TECHNISCHE HOCHSCHULE LÜBECK

DISTRIBUTED SYSTEMS

12.04.2019

CLIENT-SERVER-COMMUNICATION

LENKA KLEINAU, M.SC.

SLIDES BASED ON MATERIAL FROM PROF. KROHN

TECHNISCHE
HOCHSCHULE
LÜBECK

- **16.04.2019 + 18.04.2019, room 1-1.11 bring your laptop!**
  - Group A, 16.04., 08:15 – 09:45 o'clock
  - Group B, 18.04., 15:45 – 17:15 o'clock
- **ATTENDANCE MANDATORY**

- **One exercise**
- **We will be there to help you if needed**
- **We will ask you questions about your code**
- **Upload necessary at the end of exercise!**

CONTENT

- Repetition
- Java Streams exercise

- Client-Server-Model
- Thin vs. fat client

- File I/O exercise

REPETITION

- Data transmission in distributed systems
- Marshalling/Unmarshalling
- Why needed?

JAVA STREAMS

TECHNISCHE HOCHSCHULE LÜBECK

	Byte based		Character based	
	Input	Output	Input	Output
Basic	InputStream	OutputStream	Reader	Writer
Arrays	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
Files	FileInputStream	FileOutputStream	FileReader	FileWriter
Buffering	BufferedInputStream	BufferedOutputStream	BufferedReader	BufferedWriter
Strings			StringReader	StringWriter
Objects	ObjectInputStream	ObjectOutputStream		

Adapted from http://tutorials.jenkov.com/java-io/overview.html

EXERCISE: STREAMING IN JAVA

- Work with your neighbor!
- Re-organize the following code fragments so the result of java CoordTest is:
  - 12
  - 8
- (you might not need all the fragments, add brackets if needed)

EXERCISE: STREAMING IN JAVA

TECHNISCHE HOCHSCHULE LÜBECK

System.out.println(c.getX() + c.getY() + c.getZ());	ois.close();
c = (Coord) ois.readObject();	oos.close();
class Coord implements Serializable {	import java.io.*;
public static void main(String[] args) {	try {
FileInputStream fis = new FileInputStream(“coords.txt");	oos.writeObject(c);
System.out.println(c.getX() + c.getY() + c.getZ());	System.err.println(e);
ObjectInputStream ois = new ObjectInputStream(fis);	int getX() { return x; }

FileOutputStream fos = new FileOutputStream(“coords.txt");	public int x = 3;
	transient long y = 4;
	private short z = 5;

long getY() { return y; }	fos.writeObject(c);
} catch (Exception e) {	class CoordTest {
short getZ() { return z; }	import java.io.*;
ObjectOutputStream oos = new ObjectOutputStream(fos);	Coord c = new Coord();

CLIENT-SERVER-MODEL

- **Client**
  - Initiates communication
  - Process requesting a service from a server
  - Clients know of servers and their offered services

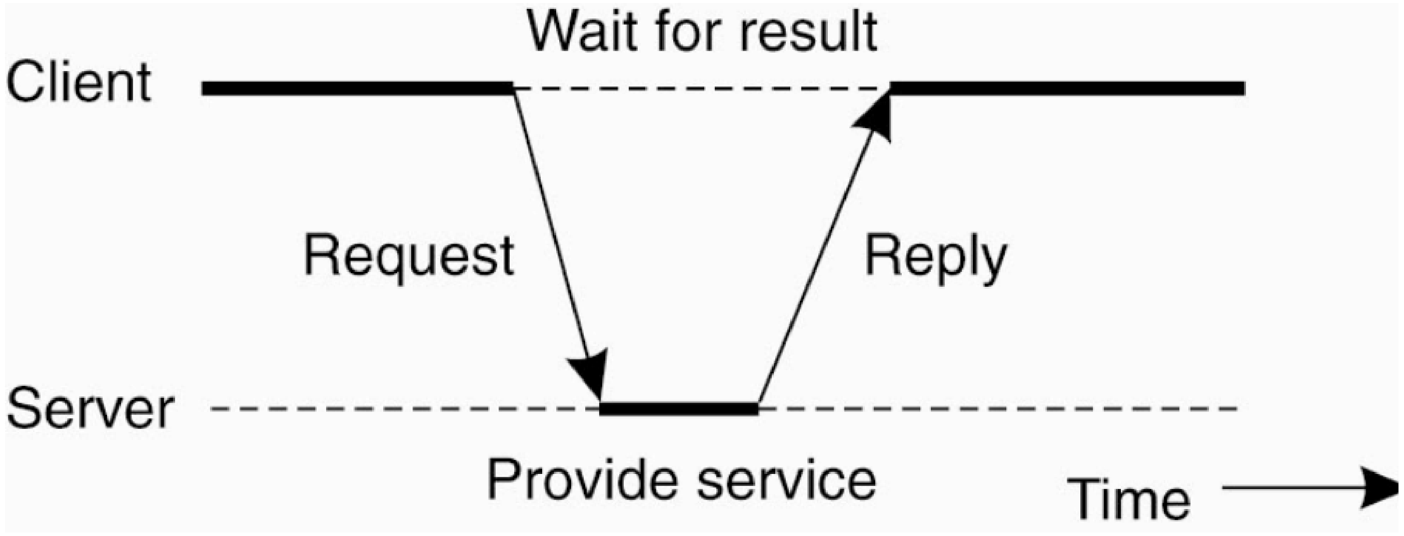
- **Server**
  - Waits for client requests
  - Process implementing services
  - No need of knowing clients

- **Service**
  - Any resource e.g. data, file, control, object, CPU time, display device, …

CLIENT-SERVER-MODEL

TECHNISCHE HOCHSCHULE LÜBECK

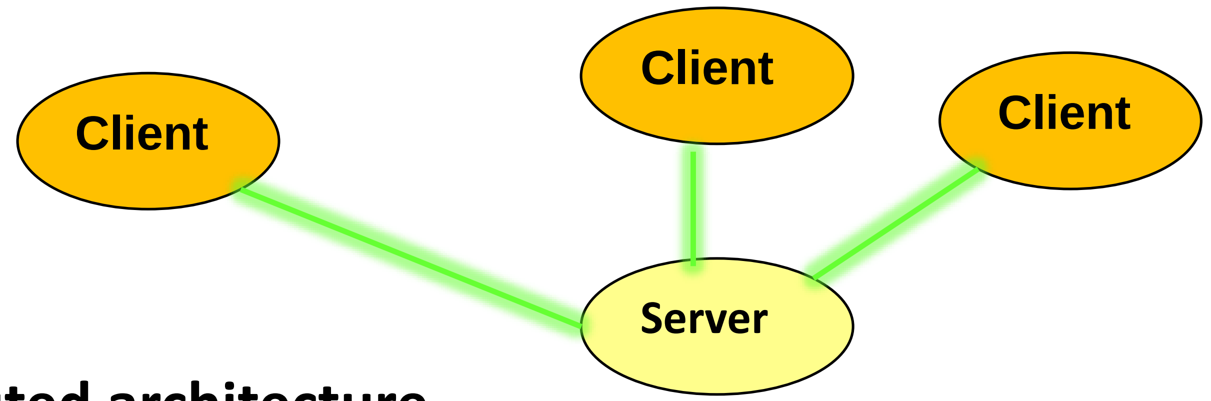
- **Request/response paradigm**



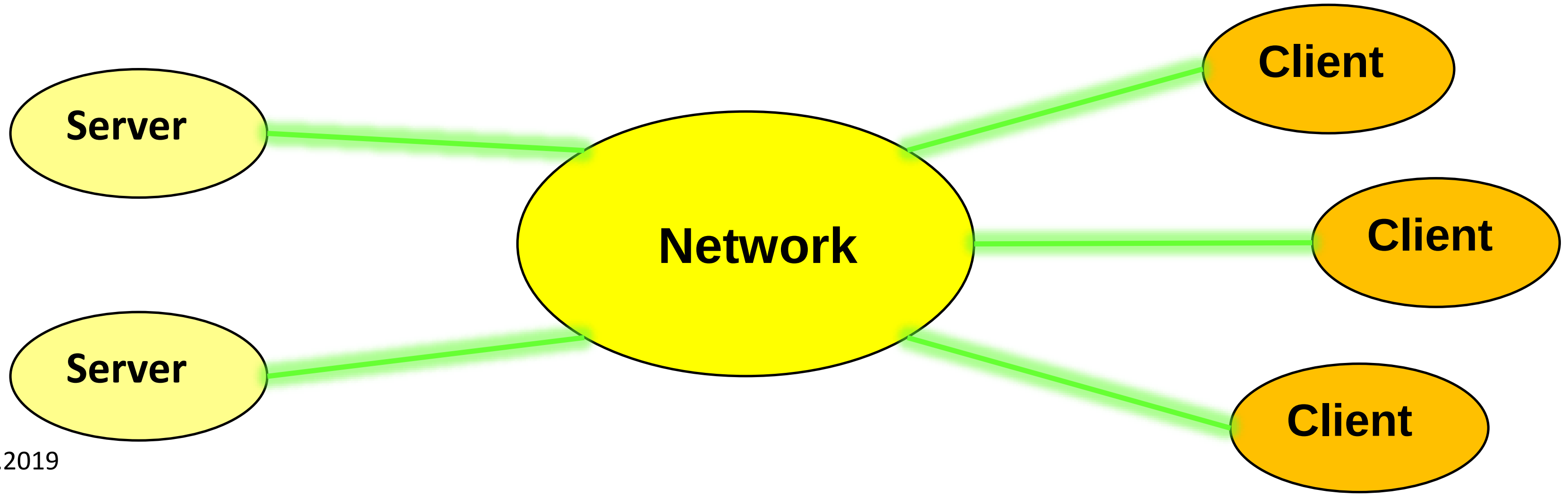
CLIENT-SERVER-MODEL

TECHNISCHE HOCHSCHULE LÜBECK

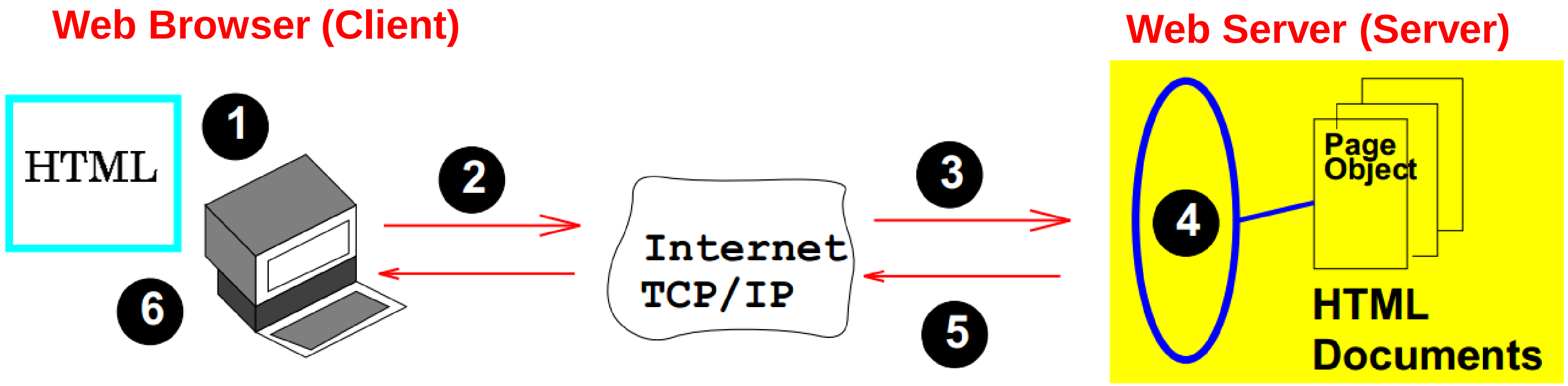
- Centralized architecture



- Distributed architecture



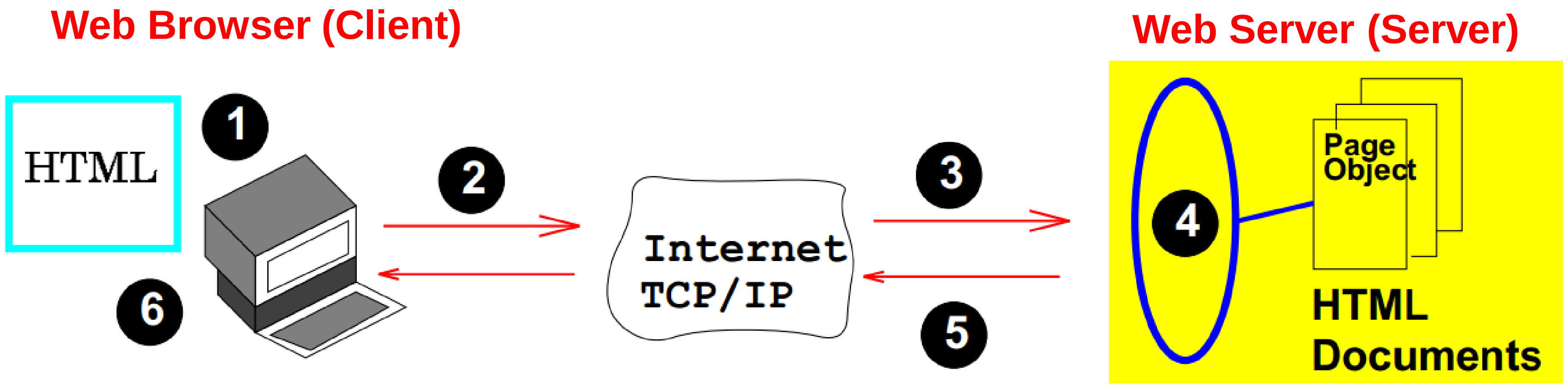
WEB CLIENT – WEB SERVER



1. Select a target URL (Universal Resource Locator - platform-independent naming scheme)
2. Browser takes the URL, embeds it inside an HTTP request, and sends it to the server (HyperText Transfer Protocol, new connection for each user request)
3. HTTP server receives the request on the port, makes a socket connection

WEB CLIENT – WEB SERVER

TECHNISCHE HOCHSCHULE LÜBECK



4. Server finds the requested HTML file (Page Object) and concatenates it with status info (HyperText Markup Language, describes the structure of a Web document)
5. Server ships back the file, and closes the connection
6. Browser interprets the HTML command and displays the page content and invokes a helper application

LAYERING

- **Many client-server systems can be divided into three layers**
  - Presentation/UI layer
  - Application processing/BL layer
  - Data management/DA layer

LAYERING

- **Presentation layer**
  - Presenting the results of a computation to system users and with collecting user inputs

  - Example SWE II project

    →**Graphical user interface**

LAYERING

- **Application processing layer**
    - Providing application specific functionality, e.g. calculating results, log in/log out

    - Example SWE II project

        → **Application part**

LAYERING

- **Data management layer**
    - Managing system databases

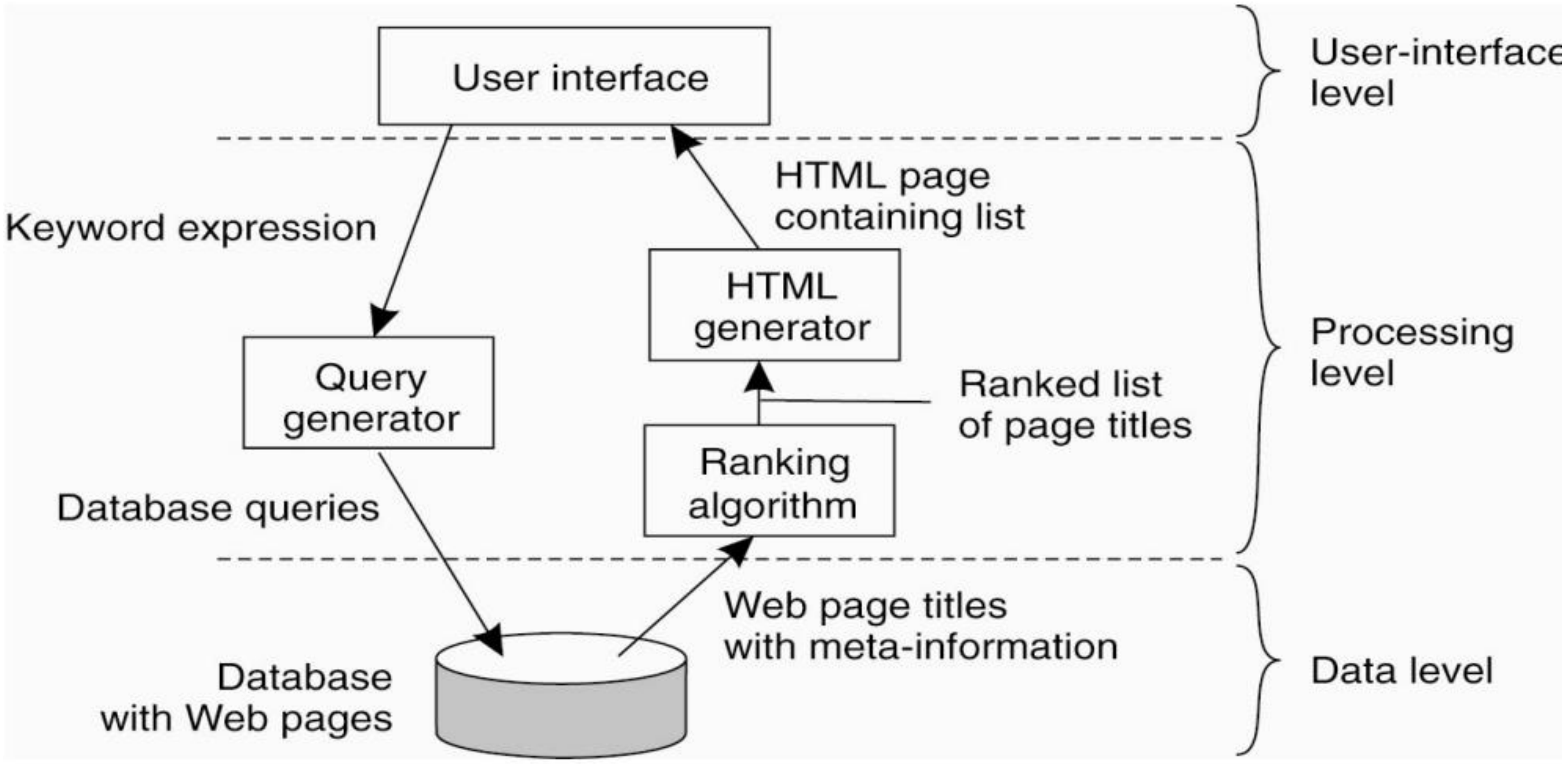
    - Example SWE II project

        → **Database interface**

LAYERING - EXAMPLE

TECHNISCHE
HOCHSCHULE
LÜBECK

- **Internet search engine**



DISTRIBUTION OF LOGIC: THIN VS. FAT CLIENT

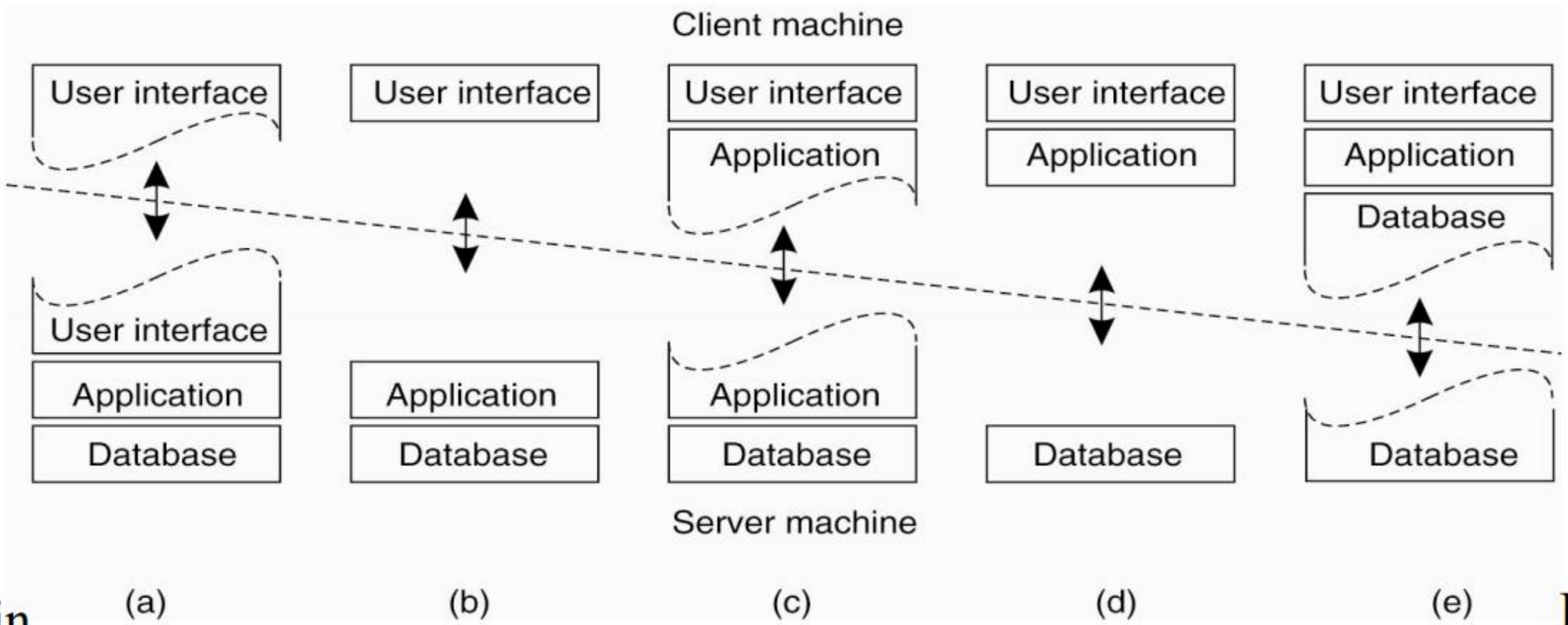
- **Thin client**
  - All of the application processing and data management is carried out on server.
  - The client is simply responsible for running the presentation software.

- **Fat client**
  - Server is only responsible for data management.
  - Software on client implements application logic and interactions with system user.

THIN VS. FAT CLIENT

TECHNISCHE HOCHSCHULE LÜBECK

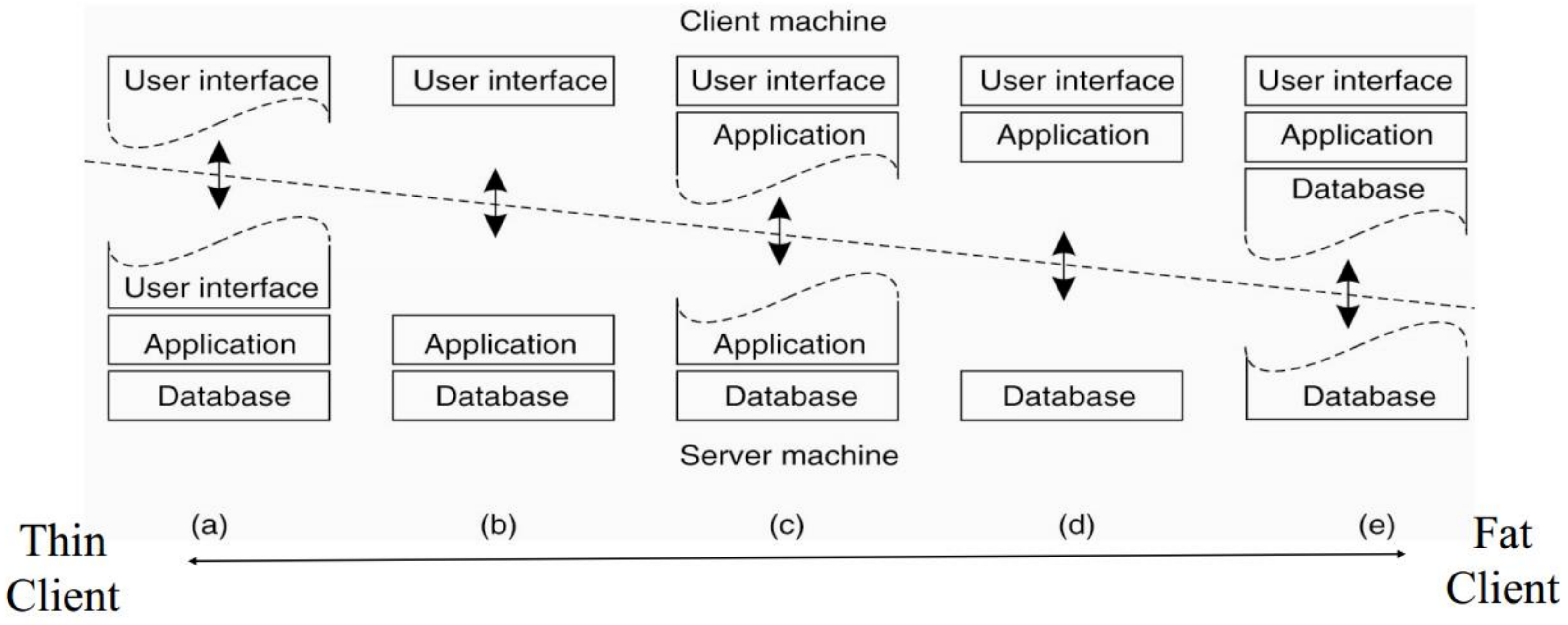
Two-tier client-server-model



Thin Client ⟷ Fat Client

THIN VS. FAT CLIENT

- **Homework!**
  - **Find examples for all cases a,b,c,d,e!**
  - **Write them down and be able to explain them in the next lecture!**



EXERCISE: FILE HANDLING IN JAVA

TECHNISCHE HOCHSCHULE LÜBECK

- Realize a class *IOFile* that exports some functionalities on text files.
- The class should have a constructor with one parameter of type String, representing the name of the file on which to operate, and should implement the following methods:
  - int countLines(): count line amount of file
  - void print(): print file on command line
  - void copy(String filename): copy file content to the file ‘filename’
  - void delete(): delete the file
  - void printDirectory(): prints the file directory
  - List<String> getOtherFiles(): returns list of other files in same directory as file
- Addtionally, create a test class which demonstrates the functionality of your IOFile class!

Upload for feedback!

Useful for next week’s exercise as well as Java Swing!

REFERENCES

- P. Nieuwenhuysen, “Client-Server-Systems”, http://www.vub.ac.be/BIBLIO/nieuwenhuysen/courses/chapters/client-server.pdf
- L. Chung, “Computer Science Program”, https://www.utdallas.edu/~chung/SA/2client.pdf
- U. Krohn, Slides from “Verteilte Systeme”

TECHNISCHE HOCHSCHULE LÜBECK

Next lecture:

26.04.2019, 10:00-11:30, room 1-1.11

Questions?

Lenka Kleinau
Contact: lenka.kleinau@th-luebeck.de
Room 18-2.02