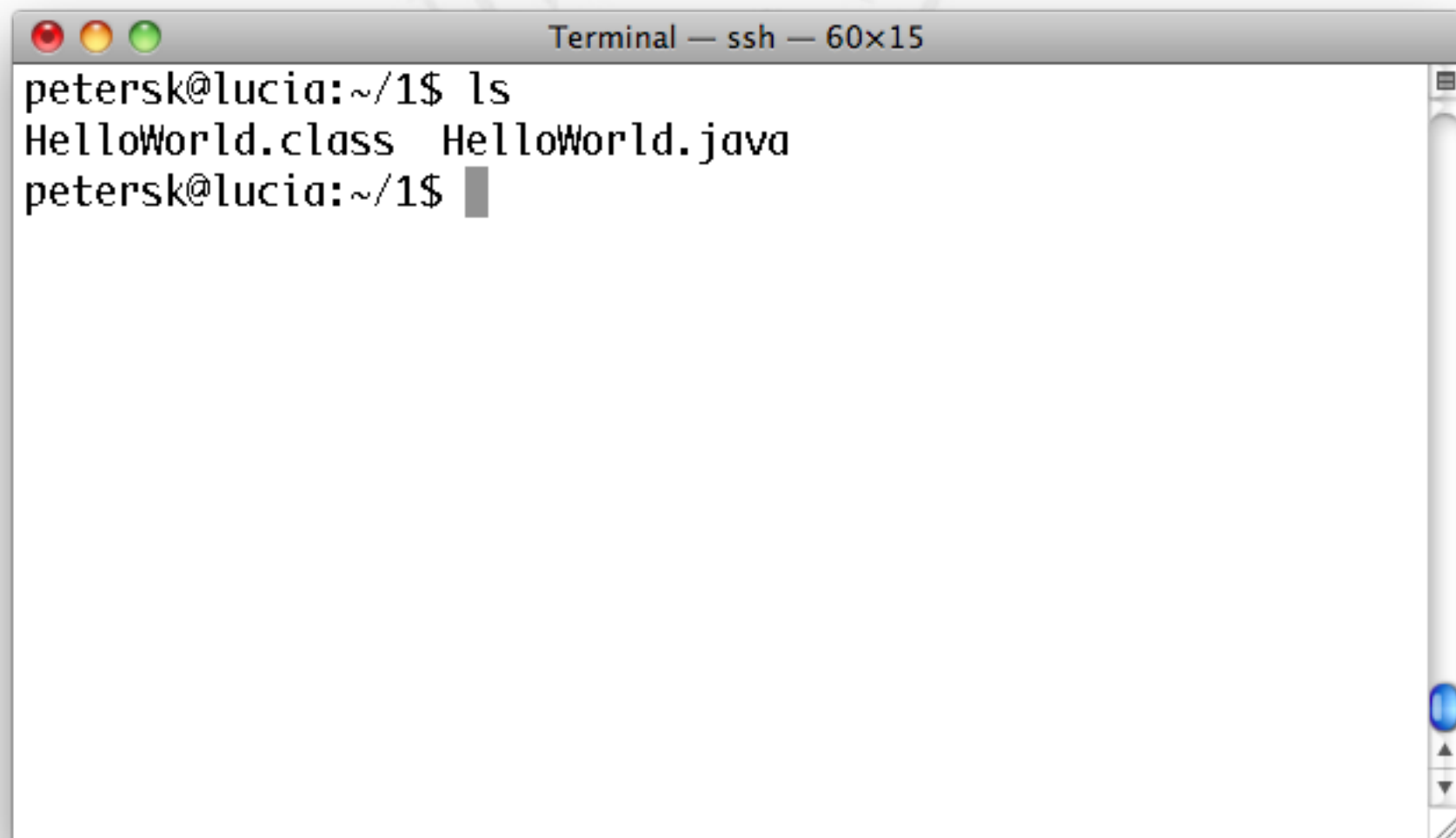
        System.out.println("Hello World!");

    }

}
```



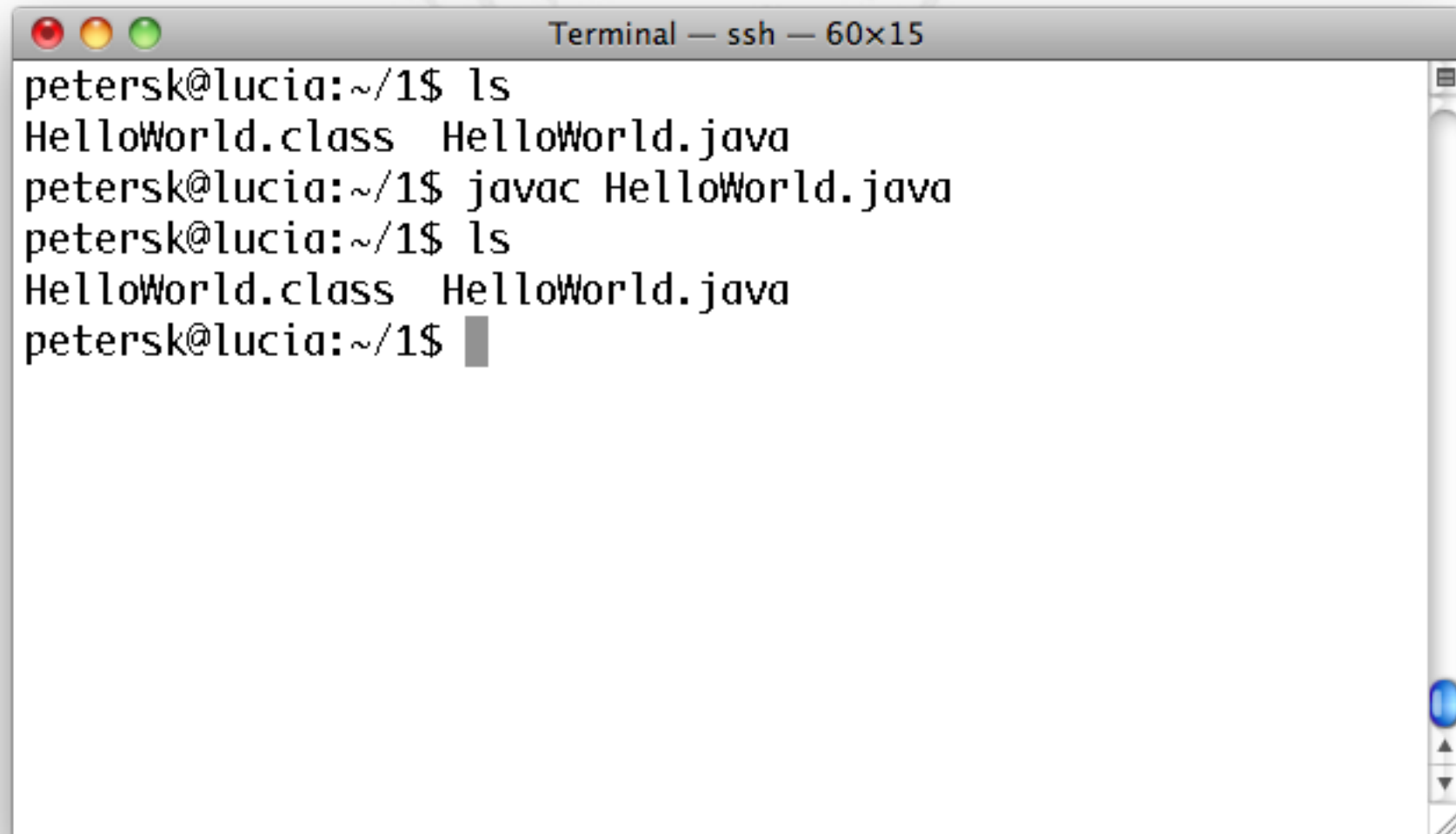
# HelloWorld.java

A terminal window titled "Terminal — ssh — 60x15" with standard macOS window controls (red, yellow, green buttons). The terminal shows the user "petersk@lucia" in the directory "~/1" running the "ls" command. The output lists "HelloWorld.class" and "HelloWorld.java". The prompt returns to "petersk@lucia:~/1\$" with a cursor.

```
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ █
```



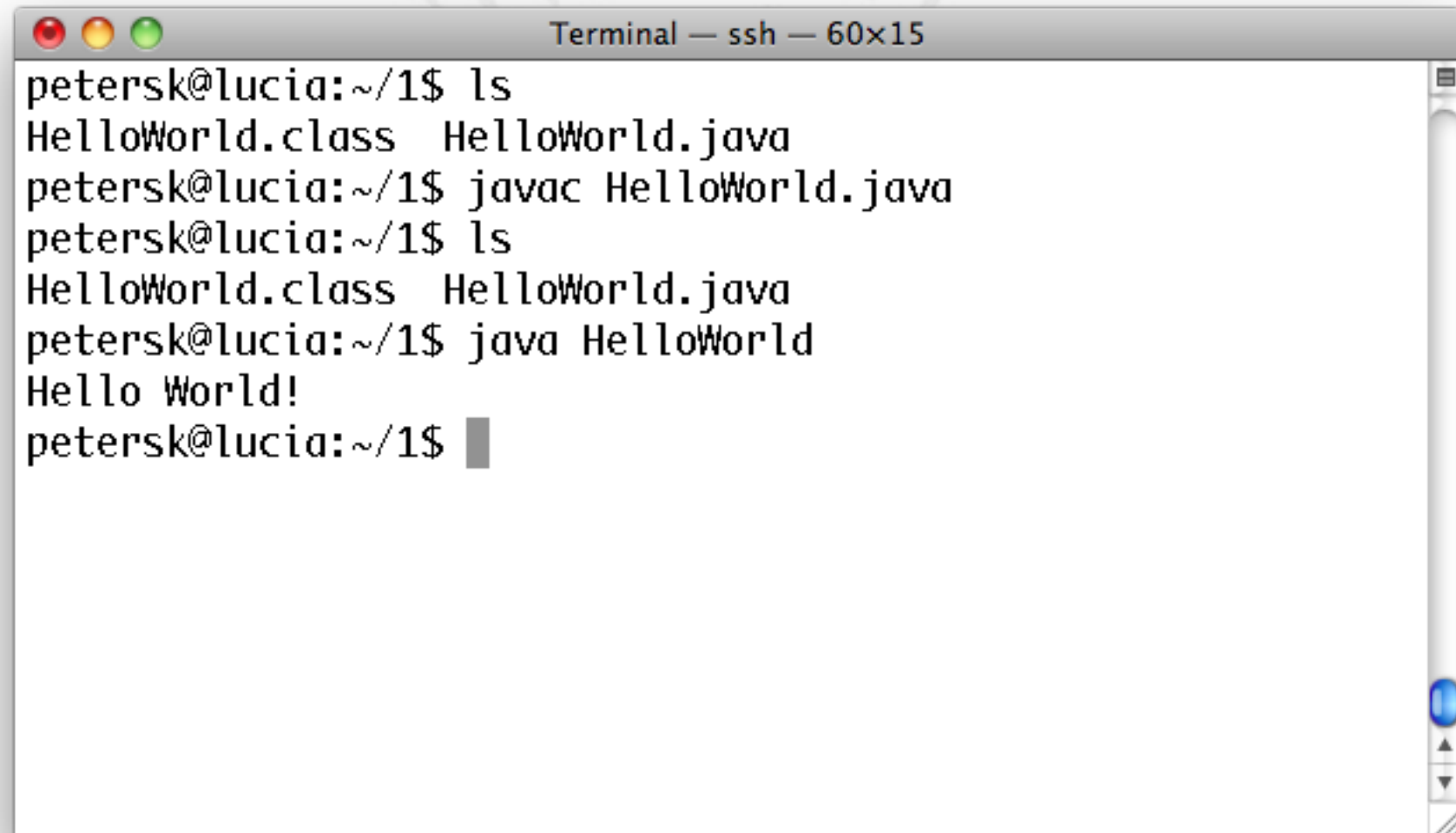
# HelloWorld.java



```
Terminal — ssh — 60x15
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ javac HelloWorld.java
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ █
```

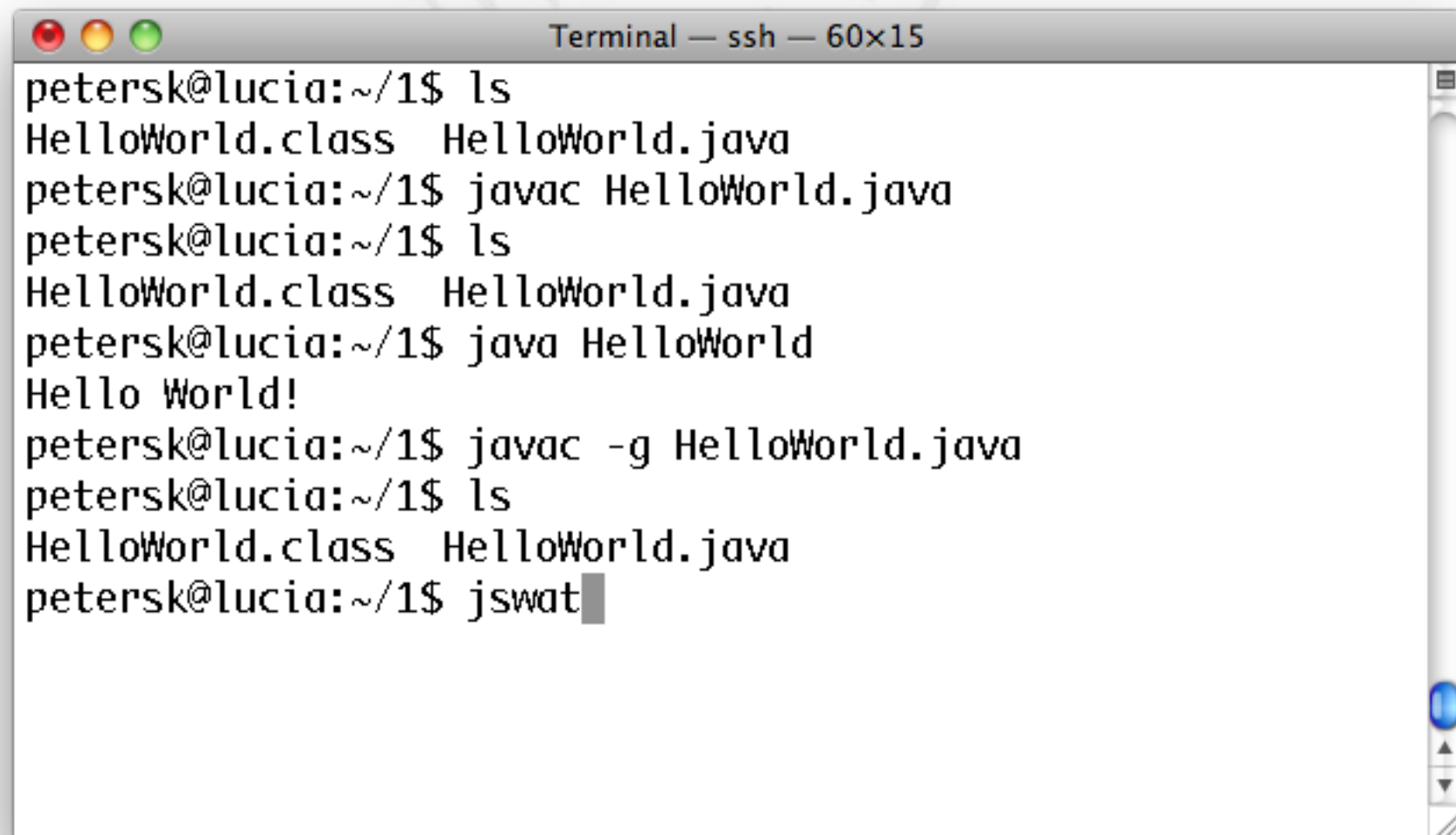


# HelloWorld.java



```
Terminal — ssh — 60x15
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ javac HelloWorld.java
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ java HelloWorld
Hello World!
petersk@lucia:~/1$
```

# HelloWorld.java



```
Terminal — ssh — 60x15
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ javac HelloWorld.java
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ java HelloWorld
Hello World!
petersk@lucia:~/1$ javac -g HelloWorld.java
petersk@lucia:~/1$ ls
HelloWorld.class HelloWorld.java
petersk@lucia:~/1$ jswat
```

# HelloWorld.java

The screenshot shows the JSwat Debugger interface. The main window displays the source code of HelloWorld.java, which is paused at line 6:1. The code is as follows:

```
// HelloWorld.java
// Author: Peter Schneider-Kamp (+millions)

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The debugger's left sidebar shows the following components:

- Runtime:** A table with columns for Name and Status. It lists the system, main, and various system threads like Signal Dispatcher, Reference Handler, and Finalizer.
- Classes:** A panel for managing loaded classes.
- Threads:** A panel for managing threads, showing the main thread as running.
- Call Stack - main:** A panel showing the current call stack, with the top entry being HelloWorld.main:6.
- Breakpoints:** A panel showing a list of breakpoints, including a line breakpoint at HelloWorld.java:6.
- Variables:** A panel for viewing variables, which is currently empty.

# Fibonacci

- Fibonacci-tallene: Summen af de to foregående tal, startende med 0 og 1
  - 0, 1, 1, 2, 3, 5, 8, ...
- Model: To tal  $a = 0$ ,  $b = 1$
- Algoritme:
  - Så længe vi vil have flere tal:

```
print a;
temp = b;
b = a + b;
a = temp;
```

# Fibonacci

- Algoritme:

```
print a;  
temp = b;  
b = a + b;  
a = temp;
```

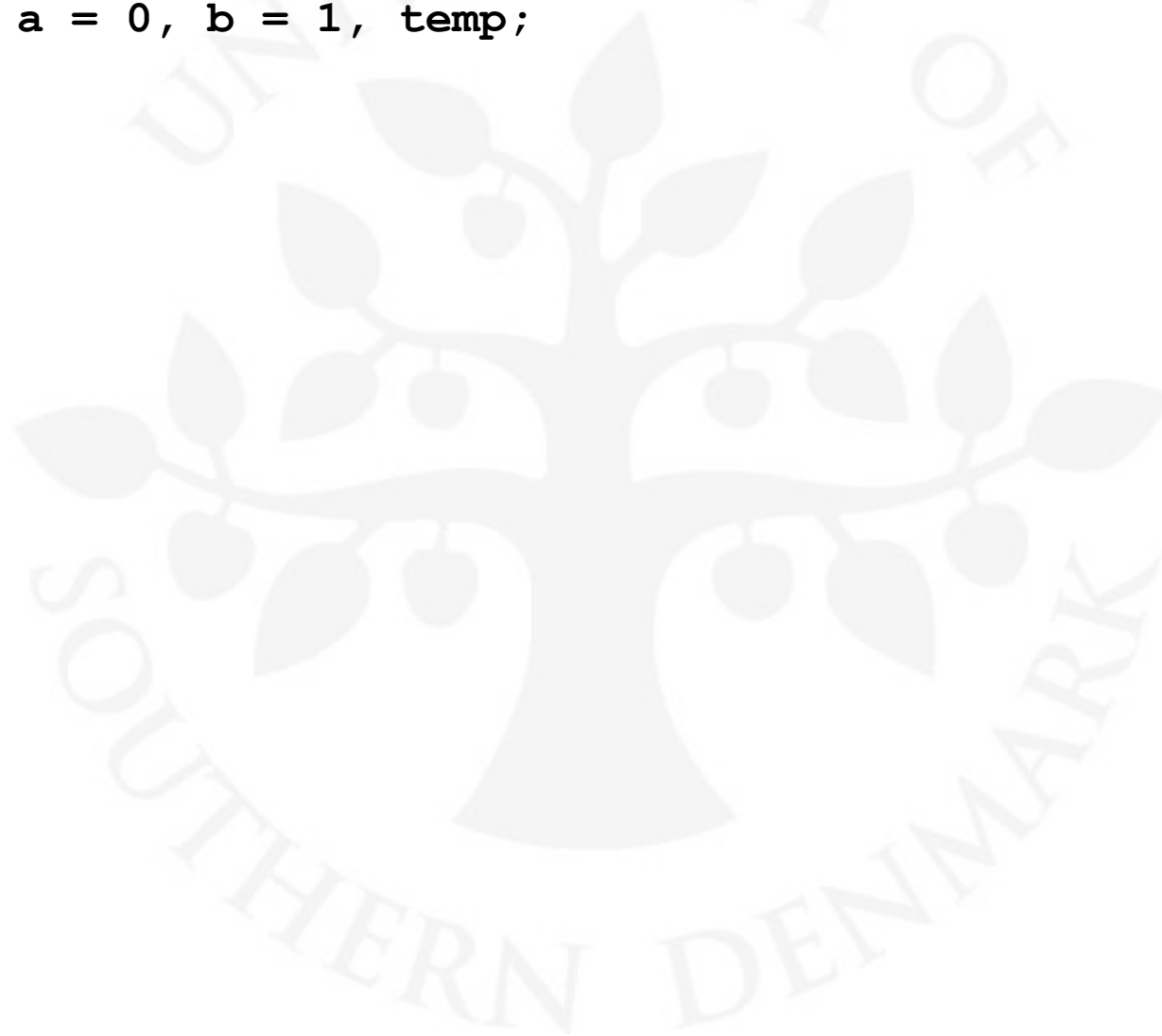
- Hvorfor ikke?:

```
b = a + b;  
a = b;
```

- Eller:

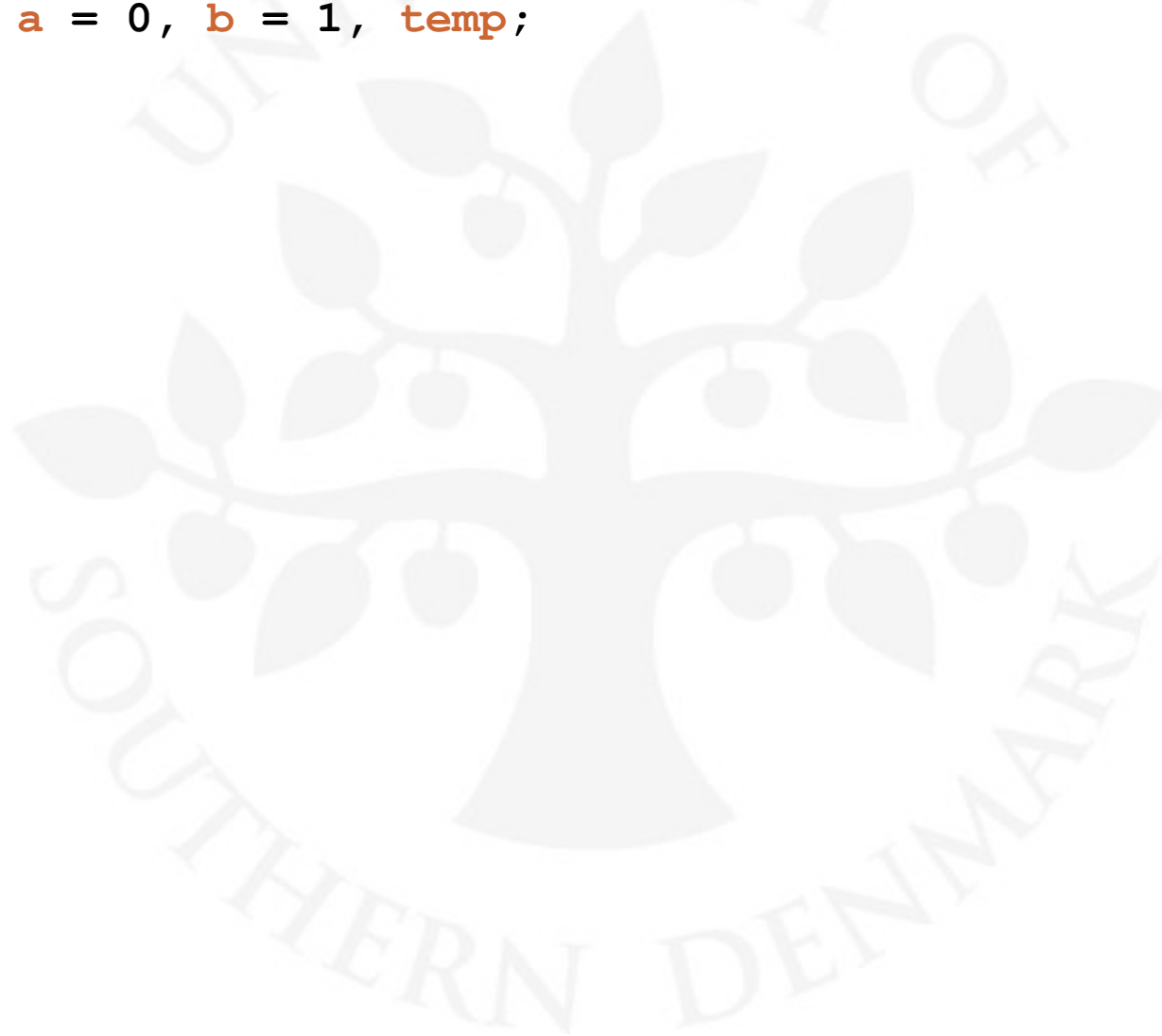
```
b = a + b;  
a = b - a;
```

```
public class Fibonacci {  
    public static void main( String[] args ) {  
        int a = 0, b = 1, temp;
```





```
public class Fibonacci {  
    public static void main( String[] args ) {  
        int a = 0, b = 1, temp;
```



```
public class Fibonacci {
    public static void main( String[] args ) {
        int a = 0, b = 1, temp;

        while( a < 10000 ) {
            System.out.print( a + " " );
            temp = b;
            b = a+b;
            a = temp;
        }
    }
}
```

```
public class Fibonacci {  
    public static void main( String[] args ) {  
        int a = 0, b = 1, temp;  
  
        while( a < 10000 ) {  
            System.out.print( a + " " );  
            temp = b;  
            b = a+b;  
            a = temp;  
        }  
    }  
}
```

```
public class Fibonacci {
    public static void main( String[] args ) {
        int a = 0, b = 1, temp;

        while( a < 10000 ) {
            System.out.print( a + " " );
            temp = b;
            b = a+b;
            a = temp;
        }

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

# Fibonacci

- Oversættelse: `javac Fibonacci.java`
- Kørsel: `java Fibonacci`
- Oversættelse til debugging: `javac -g Fibonacci.java`
- Debug med Jswat
- Fejl og mangler?

# Java Class Library

- contains standard data structures and algorithms
- contains interfaces to Everything™
- <http://download-llnw.oracle.com/javase/6/docs/api/>