Practical Concurrent and Parallel Programming XI

Java Networking & Introduction to Erlang

Raúl Pardo and Jørgen Staunstrup

Agenda

- Networking (general)
- Java sockets
- Internet protocols and JSON
- Erlang
  - The shell
  - Datatypes
  - Conditional statements
  - Pattern matching
  - Errors and exceptions
  - No loops (recursion)
  - Some useful data structures
  - A larger example

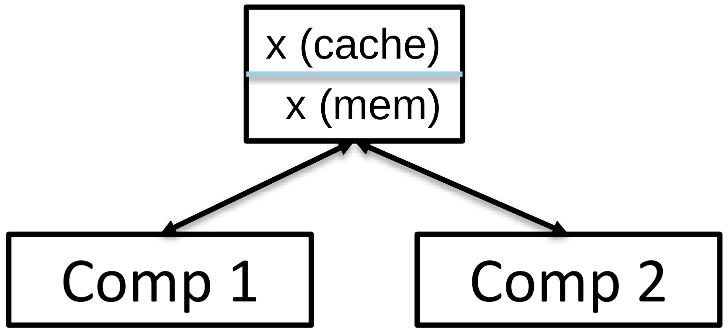
Message passing vs. shared memory

Two mental models for coordinating concurrent computations

Message passing vs. shared memory

Two mental models for coordinating concurrent computations

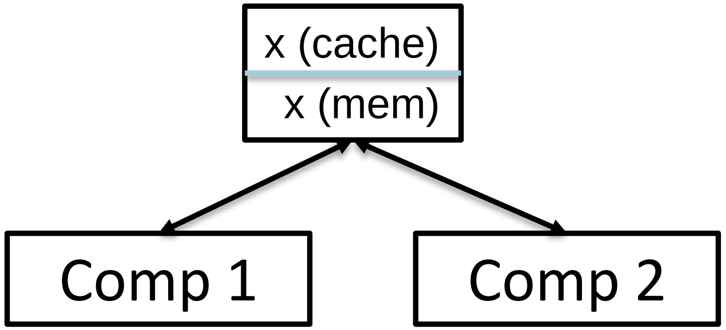
Shared memory



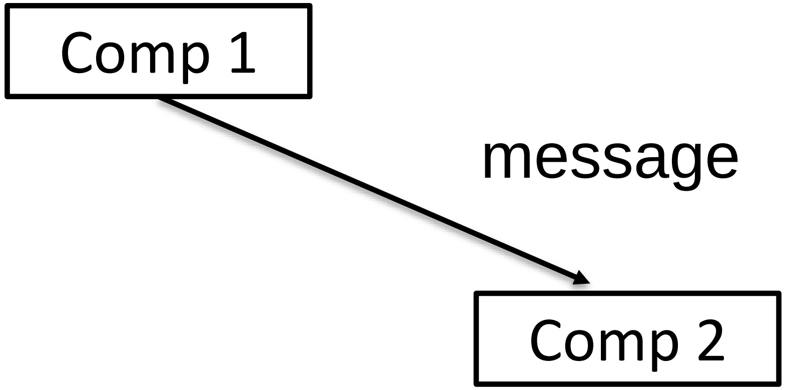
Message passing vs. shared memory

Two mental models for coordinating concurrent computations

Shared memory



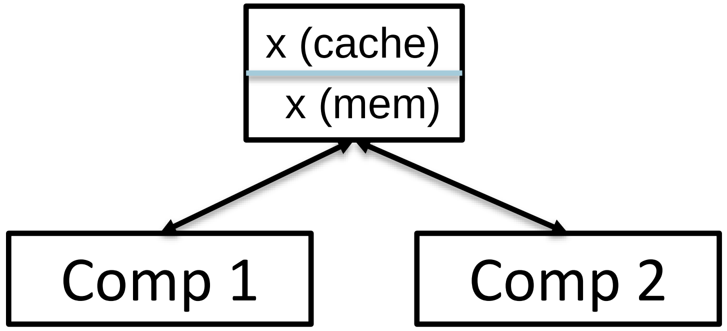
Message passing



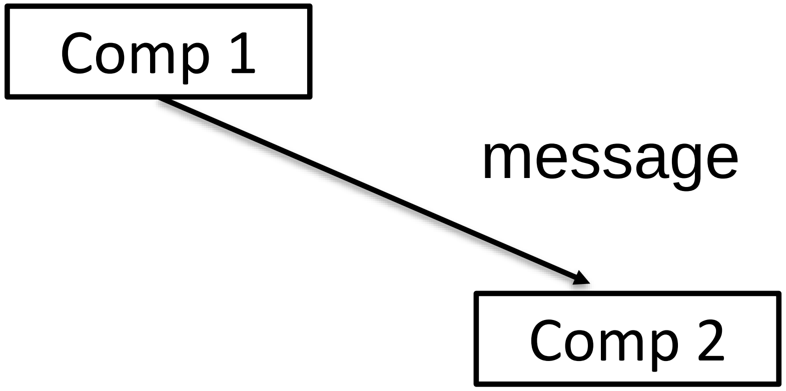
Message passing vs. shared memory

Two mental models for coordinating concurrent computations

Shared memory



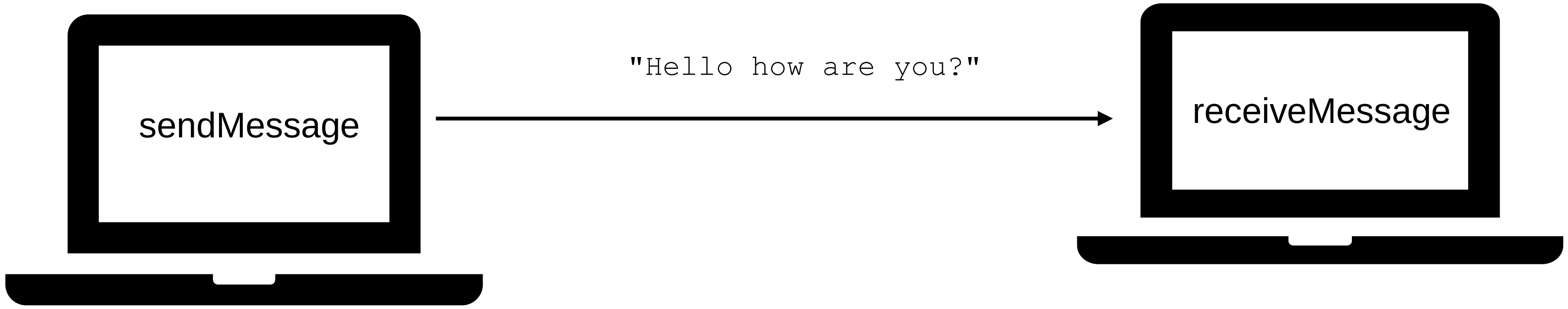
Message passing



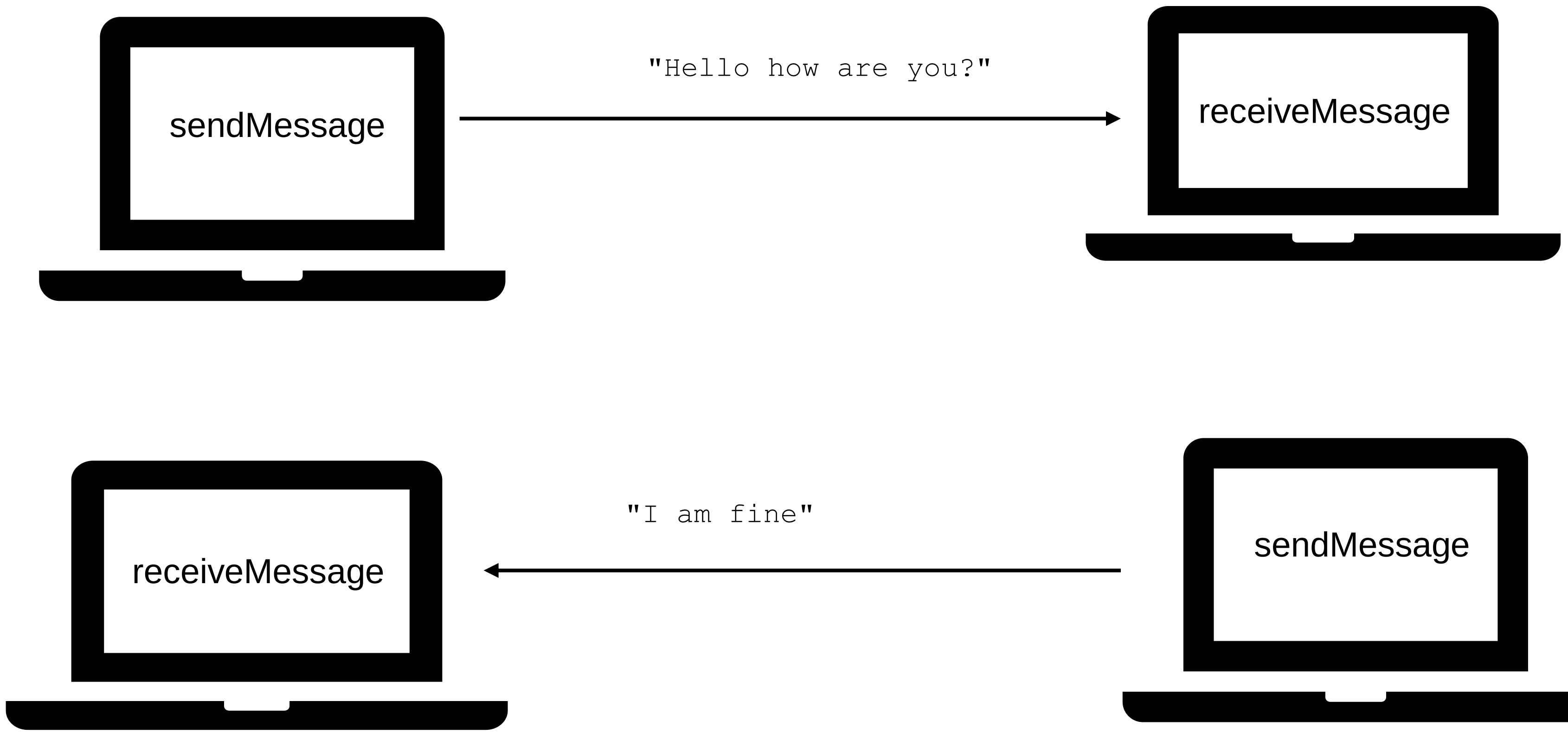
Theoretically equally powerful

each can simulate the other

Message passing on the Internet: Sockets (TCP)



Message passing on the Internet: Sockets (TCP)



Socket addressing

Addressing (IP addresses) like: 192.168.1.204

Socket addressing

Addressing (IP addresses) like: 192.168.1.204

Each computer has many independent **ports/sockets (e.g. 8080)**

Socket addressing

Addressing (IP addresses) like: 192.168.1.204

Each computer has many independent **ports/sockets (e.g. 8080)**

Socket address
192.168.1.204:8080

Addressing local sockets

Referencing sockets on local PC

Addressing local sockets

Referencing sockets on local PC

```
private final static String IP=
  "127.0.0.1";         // this PC
  //"localhost";       // this PC
```

Addressing local sockets

Referencing sockets on local PC

```
private final static String IP=
  "127.0.0.1";          // this PC
  //"localhost";        // this PC

private Socket clientSocket;
clientSocket= new Socket(IP, 8080);
```

Addressing local sockets

Referencing sockets on local PC

```
private final static String IP=
  "127.0.0.1";          // this PC
  //"localhost";        // this PC

private Socket clientSocket;
clientSocket= new Socket(IP, 8080);
```

https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

Addressing local sockets

Referencing sockets on local PC

```
private final static String IP=
  "127.0.0.1";           // this PC
  //"localhost";         // this PC

private Socket clientSocket;
clientSocket= new Socket(IP, 8080);
```

For this week's exercises both server and client are on the same PC

https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

Addressing local sockets

Referencing sockets on local PC

```
private final static String IP=
   "127.0.0.1";          // this PC
   //"localhost";        // this PC

private Socket clientSocket;
clientSocket= new Socket(IP, 8080);
```

For this week's exercises both server and client are on the same PC
(in two different windows)

https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

Addressing local sockets

Referencing sockets on local PC

```
private final static String IP=
   "127.0.0.1";        // this PC
   //"localhost";      // this PC

private Socket clientSocket;
clientSocket= new Socket(IP, 8080);
```

Use command
ipconfig
to find IP-addr
of your PC

For this week's exercises both server and client are on the same PC
(in two different windows)

https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

Java Sockets (send)

```
public class Server {







...
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages









 ...
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages

  private Socket clientSocket;        // to return responses







...
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages

  private Socket clientSocket;        // to return responses







...
  serverSocket= new ServerSocket(port);
  clientSocket= serverSocket.accept();
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages

  private Socket clientSocket;        // to return responses








...
  serverSocket= new ServerSocket(port);
  clientSocket= serverSocket.accept();
  out= new PrintWriter(clientSocket.getOutputStream(), true);
  in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages
  private BufferedReader in;
  private Socket clientSocket;         // to return responses
  private PrintWriter out;







...
  serverSocket= new ServerSocket(port);
  clientSocket= serverSocket.accept();
  out= new PrintWriter(clientSocket.getOutputStream(), true);
  in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages
  private BufferedReader in;
  private Socket clientSocket;        // to return responses
  private PrintWriter out;







...
  serverSocket= new ServerSocket(port);
  clientSocket= serverSocket.accept();
  out= new PrintWriter(clientSocket.getOutputStream(), true);
  in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

  String inputLine;
  while ((inputLine= readMessage(in)) != null) {

  ... }
```

Java Sockets (send)

```
public class Server {
  private ServerSocket serverSocket; // to receive messages
  private BufferedReader in;
  private Socket clientSocket;        // to return responses
  private PrintWriter out;

  public String readMessage(BufferedReader in) {
    try {  return in.readLine();
    } catch (IOException e) { System.out.println(e.getMessage());}
    return null;
  }
...
  serverSocket= new ServerSocket(port);
  clientSocket= serverSocket.accept();
  out= new PrintWriter(clientSocket.getOutputStream(), true);
  in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

  String inputLine;
  while ((inputLine= readMessage(in)) != null) {

  ... }
```

Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
```

Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
```

Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;



     clientSocket= new Socket(ip, port);
```

Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;



     clientSocket= new Socket(ip, port);
     out= new PrintWriter(clientSocket.getOutputStream(), true);
     in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

 public void startConnection(String ip, int port) {
   try {
     clientSocket= new Socket(ip, port);
     out= new PrintWriter(clientSocket.getOutputStream(), true);
     in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
   } catch (IOException e) {  System.out.println(e.getMessage());   }
 }
```

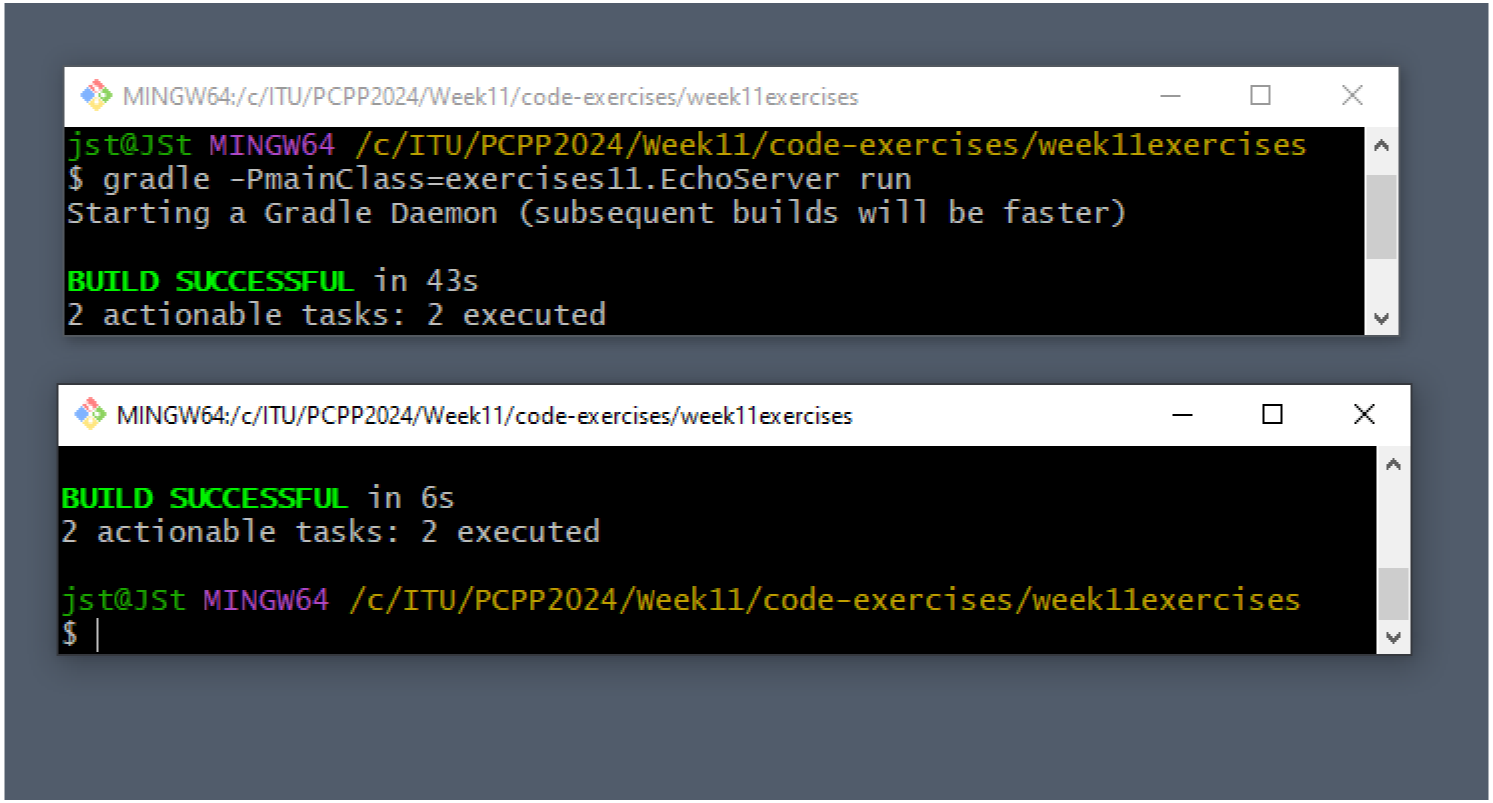
Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

 public void startConnection(String ip, int port) {
   try {
     clientSocket= new Socket(ip, port);
     out= new PrintWriter(clientSocket.getOutputStream(), true);
     in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
   } catch (IOException e) {  System.out.println(e.getMessage());   }
 }

 public String sendMessage(String msg) {
   try {
     out.println(msg);
     return in.readLine();
   } catch (Exception e) {  return null;   }
 }
```

Java Sockets (receive)

```
public class client {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

 public void startConnection(String ip, int port) {
   try {
     clientSocket= new Socket(ip, port);
     out= new PrintWriter(clientSocket.getOutputStream(), true);
     in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
   } catch (IOException e) {  System.out.println(e.getMessage());   }
 }

 public String sendMessage(String msg) {
   try {
     out.println(msg);
     return in.readLine();
   } catch (Exception e) {  return null;   }
 }
...

startConnection("127.0.0.1", 8080);
sendMessage("get")
```

Running client and server



MINGW64:/c/ITU/PCPP2024/Week11/code-exercises/week11exercises

```
jst@JSt MINGW64 /c/ITU/PCPP2024/Week11/code-exercises/week11exercises
$ gradle -PmainClass=exercises11.EchoServer run
Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 43s
2 actionable tasks: 2 executed
```

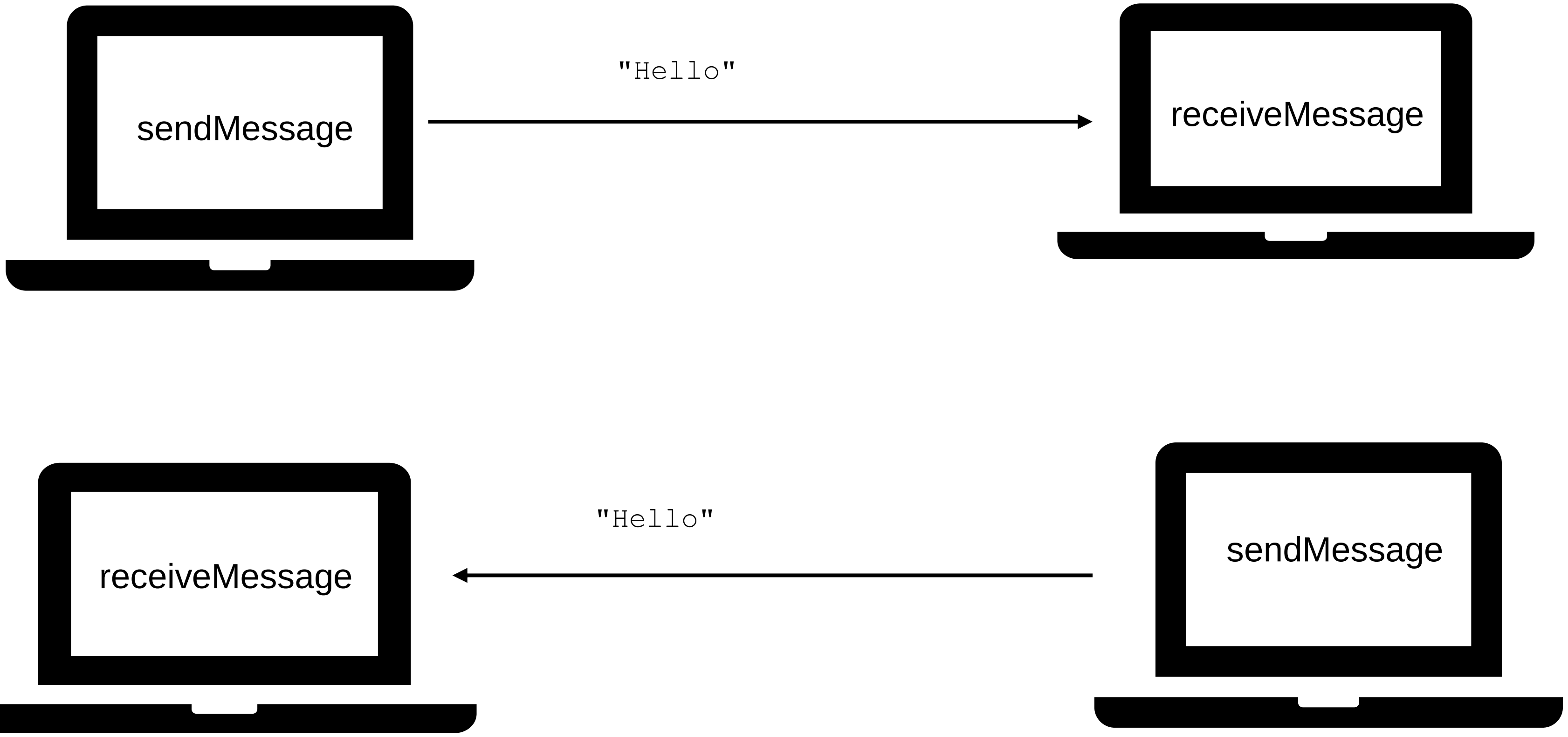
MINGW64:/c/ITU/PCPP2024/Week11/code-exercises/week11exercises

```

BUILD SUCCESSFUL in 6s
2 actionable tasks: 2 executed

jst@JSt MINGW64 /c/ITU/PCPP2024/Week11/code-exercises/week11exercises
$ |
```

You need **two** terminal windows to run both server and client

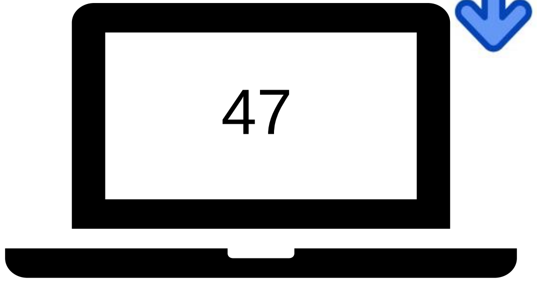
Example: EchoServer and EchoClient

complete code in: code-exercises/ …/EchoServer.java and /EchoClient.java

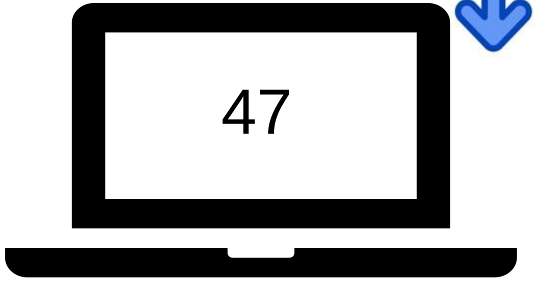


Java example: Number Server

```
public class NumberServer {
  private int count= 0;










}
```



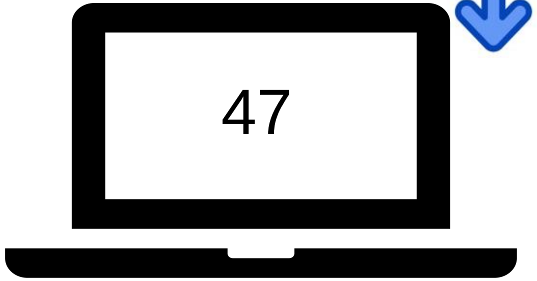
Java example: Number Server



```
public class NumberServer {
  private int count= 0;


  /*messages

    get       returns the current value of the server's number
    incr      increments the server's number by 1
    put dd    changes the server's number to dd
    stop      stops the server

  */




}
```

Java example: Number Server

```
public class NumberServer {
  private int count= 0;


  /*messages

    get      returns the current value of the server's number
    incr     increments the server's number by 1
    put dd   changes the server's number to dd
    stop     stops the server

  */

  public static void main(String[] args) {

  }
}
```



NumberServer (functionality)

```
String inputLine;
while ((inputLine= readMessage(in)) != null) {
  if ("incr".equals(inputLine)) {
     count= count+1;
     out.println(count);
  } else if ("get".equals(inputLine)) {
     out.println(count);
  } else if ("put".equals(inputLine.substring(0, 3))) {
     count= Integer.parseInt(inputLine.substring(4, inputLine.length()));
     out.println(count);
  } else if ("stop".equals(inputLine)) {
     out.println("good bye "+ count);
     stop();
     break;
  }
}
```

NumberServer (communication)

```
public class NumberServer {









  public void start(int port) {
    try {






    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;







  public void start(int port) {
    try {





    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;







  public void start(int port) {
    try {
      serverSocket= new ServerSocket(port);





    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;








  public void start(int port) {
    try {
      serverSocket= new ServerSocket(port);
      clientSocket= serverSocket.accept();

      in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));


    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;

  public String readMessage(BufferedReader in) {
    try {
      return in.readLine();
    } catch (IOException e) { System.out.println(e.getMessage());}
    return null;
  }

  public void start(int port) {
    try {
      serverSocket= new ServerSocket(port);
      clientSocket= serverSocket.accept();

      in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));


    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;

  public String readMessage(BufferedReader in) {
    try {
      return in.readLine();
    } catch (IOException e) { System.out.println(e.getMessage());}
    return null;
  }

  public void start(int port) {
    try {
      serverSocket= new ServerSocket(port);
      clientSocket= serverSocket.accept();
      out= new PrintWriter(clientSocket.getOutputStream(), true);
      in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));


    } catch (IOException e) { System.out.println(e.getMessage());}

  }
}
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;

  public String readMessage(BufferedReader in) {
    try {
      return in.readLine();
    } catch (IOException e) { System.out.println(e.getMessage());}
    return null;
  }

  public void start(int port) {
    try {
      serverSocket= new ServerSocket(port);
      clientSocket= serverSocket.accept();
      out= new PrintWriter(clientSocket.getOutputStream(), true);
      in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
      ...  // functionality --- see previous slide

    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

NumberServer (communication)

```
public class NumberServer {
  private ServerSocket serverSocket;
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
  private int count= 0;

  public String readMessage(BufferedReader in) {
    try {
      return in.readLine();
    } catch (IOException e) { System.out.println(e.getMessage());}
    return null;
  }

  public void start(int port) {
    try {
      serverSocket= new ServerSocket(port);
      clientSocket= serverSocket.accept();
      out= new PrintWriter(clientSocket.getOutputStream(), true);
      in= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
      ...  // functionality --- see previous slide

    } catch (IOException e) { System.out.println(e.getMessage());}

  }
```

Complete code is in:
code-exercises/ …
NumberServer.java

Ports

The server will read messages one at a time from a specific port.

Ports

```
public static void main(String[] args) {
    new NumberServer().start(8080);
}
```

The server will read messages one at a time from a specific port.

Ports

```
public static void main(String[] args) {
    new NumberServer().start(8080);
}
```

The server will read messages one at a time from a specific port.

Different ports can be used to differentiate different message types

Ports

```
public static void main(String[] args) {
    new NumberServer().start(8080);
}
```

The server will read messages one at a time from a specific port.

Different ports can be used to differentiate different message types

https://www.techtarget.com/searchnetworking/definition/port-number

Number client

```
public class NumberClient {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;
```

Number client

```
public class NumberClient {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

 public String sendMessage(String msg) {
   try {
     out.println(msg);
     return in.readLine();
   } catch (Exception e) {  return null;   }
 }
```

Number client

```
public class NumberClient {
   private Socket clientSocket;
   private PrintWriter out;
   private BufferedReader in;

  public String sendMessage(String msg) {
     try {
        out.println(msg);
        return in.readLine();
     } catch (Exception e) {  return null;    }
  }
   ...
      sendMessage("get")
```

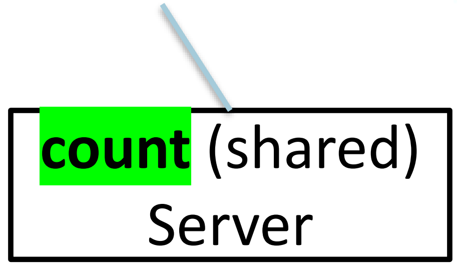
Number client

```
public class NumberClient {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

 public String sendMessage(String msg) {
   try {
     out.println(msg);
     return in.readLine();
   } catch (Exception e) {  return null;   }
 }
  ...
    sendMessage("get")

    sendMessage("put&"+1);
```

Number client

```
public class NumberClient {
  private Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

 public String sendMessage(String msg) {
   try {
     out.println(msg);
     return in.readLine();
   } catch (Exception e) {  return null;   }
 }
  ...
    sendMessage("get")

    sendMessage("put&"+1);

    sendmessage("incr");
```

Counting server



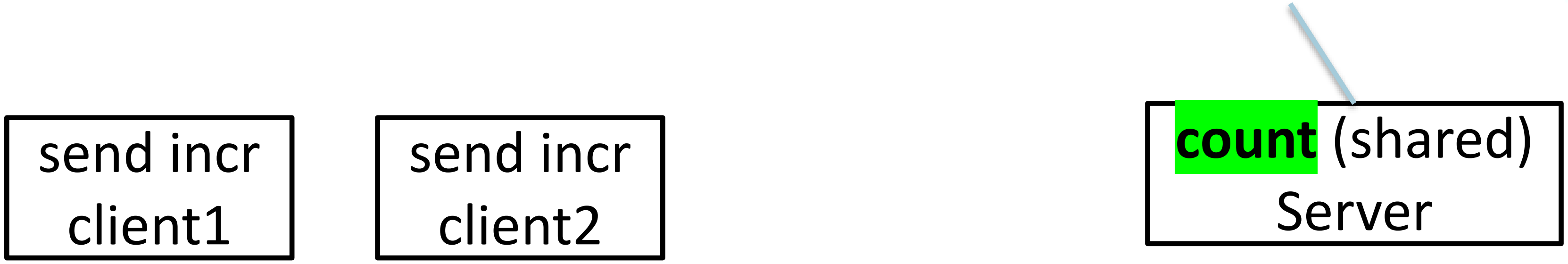
The count is similar to a volatile int

Counting server

send incr
client1
count (shared)
Server

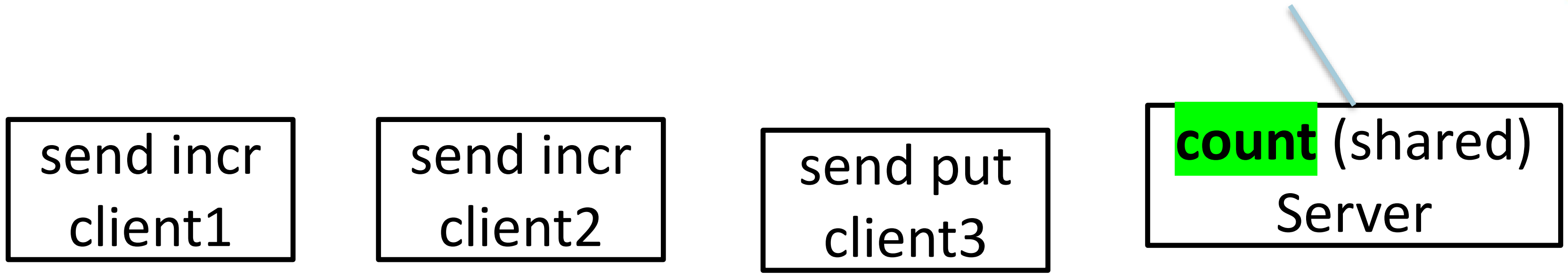
The count is similar to a volatile int

Counting server



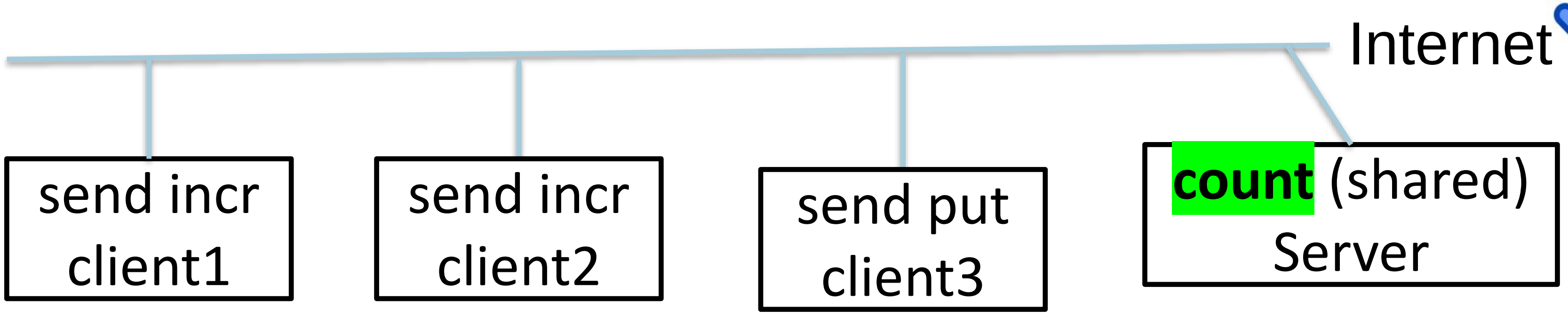
The count is similar to a volatile int

Counting server



The count is similar to a volatile int

Counting server



The count is similar to a volatile int

SocketCountingThreads

Experimenting with the shared counter:

1. Clients increments locally

```
int c=
  Integer.parseInt(sendMessage("get"));
c= c+1;
sendMessage("put&"+c);
```

2. Server locking (~volatile)

```
sendMessage(incr);
```

3. Clients and server on same PC

4. Clients and server on different PCs
   (local network)

Various observations

Various observations

```
Clients increment locally using two messages (no locking)

Run-time:    ~ 41 mS        Lots of increments lost on server
```

Various observations

```
Clients increment locally using two messages (no locking)

Run-time:    ~ 41 mS        Lots of increments lost on server
```

```
Increment counter on server (server locking)

Run-time:    ~  22 mS        No increments lost
```

Various observations

```
Clients increment locally using two messages (no locking)

Run-time:    ~ 41 mS        Lots of increments lost on server



Synchronized increment counter on client (client locking)

Run-time:    ~ 4.5 mS        No increments lost


Increment counter on server (server locking)

Run-time:    ~  22 mS        No increments lost
```

Various observations

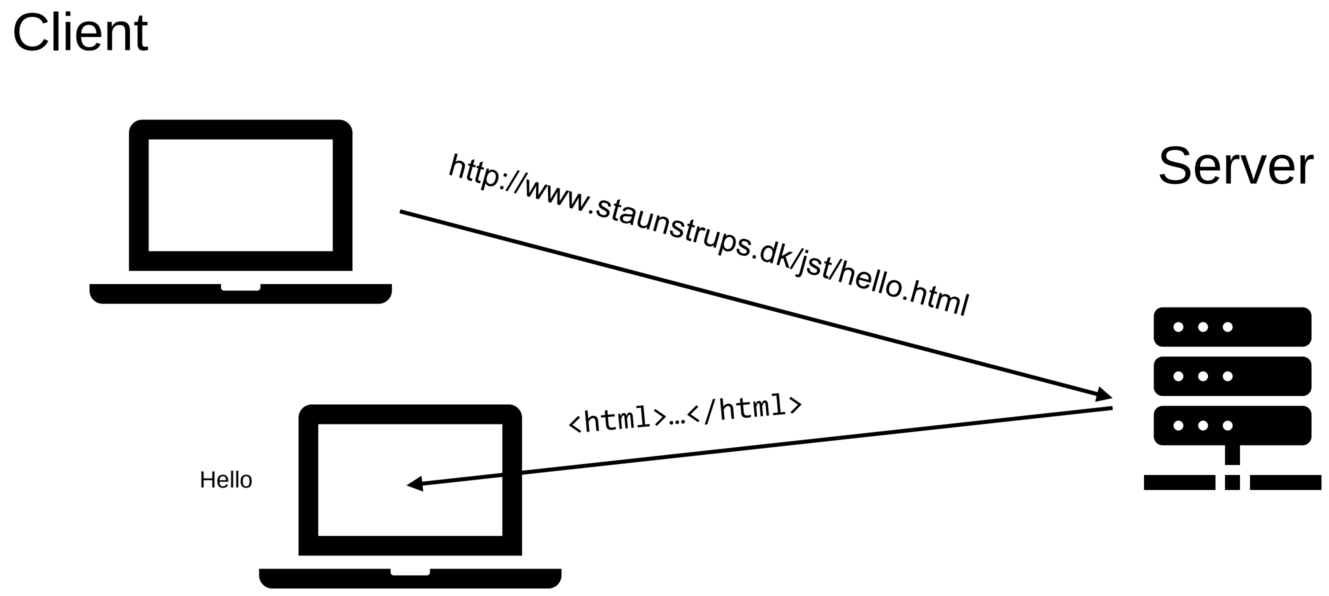
```
Clients increment locally using two messages (no locking)

Run-time:    ~ 41 mS       Lots of increments lost on server



Synchronized increment counter on client (client locking)

Run-time:    ~ 4.5 mS       No increments lost


Increment counter on server (server locking)

Run-time:    ~  22 mS       No increments lost


Increment a local counter ( sort of non-volatile) ~ 0.8 mS
```

Addressing (server)

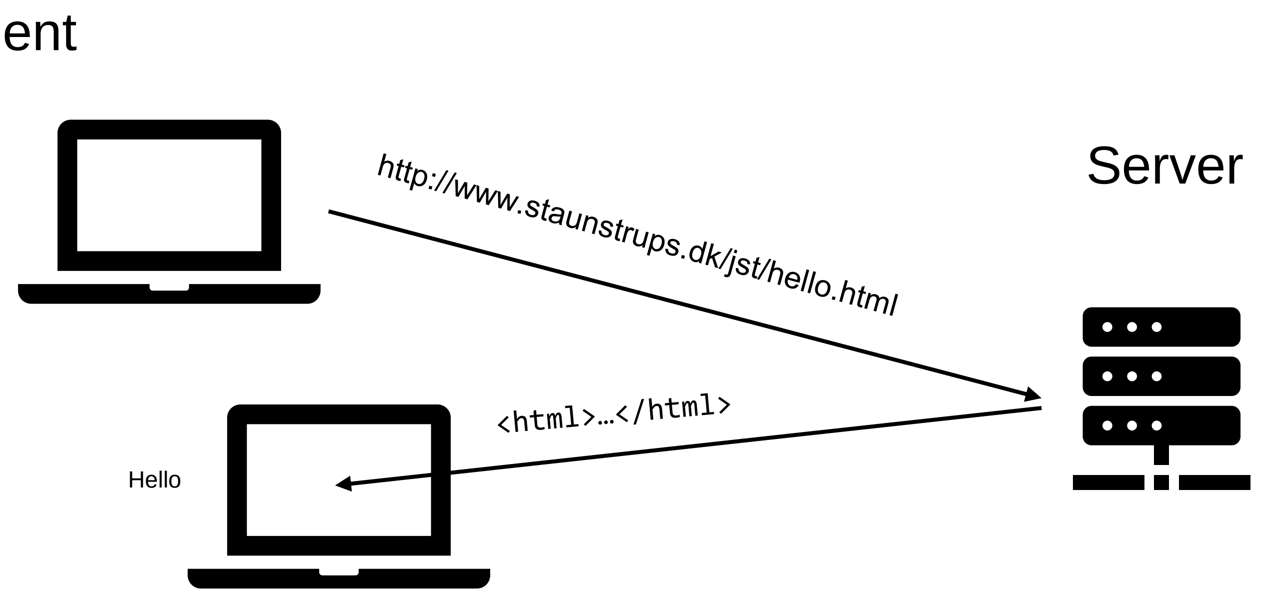
```
private final static String URL=
  //"127.0.0.1";      // this PC
  //"localhost";      // this PC
  //"192.168.1.204"; // other PC onlocal network
  "XPS-13";           // other PC onlocal network (hostname)
```

```
Increment counter on server (server locking)

Run-time (localhost):   ~  22 mS        No increments lost
Run-time (local wifi):  ~ 245 mS        No increments lost
```

HTTP protocol

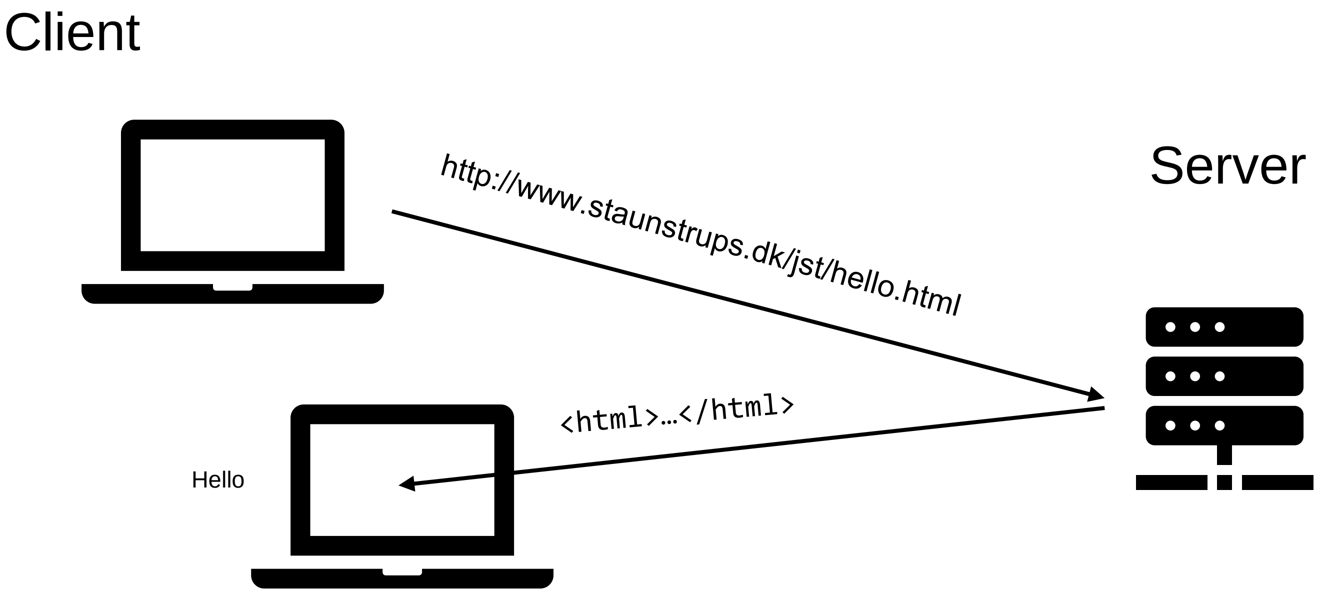


HTTP protocol



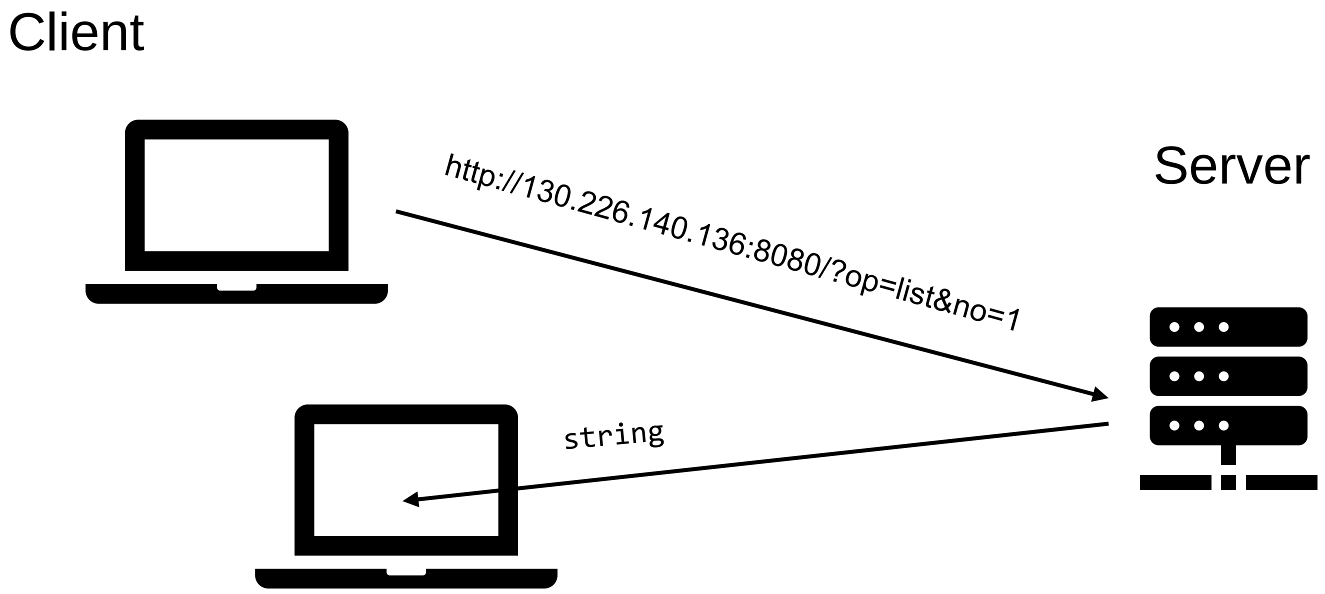
HTTP is asymmetric: **only the client can initiate communication**

HTTP protocol

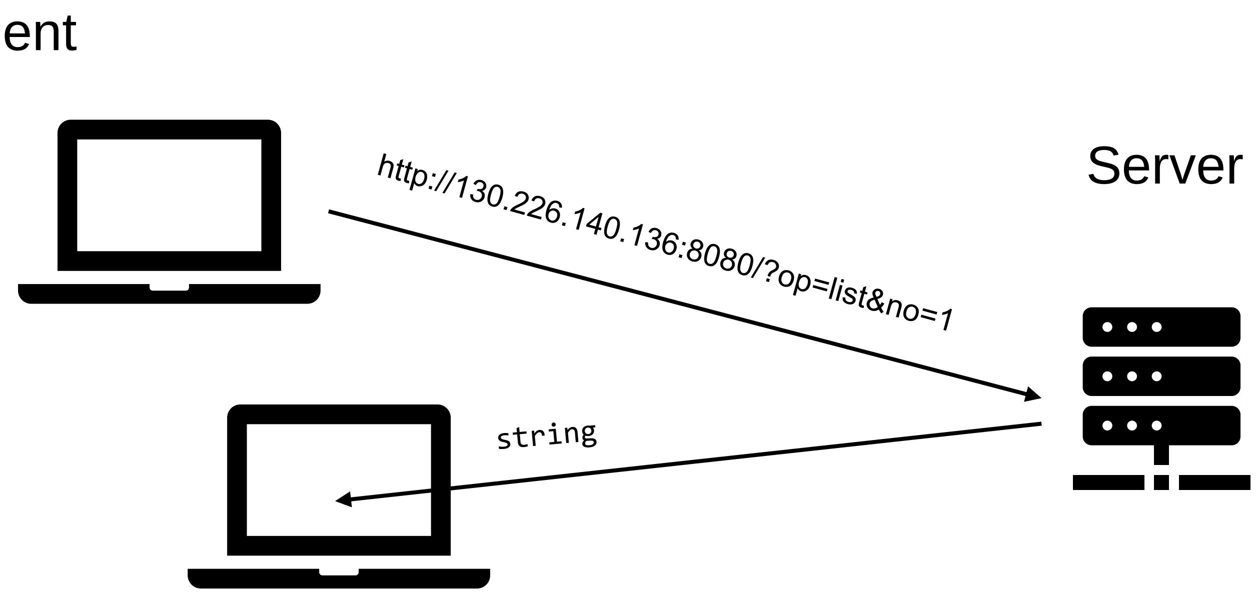


HTTP is asymmetric: **only the client can initiate communication**
and the server forgets the request when the answer has been sent

HTTP (AnswerServer)

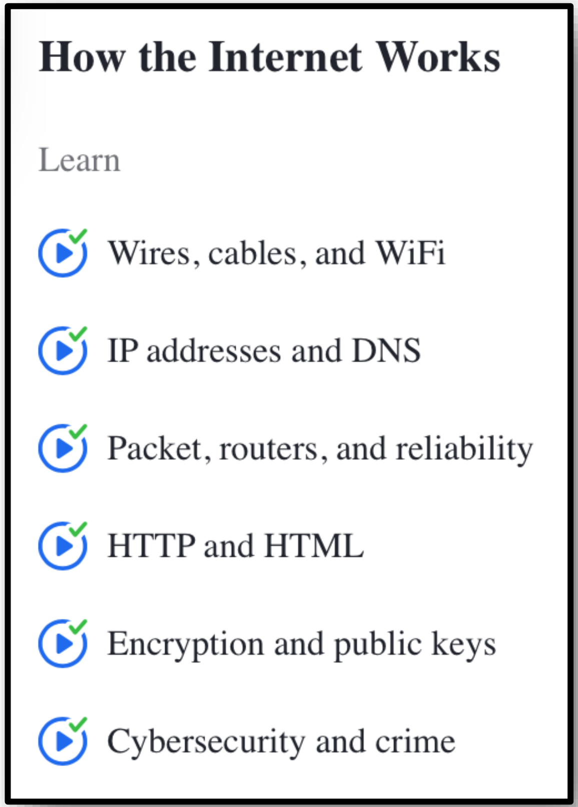


HTTP (AnswerServer)



The server returns a plain list

Khan Academy (code.org)


How the Internet Works

Learn

- Wires, cables, and WiFi
- IP addresses and DNS
- Packet, routers, and reliability
- HTTP and HTML
- Encryption and public keys
- Cybersecurity and crime

Excellent videoes explaining how the internet works

https://www.khanacademy.org/partner-content/code-org/internet-works

NetworkFetcher

Fetching an HTTP page

```
public class NetworkFetcherT {
  private static final String TAG= "NetworkFetchr";
  public byte[] getUrlBytes(String urlSpec) throws IOException {
    URL url= new URL(urlSpec);




















  }
```

NetworkFetcher

Fetching an HTTP page

```
public class NetworkFetcherT {
  private static final String TAG= "NetworkFetchr";
  public byte[] getUrlBytes(String urlSpec) throws IOException {
    URL url= new URL(urlSpec);
    HttpURLConnection connection= (HttpURLConnection)url.openConnection();
    try {









    } finally {
      connection.disconnect();
    }
  }
```

Fetching an HTTP page

```
public class NetworkFetcherT {
  private static final String TAG= "NetworkFetchr";
  public byte[] getUrlBytes(String urlSpec) throws IOException {
    URL url= new URL(urlSpec);
    HttpURLConnection connection= (HttpURLConnection)url.openConnection();
    try {
      ByteArrayOutputStream out= new ByteArrayOutputStream();
      InputStream in= connection.getInputStream();


    } finally {
      connection.disconnect();
    }
  }
```

NetworkFetcher

Fetching an HTTP page

```
public class NetworkFetcherT {
  private static final String TAG= "NetworkFetchr";
  public byte[] getUrlBytes(String urlSpec) throws IOException {
    URL url= new URL(urlSpec);
    HttpURLConnection connection= (HttpURLConnection)url.openConnection();
    try {
      ByteArrayOutputStream out= new ByteArrayOutputStream();
      InputStream in= connection.getInputStream();




      int bytesRead= 0;
      byte[] buffer= new byte[1024];
      while ((bytesRead = in.read(buffer)) > 0) {
        out.write(buffer, 0, bytesRead);
      }


    } finally {
      connection.disconnect();
    }
  }
```

NetworkFetcher

Fetching an HTTP page

```
public class NetworkFetcherT {
  private static final String TAG= "NetworkFetchr";
  public byte[] getUrlBytes(String urlSpec) throws IOException {
    URL url= new URL(urlSpec);
    HttpURLConnection connection= (HttpURLConnection)url.openConnection();
    try {
      ByteArrayOutputStream out= new ByteArrayOutputStream();
      InputStream in= connection.getInputStream();
      if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
        throw new IOException(connection.getResponseMessage() +
            ": with " +  urlSpec);     }
      int bytesRead= 0;
      byte[] buffer= new byte[1024];
      while ((bytesRead = in.read(buffer)) > 0) {
        out.write(buffer, 0, bytesRead);
      }


    } finally {
      connection.disconnect();
    }
  }
```

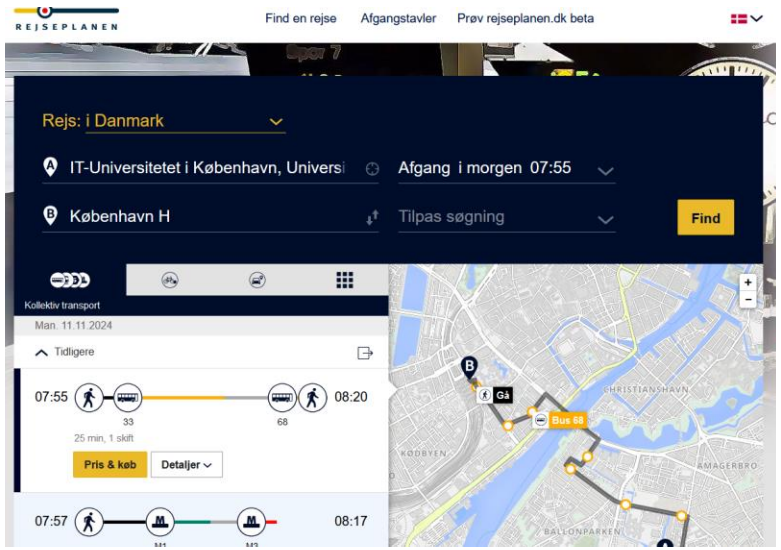
NetworkFetcher

Fetching an HTTP page

```
public class NetworkFetcherT {
  private static final String TAG= "NetworkFetchr";
  public byte[] getUrlBytes(String urlSpec) throws IOException {
    URL url= new URL(urlSpec);
    HttpURLConnection connection= (HttpURLConnection)url.openConnection();
    try {
      ByteArrayOutputStream out= new ByteArrayOutputStream();
      InputStream in= connection.getInputStream();
      if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
        throw new IOException(connection.getResponseMessage() +
            ": with " +  urlSpec);      }
      int bytesRead= 0;
      byte[] buffer= new byte[1024];
      while ((bytesRead = in.read(buffer)) > 0) {
        out.write(buffer, 0, bytesRead);
      }
      out.close();
      return out.toByteArray();
    } finally {
      connection.disconnect();
    }
  }
```

code-exercises/…/NetworkFetcher

Your personal "Rejseplan"


REJSEPLANEN

Find en rejse | Afgangstavler | Prøv rejseplanen.dk beta

Rejs: i Danmark

IT-Universitetet i København, Universi

Afgang i morgen 07:55

København H

Tilpas søgning

Find

Kollektiv transport

Man. 11.11.2024

Tidligere

07:55 — 33 — 68 — 08:20

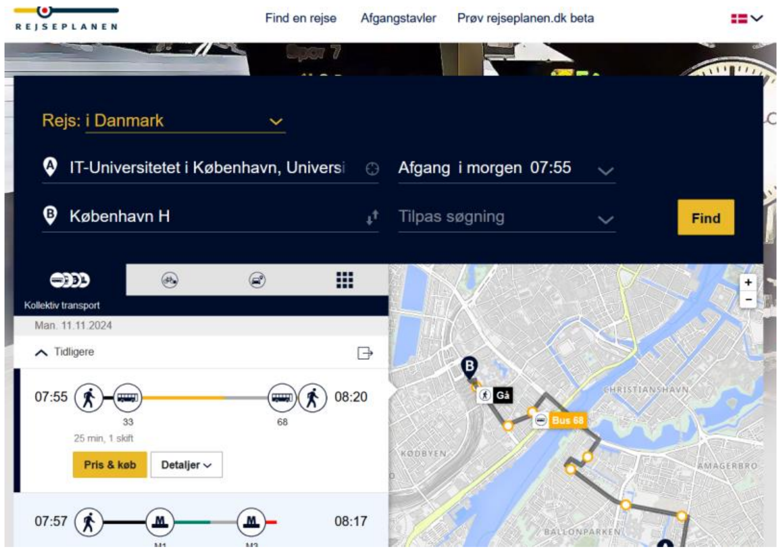
25 min, 1 skift

Pris & køb | Detaljer

07:57 — M1 — M3 — 08:17

Your personal "Rejseplan"

or


REJSEPLANEN

Find en rejse | Afgangstavler | Prøv rejseplanen.dk beta

Rejs: i Danmark

IT-Universitetet i København, Universi

Afgang i morgen 07:55

København H

Tilpas søgning

Find

Kollektiv transport

Man. 11.11.2024

Tidligere

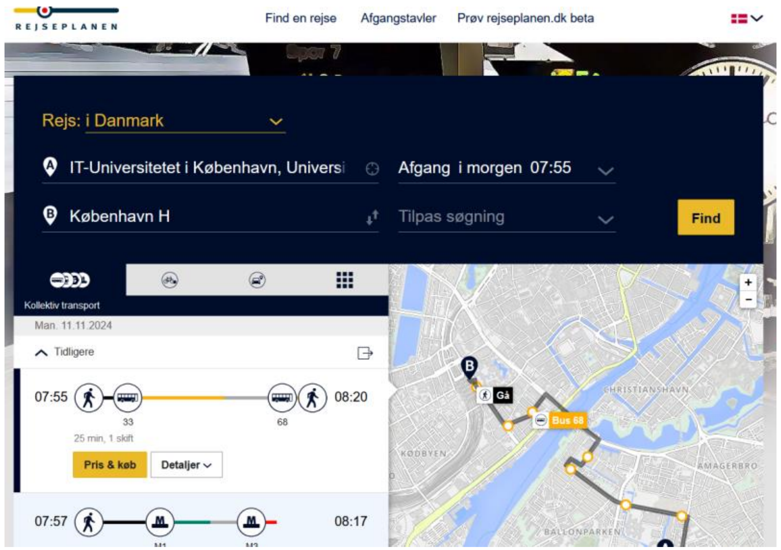
07:55 — 33 — 68 — 08:20

25 min, 1 skift

Pris & køb | Detaljer

07:57 — M1 — M3 — 08:17

Your personal "Rejseplan"



or

Simple Java program

```
Bus 33: 11:59 mod Rådhuspladsen St. (H.C. Andersens Boulevard)
Bus 33: 12:02 mod Nøragersmindevej (Kongelundsvej)
Bus 33: 12:14 mod Rådhuspladsen St. (H.C. Andersens Boulevard)
Bus 33: 12:17 mod Dragør Stationsplads
```

Finding your bus stop

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=xxx

Finding your bus stop

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=xxx

Replace xxx with a string e.g.

Lyngby

Vesterport

Personalized rejseplan

Personalized rejseplan

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
    } catch (IOException e) {  System.out.println(e.getMessage());   }
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
    } catch (IOException e) {  System.out.println(e.getMessage());    }
    System.out.println(new String(res, StandardCharsets.UTF_8));
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
    } catch (IOException e) {  System.out.println(e.getMessage());    }
    System.out.println(new String(res, StandardCharsets.UTF_8));
  }
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
    } catch (IOException e) {  System.out.println(e.getMessage());    }
    System.out.println(new String(res, StandardCharsets.UTF_8));
  }
  public static void main(String[] args) {  new BusDepart();  }
```

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
    } catch (IOException e) {  System.out.println(e.getMessage());   }
    System.out.println(new String(res, StandardCharsets.UTF_8));
  }
  public static void main(String[] args) {  new BusDepart();  }
}
```

code-exercises/…/NetworkFetcher

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

Personalized rejseplan

Rejseplanen has an open API, see file ReST_documentation_Rejseplanen_Latest.pdf

```
public class BusDepart {

  final static String RejseplanURL =
    "https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=";
  final static String ITU = "000000900";

  NetworkFetcher nf= new NetworkFetcher();

  public BusDepart(){
    byte[] res= null;
    try { res= nf.getUrlBytes(RejseplanURL+ITU);
    } catch (IOException e) {  System.out.println(e.getMessage());   }
    System.out.println(new String(res, StandardCharsets.UTF_8));
  }
  public static void main(String[] args) {  new BusDepart();  }
}
```

code-exercises/…/NetworkFetcher

https://xmlopen.rejseplanen.dk/bin/rest.exe/departureBoard?offsetTime=0&format=json&id=000000900

JSON version of "rejseplanen"

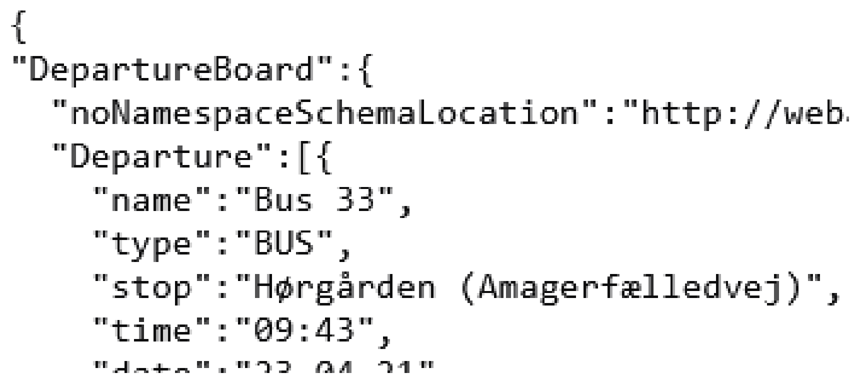
See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

JSON version of "rejseplanen"

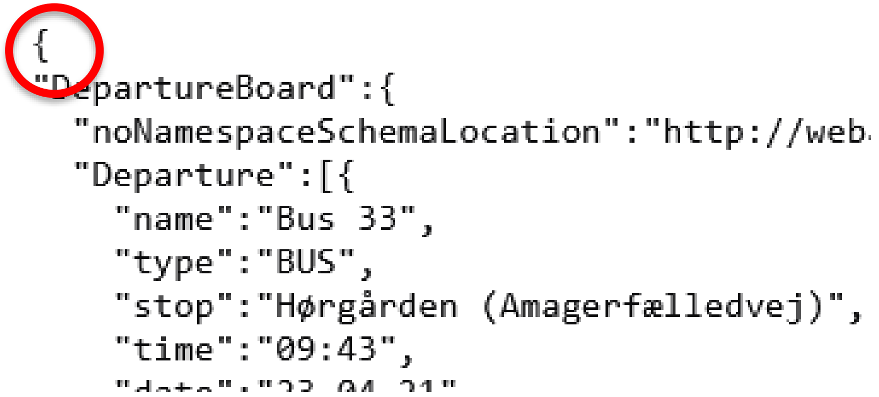
See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf



```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
    "date":"23.04.21"
```

JSON version of "rejseplanen"

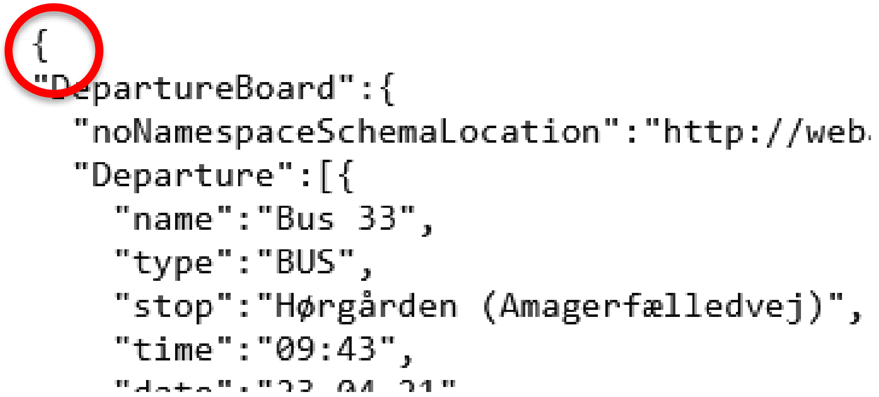
See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf



```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
    "date":"23.04.21"
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

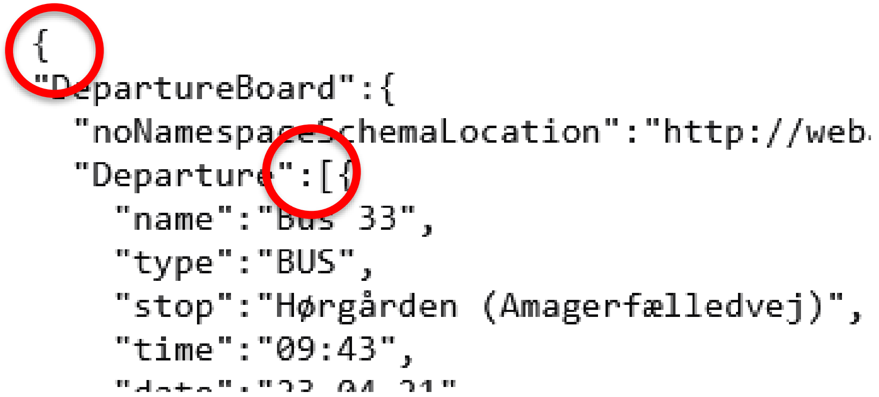


```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
    "date":"23.04.21"
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

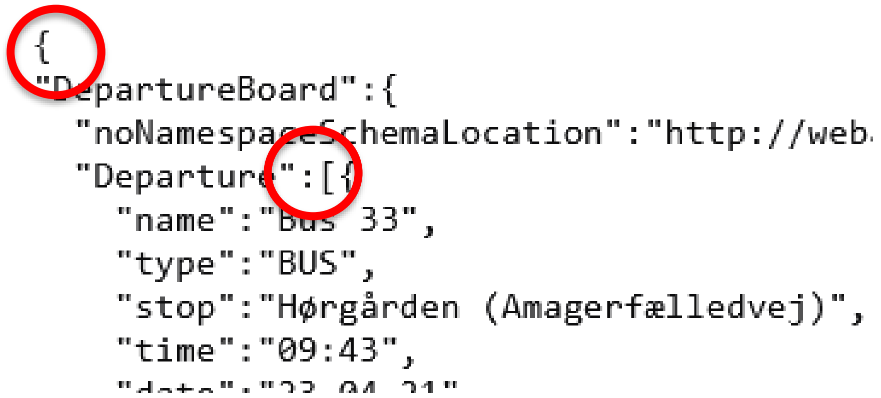


```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
    "date":"23.04.21"
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

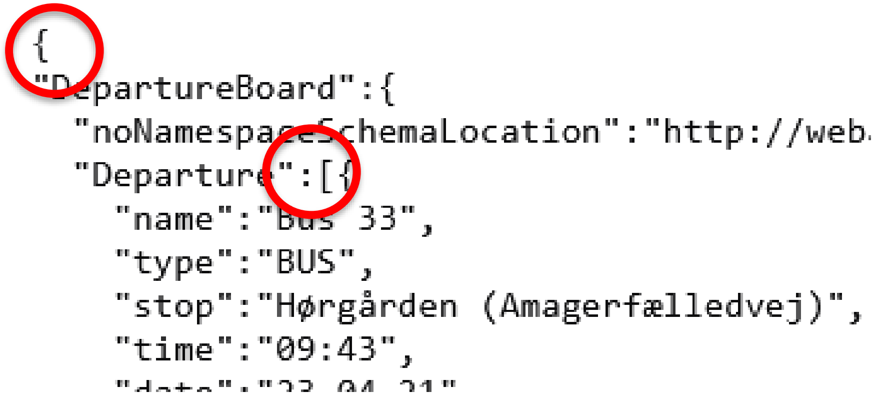


```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
JSONArray depArray= depBoard.getJSONArray("Departure");
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

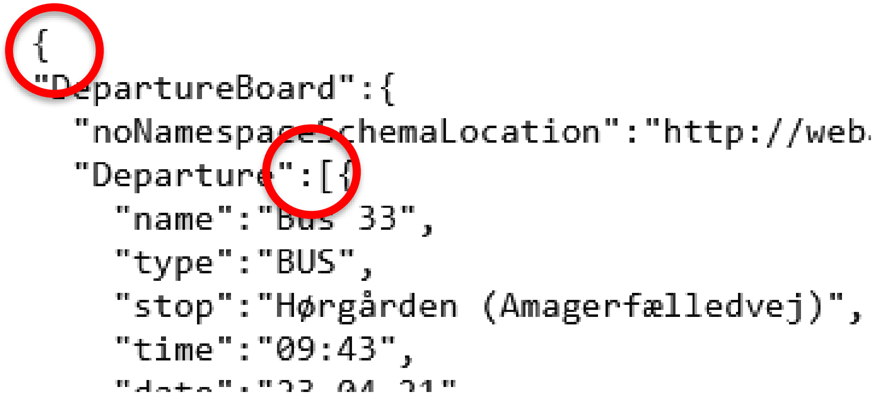


```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
    "date":"[illegible]"
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
JSONArray depArray= depBoard.getJSONArray("Departure");
if (depArray.length()>0) {
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

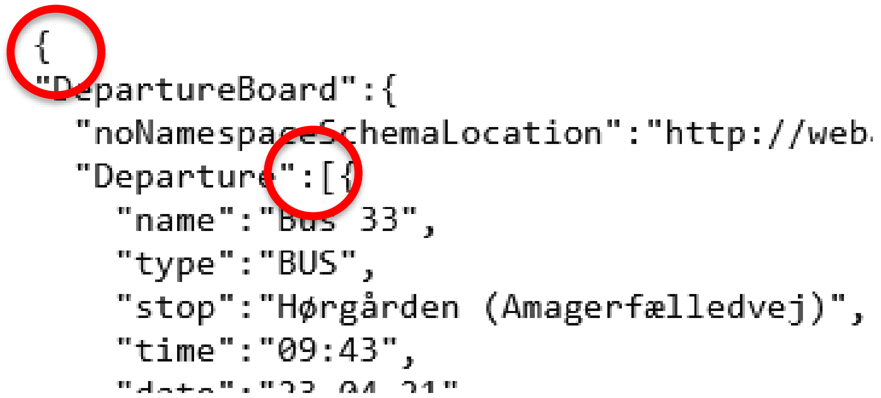


```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
    "date":"22.04.21"
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
JSONArray depArray= depBoard.getJSONArray("Departure");
if (depArray.length()>0) {
for (int i=0; ((i<depArray.length() && (found<4))); i++) {
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf

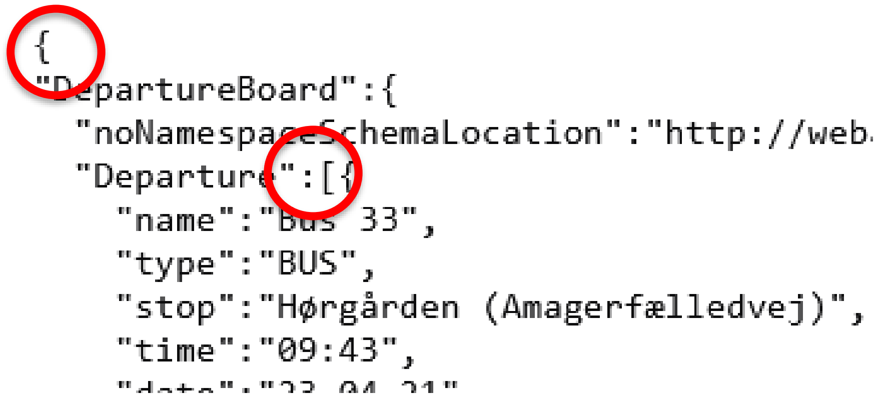


```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
JSONArray depArray= depBoard.getJSONArray("Departure");
if (depArray.length()>0) {
for (int i=0; ((i<depArray.length() && (found<4))); i++) {

   String bName= depArray.getJSONObject(i).getString("name");
   ...
```

JSON version of "rejseplanen"

See GitHub week11: ReST_documentation_Rejseplanen_Latest.pdf



```
{
"DepartureBoard":{
  "noNamespaceSchemaLocation":"http://web
  "Departure":[{
    "name":"Bus 33",
    "type":"BUS",
    "stop":"Hørgården (Amagerfælledvej)",
    "time":"09:43",
```

```
JSONObject depBoard= jsonBody.getJSONObject("DepartureBoard");
JSONArray depArray= depBoard.getJSONArray("Departure");
if (depArray.length()>0) {
for (int i=0; ((i<depArray.length() && (found<4))); i++) {

   String bName= depArray.getJSONObject(i).getString("name");
   ...

}
```

JSON:
lightweight data interchange format

Java**S**cript **O**bject **N**otation

JavaScript object

```
var item= {
  what: "can",
  whereC: "metal"
};
```

JSON (String):

```
{"what":"can", "whereC":"metal"}
```

JSON

JSON:

lightweight data interchange format

Java**S**cript **O**bject **N**otation

JavaScript object

```
var item= {
  what: "can",
  whereC: "metal"
};
```

JSON (String):

```
{"what":"can", "whereC":"metal"}
```

JSON String is a serialized version of a JavaScript object

JSON

JSON:

lightweight data interchange format

Java**S**cript **O**bject **N**otation

JavaScript object

```
var item= {
  what: "can",
  whereC: "metal"
};
```

JSON (String):

```
{"what":"can", "whereC":"metal"}
```

JSON String is a serialized version of a JavaScript object

https://www.w3schools.com/js/js_json.asp

Fetching elements from a JSONSTRING

Object

```
o: {"what":"can", "whereC":"metal"}
```

Fetching elements from a JSONSTRING

Object

```
o: {"what":"can", "whereC":"metal"}
```

```
o.getString("what")
```

Fetching elements from a JSONSTRING

Object

```
o: {"what":"can", "whereC":"metal"}
```

```
o.getString("what")
o.getString("whereC")
```

Fetching elements from a JSONSTRING

Object

```
o: {"what":"can", "whereC":"metal"}
```

```
o.getString("what")
o.getString("whereC")
```

Array

```
a: [   ...   { }    ...            ]
```

Fetching elements from a JSONSTRING

Object

```
o: {"what":"can", "whereC":"metal"}
```

```
o.getString("what")
o.getString("whereC")
```

Array

```
         0             i
a: [   ...   { }    ...              ]
```

Fetching elements from a JSONSTRING

Object

```
o: {"what":"can", "whereC":"metal"}
```

```
o.getString("what")
o.getString("whereC")
```

Array

```
         0        i
a: [   ...    { }    ...           ]
```

```
a.getJSONObject(i)
```

JSON library

```
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

JSON library

```
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

build.gradle

```
...
dependencies {
    // Use JUnit test framework.
    testImplementation 'junit:junit:4.13.2'

    // This dependency is used by the application.
    implementation 'com.google.guava:guava:30.1.1-jre'

    implementation 'org.json:json:20240303'
...
}
```

JSON library

```
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

Tutorial: https://www.w3schools.com/js/js_json.asp

```
build.gradle
...
dependencies {
    // Use JUnit test framework.
    testImplementation 'junit:junit:4.13.2'

    // This dependency is used by the application.
    implementation 'com.google.guava:guava:30.1.1-jre'

    implementation 'org.json:json:20240303'
...
}
```

Rejseplanen info in Java

BusDepart.java

and

NetworkFetcher.java

Both in exercises directory

```
Bus 33: 11:59 mod Rådhuspladsen St. (H.C. Andersens Boulevard)
Bus 33: 12:02 mod Nøragersmindevej (Kongelundsvej)
Bus 33: 12:14 mod Rådhuspladsen St. (H.C. Andersens Boulevard)
Bus 33: 12:17 mod Dragør Stationsplads
```

Erlang Introduction

Agenda

- The shell
- Datatypes
- Conditional statements
- Pattern matching
- Errors and exceptions
- No loops (recursion)
- Some useful data structures
- A larger example

About Erlang

- Developed by Joe Armstrong, Robert Virding, and Mike Williams in 1986
  - Open-sourced in 1998; despites Ericsson's attempts to prevent it
- Erlang = **Er**icsson **Lang**uage
  - (Presumably) named after the Danish mathematician Agner Krarup Erlang (1878 –1929) for his pioneering and influential work in the field of telecommunications
- Language developed for telephony applications
  - Erlang/OTP is supported and maintained by the Open Telecom Platform (OTP) product unit at Ericsson.
- Famously used at WhatsApp (among many other companies)
  - In 2014, there were only 32 engineers at WhatsApp developing/maintaining software for 450 million users
- Multiple companies use Erlang in production
  - https://erlang-companies.org/

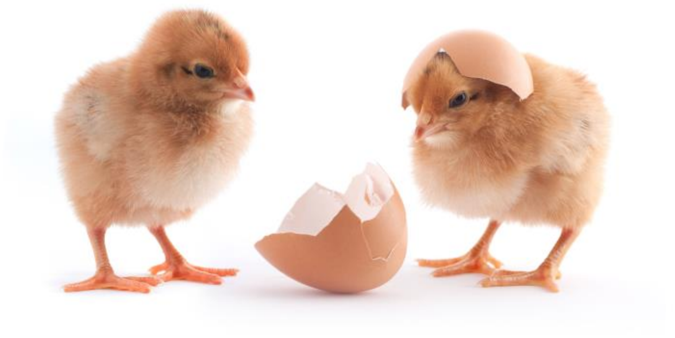
...

Erlang references for the course

- Online textbook that we will follow
    - Learn you Some Erlang for Great Good
    - https://learnyousomeerlang.com/content
- Official documentation (for OTP 26)
    - https://www.erlang.org/docs/26/

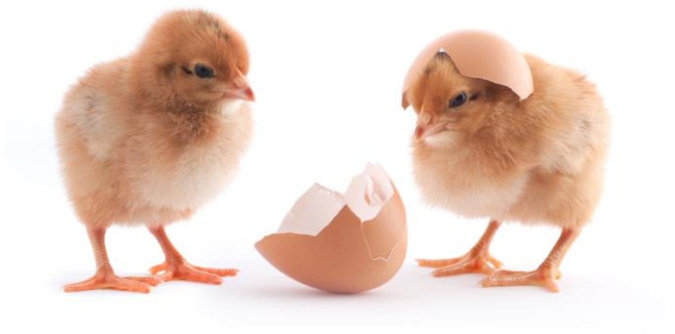
Erlang shell - `erl`

- You can start the shell by issuing the command `erl` in your terminal
  - This is also a sanity check that your implementation is correct
  - Recall that all commands finish with a ”.”
- To exit the shell, issue the command `>q().`
- For compiling, we will use `erl -make` in the folder with all the Erlang code



Erlang shell - **`erl`**

- You can use the **`h()`** function to print the shell help page
- Also, the functions **`h(Module)`** or **`h(Module,Function)`** to access the documentation (manual pages) for modules and functions



Immutable bind

- In Erlang, you can bind variables at most once

```
1> X = 1.
1

2> X = 2.
** exception error: no match of right hand side value 2

3> X = 1.
1
```

Immutable bind

- Variable binds and **`erl`**
  - **`b()`** to view current variable binds
  - **`f()`** or **`f(Var)`** to remove all existing variable binds or a specific variable bind, respectively

```
1> X=1, Y=2, Z=3.
3

2> b().
X = 1
Y = 2
Z = 3
ok
```

```
3> f(X).
ok

4> b().
Y = 2
Z = 3
ok

5> f().
ok

6> b().
ok
```

Common types

- Erlang is dynamically typed
  - You do not need to write types explicitly, but it will crash if you issue function calls or expressions with wrong types

- Common data types
  - Atoms
  - Integers, Booleans, etc.
  - Tuples, Lists, etc.
  - Records

Common types

- Atoms
  - Character array starting with a lower-case character
  - Constant literal representing its own value
  - Useful IDs

```
1> atom.
atom

3> atom == 'atom'.
true

4> atom == "atom".
false
```

Common types

- Numbers: Integer, floats, … , and operations

```
1> 2.
2


2> 2 + 3.0.
5.0

3> 2 * 3.
6

4> 3 / 2.
1.5

5> 3 div 2.
1

6> 3 rem 2.
1
```

- Boolean values and operations

```
1> 2 == 2.0.
true
2> 2 =:= 2.0.
false
3> 2 /= 2.0.
false
4> 2 =/= 2.0.
true
5> 2 >= 2.0.
true
6> 2 >= 2.
true
7> 2 > 2.
false
8> 2 > 2.0.
false
```

Common types

- Tuples
  - Surrounded by curly brackets "**{ }**"
  - Do not confuse them with sets
  - You can mix types

```
1> {1,2}.
{1,2}

2> {1,1,2}.
{1,1,2}

3> {1, 1.0, one, "one"}.
{1,1.0,one,"one"}
```

Common types

- Lists
  - Surrounded by square brackets “[ ]”
  - Unbounded (but countable) size
  - You can mix types

```
1> [1,2].
[1,2]

2> [1,1,2].
[1,1,2]

3> [1, 1.0, one, "one"].
{1,1.0,one,"one"}
```

Common types

- Lists comprehensions
  - Erlang supports list comprehensions
  - They are specified with the operator **||**
  - Generators are specified with **<-**
  - You can concatenate conditions and generators using “**,**”

```
1> [X || X <- [1,2,3]].
[1,2,3]

2> [X || X <- [1,2,3,4]].
[1,2,3,4]

3> [X || X <- [1,2,3,4], X > 2].
[3,4]

4> [{X,Y} || X <- [1,2,3,4], Y <- [1,2]].
[{1,1},{1,2},{2,1},{2,2},{3,1},{3,2},{4,1},{4,2}]

5> [{X,Y} || X <- [1,2,3,4], Y <- [1,2], X+Y > 4].
[{3,2},{4,1},{4,2}]
```

Common types

By Pedro Aragão - CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=28967487

- Records
  - Structured tuples
  - An ad-hoc way to define data-structure-like objects
  - Define with **`-record(<atom>, {<attr1>, <attr2>, …, <attrn>}).`**
  - Attributes may have default values
  - Access via attribute names instead of indexes (as in tuples)

```
1> -record(vehicle, {type=car, brand, model, color}).
ok

2> #vehicle{brand=lamborghini, model=huracan, color=blue}.
#vehicle{type = car,brand = lamborghini,model = huracan,
         color = blue}

3> Plane1 = #vehicle{type=plane, brand=airbus, model=a320,
color=white}.
#vehicle{type = plane,brand = airbus,model = a320,
         color = white}

4> Plane1#vehicle.model.
a320

5> Plane1#vehicle{color=blue}.
#vehicle{type = plane,brand = airbus,model = a320,
         color = blue}
```

Functions!

- Regular functions
  - A function can be composed from one or more expressions
  - Expressions are separated by ” , ”
  - The last expression is marked with ” . ” and value resulting from is evaluation is the return value of the function

```
square_plus_2(X) ->
    Y=X*X,
    2+Y.
```

- Higher order functions
  - Functions can take functions as parameters or return functions
  - Anonymous functions (lambdas) are defined using `fun (P1, P2, …) -> … end.`

```
apply_after_squared(X, F) ->
    F(X*X).

%% Later in the erl shell
apply_after_squared(2, fun (X) -> X+2 end)
```

Printing

- You can print in standard output with the functions **`format`** or **`fwrite`** from the module **`io`**
  - These functions take a String and list of Erlang terms to print
  - The control sequence **`~p`** can be used to print Erlang terms, and **`~n`** for line breaks.
  - There are other control sequences, see https://www.erlang.org/docs/26/man/io#format-3

```
1> I = 42, F = 42.0, A = forty_two, S = "forty two".
"forty two"

2> io:format("Here is are examples of an integer: ~p, a float ~p, an atom: ~p, and a String: ~p ~n",[I, F, A, S]).
Here is are examples of an integer: 42, a float 42.0, an atom: forty_two, and a String: "forty two"
ok
```

Conditional statements

- If-statements
  - At least one branch of the if-statement must evaluate to true for any input to the function
  - Setting the condition to true is the same as having else
  - Different if-branches are separated by ”;”, and note that the last case does not end with any punctuation mark
  - Conditions are evaluated from top to bottom

```
if Vehicle#vehicle.type == car ->
        io:format("It is a car~n");
   Vehicle#vehicle.type == plane ->
        io:format("It is a plane~n");
   Vehicle#vehicle.type == plane, Vehicle#vehicle.color == white ->
        io:format("It is a white plane~n");
   true -> % this is the else
        io:format("Vehicle type unknown~n")
end.
```

Conditional statements

- If-statements
  - At least one branch of the if-statement must evaluate to true for any input to the function
  - Setting the condition to true is the same as having else
  - Different if-branches are separated by ”;”, and note that the last case does not end with any punctuation mark
  - Conditions are evaluated from top to bottom

```
if Vehicle#vehicle.type == car ->
        io:format("It is a car~n");
   Vehicle#vehicle.type == plane ->
        io:format("It is a plane~n");
   Vehicle#vehicle.type == plane, Vehicle#vehicle.color == white ->
        io:format("It is a white plane~n");
   true -> % this is the else
        io:format("Vehicle type unknown~n")
end.
```

Conditional statements

There are also case-statements; we will see them in a few slides

- If-statements
  - At least one branch of the if-statement must evaluate to true for any input to the function
  - Setting the condition to true is the same as having else
  - Different if-branches are separated by ";", and note that the last case does not end with any punctuation mark
  - Conditions are evaluated from top to bottom

```
if Vehicle#vehicle.type == car ->
        io:format("It is a car~n");
   Vehicle#vehicle.type == plane ->
        io:format("It is a plane~n");
   Vehicle#vehicle.type == plane, Vehicle#vehicle.color == white ->
        io:format("It is a white plane~n");
   true -> % this is the else
        io:format("Vehicle type unknown~n")
end.
```

Conditional statements

- Function guards
  - This function has the same behavior as the if-statement in the previous slide
  - The guard is specified after the function definition with the keyword when followed by a condition
  - Function cases are separated by “;” and the last one is marked by ”.”
  - Function cases are evaluated from top to bottom

```
what_type_is_vehicle_guard(Vehicle) when Vehicle#vehicle.type == car ->
    io:format("It is a car~n");

what_type_is_vehicle_guard(Vehicle) when Vehicle#vehicle.type == plane ->
    io:format("It is a plane~n");

what_type_is_vehicle_guard(Vehicle) when Vehicle#vehicle.type == plane, Vehicle#vehicle.color == white ->
    io:format("It is a white plane~n");

what_type_is_vehicle_guard(_Vehicle) ->
    io:format("Vehicle type unknown~n").
```

Pattern matching

- Erlang provides many pattern matching advanced features
  - Assignments
  - Conditional statements
  - Function definitions
  - ...

Pattern matching

- Erlang provides many pattern matching advanced features
  - Assignments
  - Conditional statements
  - Function definitions
  - …

```
1> #vehicle{brand=RX} = #vehicle{brand="Volvo"}.
#vehicle{type = car,brand = "Volvo",model = undefined,
         color = undefined}
2> RX.
"Volvo"
```

```
1> {X,Y} = {1,2}.
{1,2}

2> X.
1

3> Y.
2
```

```
1> [H|T] = [1,2,3].
[1,2,3]

2> H.
1

3> T.
[2,3]
```

```
1> [E1,E2|T1] = [1,2,3,4].
[1,2,3,4]

2> E1.
1

3> E2.
2

4> T1.
[3,4]
```

Pattern matching

- Erlang provides many pattern matching advanced features
  - Assignments
  - Conditional statements
  - Function definitions
  - …

```
case X of
    {Y,_Z} ->
        io:format("The first element in the 2-tuple is ~p~n", [Y]);
    [Y|_Z] ->
        io:format("The head of the list is ~p~n", [Y]);
    #vehicle{color=Y} ->
        io:format("The color of the ~p is ~p~n", [X#vehicle.type, Y]);
    X ->
        io:format("The value of the input is ~p and we do not know the type ~n", [X])
end.
```

Pattern matching

- Erlang provides many pattern matching advanced features
  - Assignments
  - Conditional statements
  - Function definitions
  - …

You can prefix a variable name with _ to specify that it is unused, e.g., `_Z` in the code below

```
case X of
    {Y,_Z} ->
        io:format("The first element in the 2-tuple is ~p~n", [Y]);
    [Y|_Z] ->
        io:format("The head of the list is ~p~n", [Y]);
    #vehicle{color=Y} ->
        io:format("The color of the ~p is ~p~n", [X#vehicle.type, Y]);
    X ->
        io:format("The value of the input is ~p and we do not know the type ~n", [X])
end.
```

Pattern matching

- Erlang provides many pattern matching advanced features
  - Assignments
  - Conditional statements
  - Function definitions
  - …

```
equal(X,X) -> true;
equal(_,_) -> false.

len([]) -> 0;
len([_H|T]) -> 1 + len(T);
len(_Other) ->
    io:format("Please input a list~n"),
    exit(badarg).
```

Pattern matching

- Erlang provides many pattern matching advanced features
  - Assignments
  - Conditional statements
  - Function definitions
  - …

Also, you can use _ in pattern matching to match on any value and not bind it to any variable, .e.g., `equal(_,_)` below

```
equal(X,X) -> true;
equal(_,_) -> false.

len([]) -> 0;
len([_H|T]) -> 1 + len(T);
len(_Other) ->
    io:format("Please input a list~n"),
    exit(badarg).
```

Exception handling

- Erlang processes can exit:
  - Normally (this is equivalent to **`exit(normal)`**)
  - Abnormally (different type of errors/exceptions)
- Try-catch-after blocks may be used to handle different types of exceptions

```
try F() of %% "of" is optional and you can have several statements in sequence separeted by ","
    _ ->
        ok
catch
    exit:Exit ->
        io:format("The function has thrown an exit error: ~p~n",[Exit]),
        {exit, Exit};
    error:specific_error ->
        io:format("The function has thrown a specfic error~n"),
        {error, specific_error};
    error:Error ->
        io:format("The function has thrown an error error: ~p~n",[Error]),
        {error, Error};
    throw:Throw ->
        io:format("The function has thrown an throw error: ~p~n",[Throw]),
        {error, Throw};
    _:AnyOtherException -> % sink case (not recommended)
        io:format("The function has thrown an error of any other type: ~p~n",[AnyOtherException]),
        {any_other_exception, AnyOtherException}
 after
    io:format("This is similar (but not equivalent) to a finally block in Java's try-catch-finally~n")
 end.
```

Exception handling

- Erlang processes can exit:
  - Normally (this is equivalent to `exit(normal)`)
  - Abnormally (different type of errors/exceptions)
- It is also possible to use the catch function
- This function returns a tuple with the stack trace

```
case catch(F()) of
    {'EXIT', Reason} ->
        Reason;
    _ -> ok
end.
```

```
1> erlang_intro:catcher(fun () -> lists:member(23) end).
{undef,[{lists,member,[23],[]},
        {erlang_intro,catcher,1,
                      [{file,"erlang_intro.erl"},{line,80}]},
        {erl_eval,do_apply,7,[{file,"erl_eval.erl"},{line,750}]},
        {shell,exprs,7,[{file,"shell.erl"},{line,783}]},
        {shell,eval_exprs,7,[{file,"shell.erl"},{line,739}]},
        {shell,eval_loop,4,[{file,"shell.erl"},{line,724}]}]}
```

No loops → Recursion

- As you probably know, functional programming languages do not include loops
- Loop like behavior is modelled by iterating data structures like lists or with recursion
- There are two main types of recursion: standard or tail-recursive
- Example → Factorial of $n \in \mathbb{N}$: $n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdot \ldots \cdot n$

```
% standard

factorial(0) ->
    1;

factorial(N) ->
    N*factorial(N-1).
```

```
% tail recursive

factorial_tail(N) ->
    factorial_tail(N,1).

factorial_tail(0, Acc) ->
    Acc;
factorial_tail(N, Acc) ->
    factorial_tail(N-1,Acc*N).
```

Modules

- Erlang programs consists of modules; one or more
- A module is defined by the statement **`-module`**
- A module consists a collection of function and/or record definitions
- Functions to be used in external modules must be declared using the **`-export`** statement
- You can compile and load a module in the shell with **`>c(module)`**

```
%% module definition
-module(erlang_intro).

%% exporting functions
-export([square_plus_2/1, apply_after_squared/2, what_type_is_vehicle_if/1,
         what_type_is_vehicle_guard/1, equal/2, len/1, try_catcher/1,
         Catcher/1, factorial/1, factorial_tail/1]).
```

Header files

- It is useful to define records separately and then import them in the module
- You can load all records defined/included in a module with `>rr(module)`

```
%% erlang_intro.erl

%% module definition
-module(erlang_intro).


…

%% import records
-include("header.hrl").
```

```
%% header.hrl

%% Header file with record definitions
-record(vehicle, {type=car, brand, model, color}).
```

Useful data structures

- Lists
  - Standard module: **`lists`**

```
1> L = lists:seq(1,10).
[1,2,3,4,5,6,7,8,9,10]

2> lists:member(4,L).
True

3> lists:map(fun (X) ->  erlang_intro:factorial(X) end,L).
[1,2,6,24,120,720,5040,40320,362880,3628800]

4> lists:foreach(fun (X) ->  erlang_intro:factorial(X) end,L).
ok

5> lists:foldl(fun (X,Y) ->  erlang_intro:factorial(X)-Y end,0,L).
3301819

6> lists:foldr(fun (X,Y) ->  erlang_intro:factorial(X)-Y end,0,L).
-3301819
```

Useful data structures

- Key-value store
  - Standard modules: **maps** and dictionaries (**orddict**, **dict**)

```
1> M = #{a => 1, b => 2}.
#{a => 1,b => 2}

2> maps:from_list([{a,1},{b,2}]).
#{a => 1,b => 2}

3> maps:get(b,M).
2

4> maps:get(c,M).
** exception error: bad key: c
     in function  maps:get/2
        called as maps:get(c,#{a => 1,b => 2})
        *** argument 1: not present in map

5> maps:find(c,M).
error
```

Useful data structures

- Key-value store
  - Standard modules: **`maps`** and dictionaries (**`orddict`**, **`dict`**)

```
6> maps:put(c,3,M).
#{c => 3,a => 1,b => 2}

7> maps:remove(a,M).
#{b => 2}

8>maps:map(fun (K,V) -> erlang_intro:square_plus_2(V) end, M).
#{a => 3,b => 6}

9> maps:fold(fun (K,V,Acc) -> erlang_intro:square_plus_2(V) + Acc end, 0, M).
9
```

Useful data structures

Documentation about the internal implementation of sets: *”Any code assuming knowledge of the format is running on thin ice.”*

- Sets
  - Standard modules: **`sets`**

```
1> S = sets:from_list([1,1,2,2,3,3,3,3,4]).
{set,4,16,16,8,80,48,
     {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
     {{[],[3],[],[],[],[],[2],[],[],[],[],[1],[4],[],[],[]}}}

2> T = sets:from_list([2,2,4]).
{set,2,16,16,8,80,48,
     {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
     {{[],[],[],[],[],[],[2],[],[],[],[],[],[4],[],[],[]}}}

3> sets:is_subset(S,T).
false

4> sets:is_subset(T,S).
true

5> sets:intersection(S,T).
{set,2,16,16,8,80,48,
     {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
     {{[],[],[],[],[],[],[2],[],[],[],[],[],[4],[],[],[]}}}
```

Module and function documentation

- In **`erl`** you can use
  - **`h(Module)`** to access the documentation for modules
  - **`h(Module,Function)`** to access the documentation for a function within the module
- Alternatively, use the HTML version of the documentation for modules
  - https://www.erlang.org/docs/26/man_index

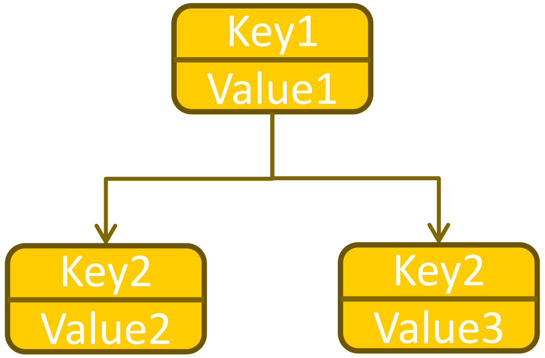
Long(er) example – Time Formatter

- Example function implementation to convert seconds into the format HH:MM:SS

```
-module(time_formatter).

-export([time/1]).

time(Seconds) ->
    Hours = Seconds div 3600,
    RemainingSecs = Seconds rem 3600,
    Minutes = RemainingSecs div 60,
    FinalSeconds = RemainingSecs rem 60,
    io:format('Time in format HH:MM:SS\n'),
    io:format('~p:~p:~p~n', [Hours, Minutes, FinalSeconds]).
```

Long(er) example – Tree data structure

- We revisit the tree data structure in "Learn You Some Erlang for Great Good"
- In this tree, nodes are represented as
  - key-value pairs with (possibly) two children
  - Or empty nodes
- We use tuples to represent nodes:
  - `{node, Key, Value, Left, Right}`
  - `{node, nil}`

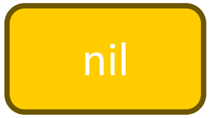


As expected, it must hold that
Key2 < Key1 and **Key1 < Key3**

Long(er) example – Tree data structure

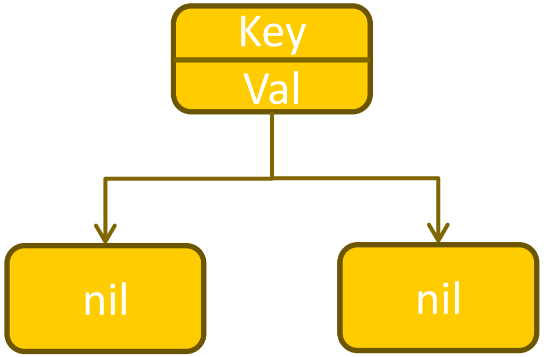
- Empty tree

```
empty() -> {node, nil}.
```



- Insert on an empty tree

```
insert(Key, Val, {node, nil}) ->
    {node, {Key, Val, {node, nil}, {node, nil}}};
```



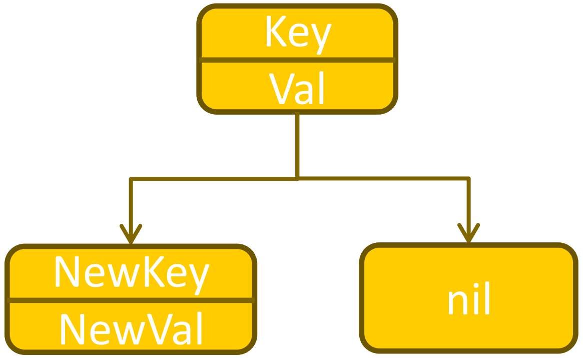
Long(er) example – Tree data structure

- Empty tree

```
empty() -> {node, nil}.
```

- Insert on an empty tree

```
insert(Key, Val, {node, nil}) ->
    {node, {Key, Val, {node, nil}, {node, nil}}};
```



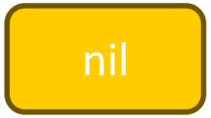
- Insert on non-empty tree

```
insert(NewKey, NewVal, {node, {Key, Val, Smaller, Larger}}) when NewKey < Key ->
    {node, {Key, Val, insert(NewKey, NewVal, Smaller), Larger}};
insert(NewKey, NewVal, {node, {Key, Val, Smaller, Larger}}) when NewKey > Key ->
    {node, {Key, Val, Smaller, insert(NewKey, NewVal, Larger)}};
insert(Key, Val, {node, {Key, _, Smaller, Larger}}) ->
    {node, {Key, Val, Smaller, Larger}}.
```

Long(er) example – Tree data structure

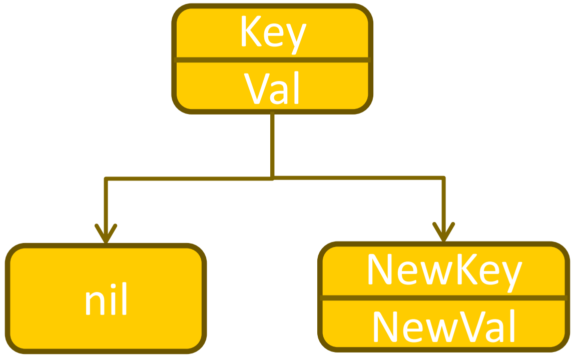
- Empty tree

```
empty() -> {node, nil}.
```



- Insert on an empty tree

```
insert(Key, Val, {node, nil}) ->
    {node, {Key, Val, {node, nil}, {node, nil}}};
```



- Insert on non-empty tree

```
insert(NewKey, NewVal, {node, {Key, Val, Smaller, Larger}}) when NewKey < Key ->
    {node, {Key, Val, insert(NewKey, NewVal, Smaller), Larger}};
insert(NewKey, NewVal, {node, {Key, Val, Smaller, Larger}}) when NewKey > Key ->
    {node, {Key, Val, Smaller, insert(NewKey, NewVal, Larger)}};
insert(Key, Val, {node, {Key, _, Smaller, Larger}}) ->
    {node, {Key, Val, Smaller, Larger}}.
```

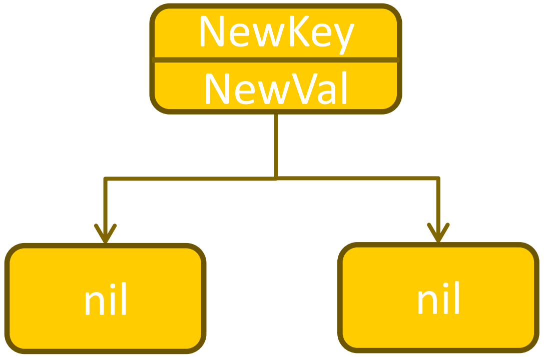
Long(er) example – Tree data structure

- Empty tree

```
empty() -> {node, nil}.
```

- Insert on an empty tree

```
insert(Key, Val, {node, nil}) ->
    {node, {Key, Val, {node, nil}, {node, nil}}};
```



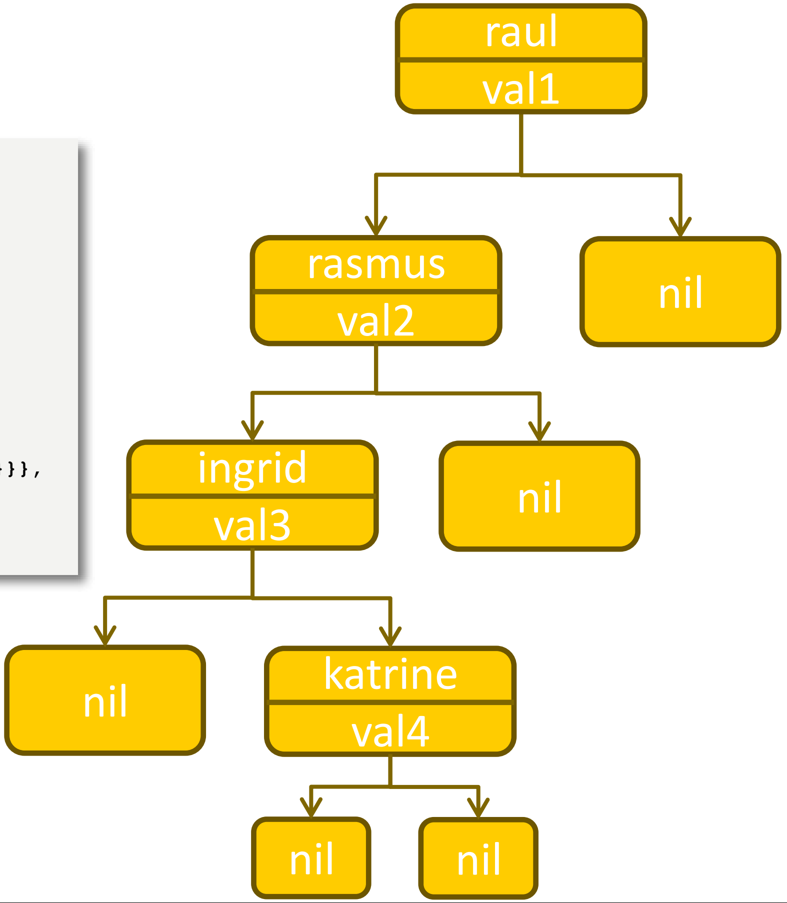
- Insert on non-empty tree

```
insert(NewKey, NewVal, {node, {Key, Val, Smaller, Larger}}) when NewKey < Key ->
    {node, {Key, Val, insert(NewKey, NewVal, Smaller), Larger}};
insert(NewKey, NewVal, {node, {Key, Val, Smaller, Larger}}) when NewKey > Key ->
    {node, {Key, Val, Smaller, insert(NewKey, NewVal, Larger)}};
insert(Key, Val, {node, {Key, _, Smaller, Larger}}) ->
    {node, {Key, Val, Smaller, Larger}}.
```

Long(er) example – Tree data structure

- Consider this tree

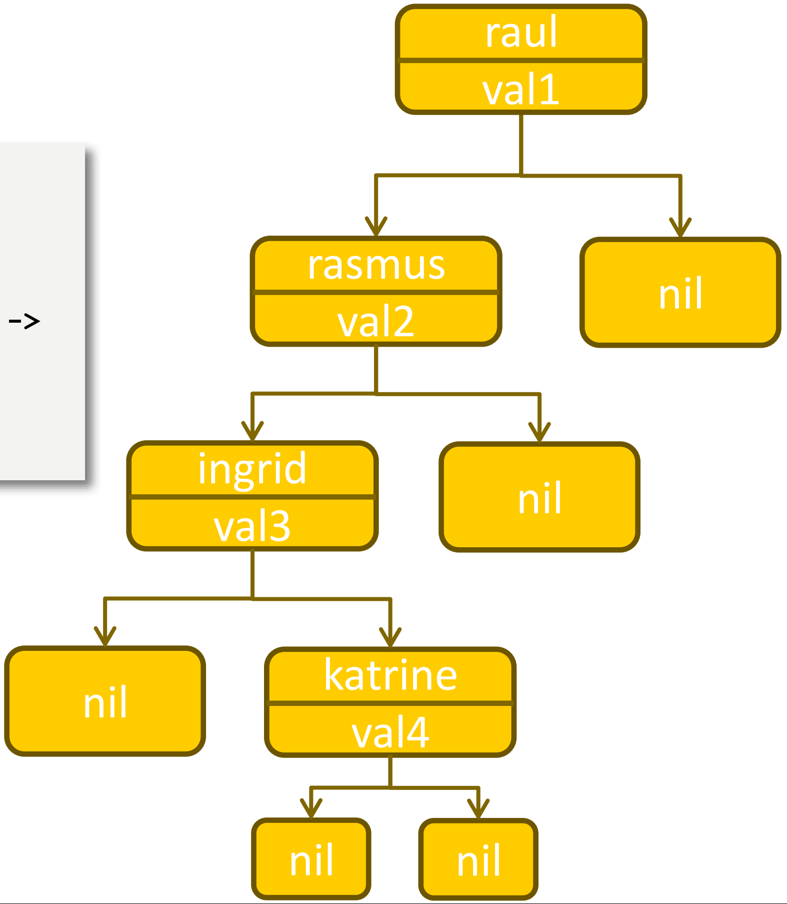
```
T1 = tree:insert(mathias, val1, tree:empty()),
T2 = tree:insert(rasmus, val2, T1),
T3 = tree:insert(ingrid, val3, T2),
T4 = tree:insert(katrine, val4, T3).
{node,{raul,val1,
            {node,{rasmus,val2,
                          {node,{ingrid,val2,
                                        {node,nil},
                                        {node,{katrine,val2,{node,nil},{node,nil}}}}},
                          {node,nil}}},
            {node,nil}}}
```



Long(er) example – Tree data structure

- Lookup on a tree

```
lookup(_, {node, nil}) ->
    undefined;
lookup(Key, {node, {Key, Val, _, _}}) ->
    {ok, Val};
lookup(Key, {node, {NodeKey, _, Smaller, _}}) when Key < NodeKey ->
    lookup(Key, Smaller);
lookup(Key, {node, {_, _, _, Larger}}) ->
    lookup(Key, Larger).
```

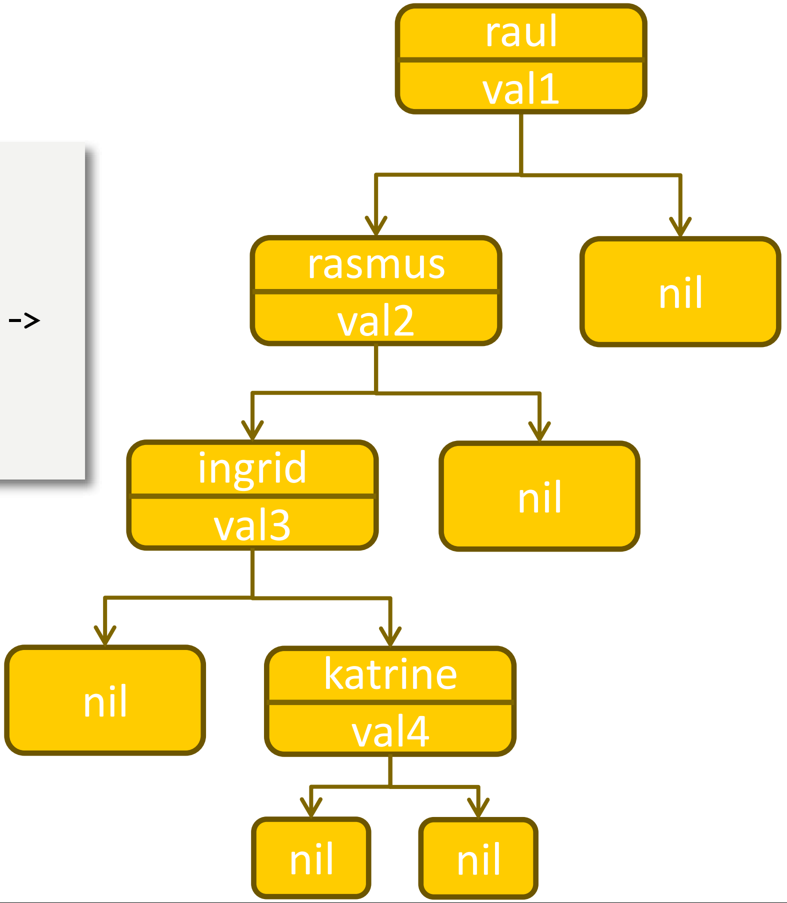


Long(er) example – Tree data structure

- Lookup on a tree

```
lookup(_, {node, nil}) ->
    undefined;
lookup(Key, {node, {Key, Val, _, _}}) ->
    {ok, Val};
lookup(Key, {node, {NodeKey, _, Smaller, _}}) when Key < NodeKey ->
    lookup(Key, Smaller);
lookup(Key, {node, {_, _, _, Larger}}) ->
    lookup(Key, Larger).
```

```
1> tree:lookup(ingrid,T4).
```

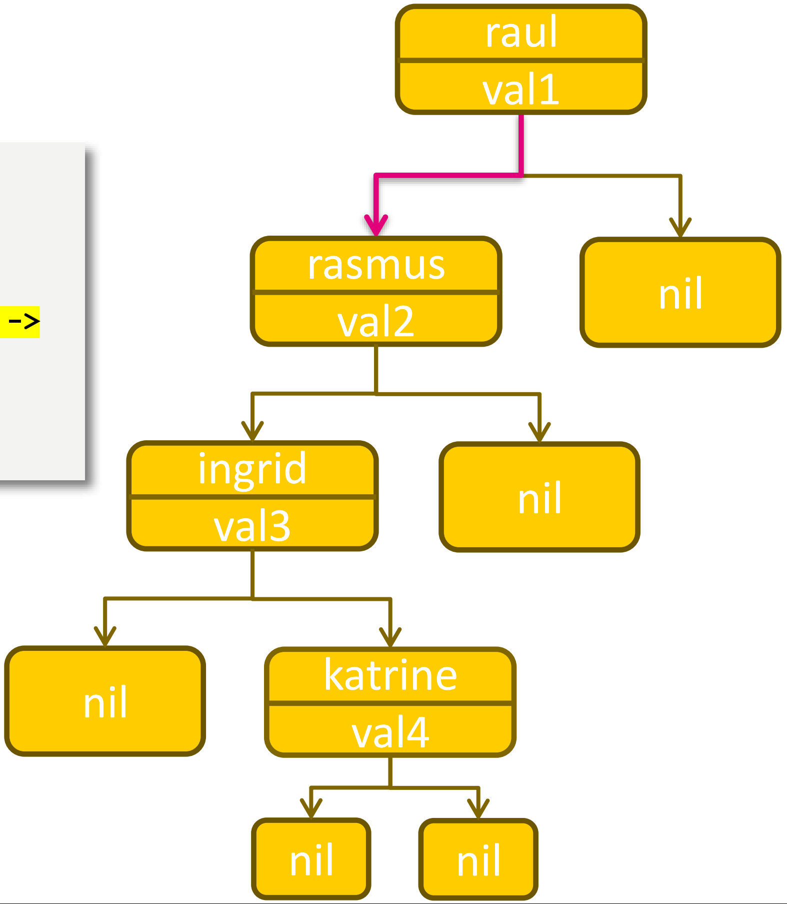


Long(er) example – Tree data structure

- Lookup on a tree

```
lookup(_, {node, nil}) ->
    undefined;
lookup(Key, {node, {Key, Val, _, _}}) ->
    {ok, Val};
lookup(Key, {node, {NodeKey, _, Smaller, _}}) when Key < NodeKey ->
    lookup(Key, Smaller);
lookup(Key, {node, {_, _, _, Larger}}) ->
    lookup(Key, Larger).
```

```
1> tree:lookup(ingrid,T4).
```

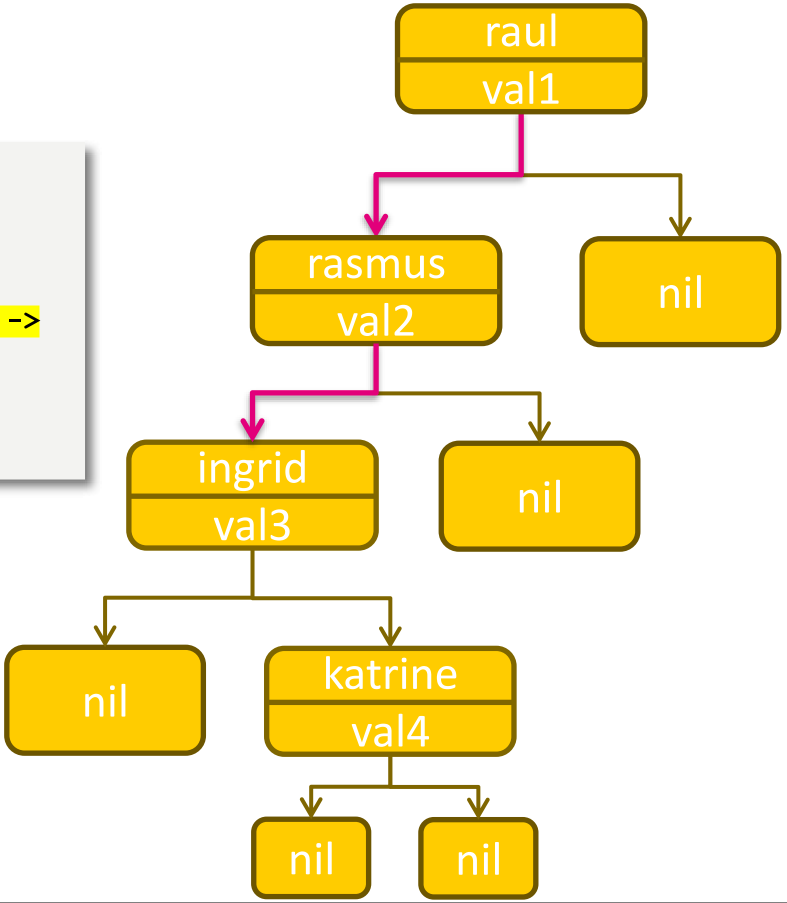


Long(er) example – Tree data structure

- Lookup on a tree

```
lookup(_, {node, nil}) ->
    undefined;
lookup(Key, {node, {Key, Val, _, _}}) ->
    {ok, Val};
lookup(Key, {node, {NodeKey, _, Smaller, _}}) when Key < NodeKey ->
    lookup(Key, Smaller);
lookup(Key, {node, {_, _, _, Larger}}) ->
    lookup(Key, Larger).
```

```
1> tree:lookup(ingrid,T4).
```

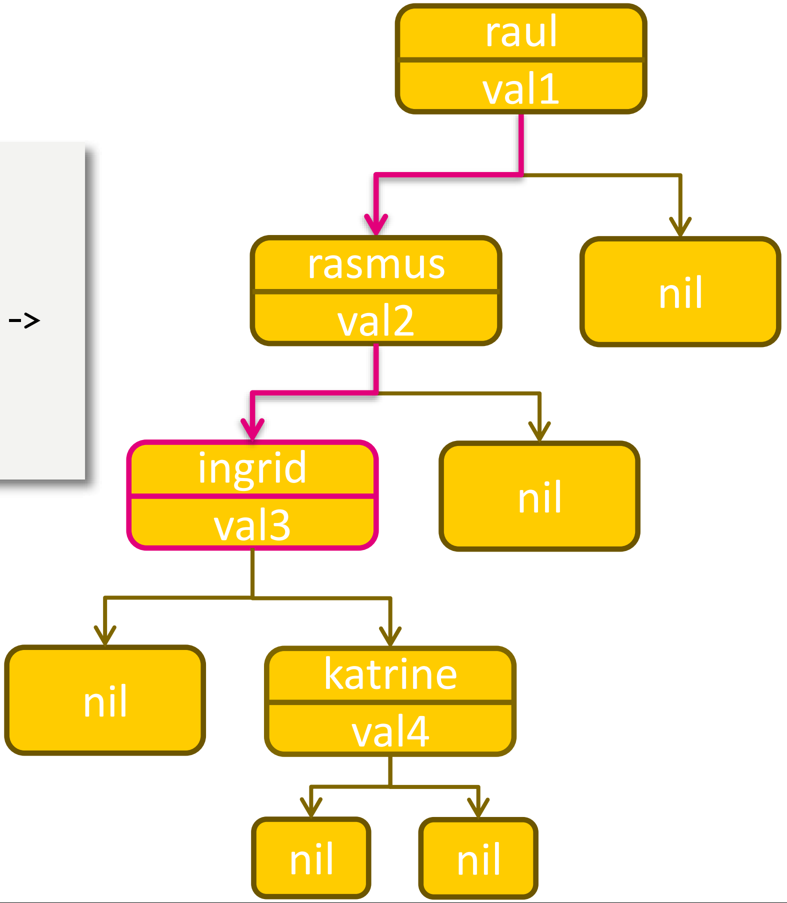


Long(er) example – Tree data structure

- Lookup on a tree

```
lookup(_, {node, nil}) ->
    undefined;
lookup(Key, {node, {Key, Val, _, _}}) ->
    {ok, Val};
lookup(Key, {node, {NodeKey, _, Smaller, _}}) when Key < NodeKey ->
    lookup(Key, Smaller);
lookup(Key, {node, {_, _, _, Larger}}) ->
    lookup(Key, Larger).
```

```
1> tree:lookup(ingrid,T4).
{ok,val3}
```


raul
val1
rasmus
val2
nil
ingrid
val3
nil
nil
katrine
val4
nil
nil