

Practical Java

Kristof Werling

April 2022

Organizational Info

Each day of the lecture

- 2 Parts
 - Part 1:
 - Learn about new Java concepts and functionality
 - Easy and short exercises to enhance the understanding of the material
 - Part 2:
 - Use the new concepts and functionality of Part 1 to enhance and extend the Converter project

Part 1 - more details

- There will be exercises
- Each exercise comes with a solution
- Each exercise will be discussed with the whole group and problems / issues will be addressed
- The solution provided also will be discussed

Depth of the information provided

- For the most part the information provided is sufficient to work out the solution to the exercises
- For most of the concepts and functionality shown there is a vast body of knowledge we cannot explore in any kind of practical manner.

Project for this lecture

- Converter: Markdown to Latex and later to Html
- Simple solution.
- Each lecture works on one aspect of the solution (like: GUI, DB, ...)

Markdown Tags

| Headings | Text Formatting | Rendered Output |
|-----------------------|---|---|
| # Heading level 1 | **This is bold text** | Link to [Google](https://www.google.com/) |
| ## Heading level 2 | __This is bold text__ | - Unordered List Item 1 |
| ### Heading level 3 | <i>*This text is italicized*</i> | - Unordered List Item 2 |
| #### Heading level 4 | ~~Strikethrough text~~ | - Unordered List Item 3 |
| ##### Heading level 5 | <i>***Bold and italics text***</i> | 1. Ordered List Item 1 |
| ##### Heading level 6 | <i>**Bold and *nesting italics* text**</i> | 1. Ordered List Item 2 |

Source: <https://www.markdownguide.org/basic-syntax>

Or

<https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

All the gory details (specs): <https://github.github.com/gfm/>

Latex

Boiler Plate for a Latex Document

```
\documentclass[12pt, a4paper] {article}
```

```
\begin{document}
```

Here goes the document.

```
\end{document}
```

Source: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

Latex

| | |
|----------------|--|
| Bold | This is <code>\textbf{bold text}</code> |
| Italic | This is <code>\textit{text in italic}</code> |
| Strikethrough | |
| Unordered List | <code>\begin{itemize}</code> |
| | <code>\item Item 1</code> |
| | <code>\item ...</code> |
| Ordered List | <code>\end{itemize}</code> |
| | <code>\begin{enumerate}</code> |
| | <code>\item Item 1</code> |
| | <code>\item ...</code> |
| Link | Use the <code>hyperref</code> package |

Source: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

Latex

| | |
|-----------|--|
| Heading 1 | <code>\section{section}</code> |
| Heading 2 | <code>\subsection{subsection}</code> |
| Heading 3 | <code>\subsubsection{subsubsection}</code> |
| Heading 4 | <code>\paragraph{paragraph}</code> |
| Heading 5 | <code>\subparagraph{subparagraph}</code> |

Source: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

What we need to get started

- IntelliJ Community Edition [\(Download here\)](#)
- MiKTeX (or any other Tex that can process Latex) [\(Download here\)](#)
- OpenJDK Java 18 [\(Download here\)](#)
- Markdown Viewer [\(For example: Windows Markdown Viewer\)](#)
- Git for Windows [\(Download here\)](#)

Github.Com Account



Product ▾ Team Enterprise Explore ▾ Marketplace Pricing ▾

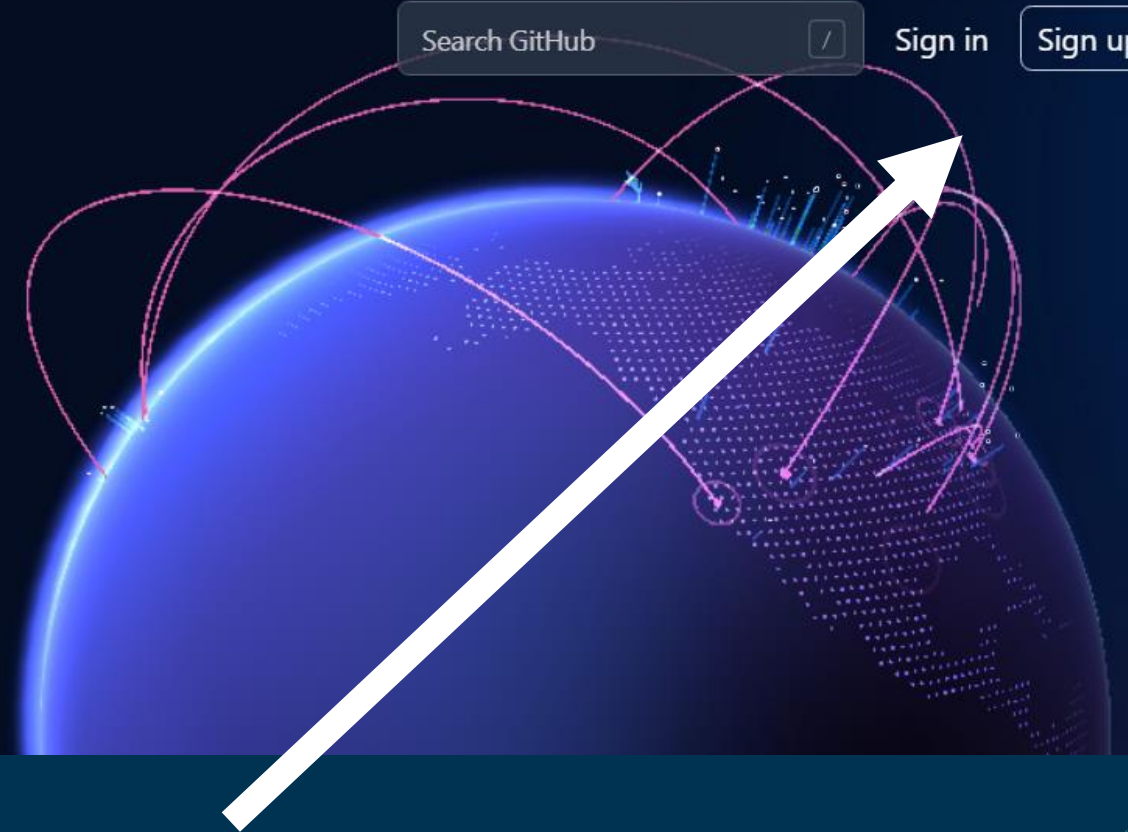
Search GitHub



Sign in

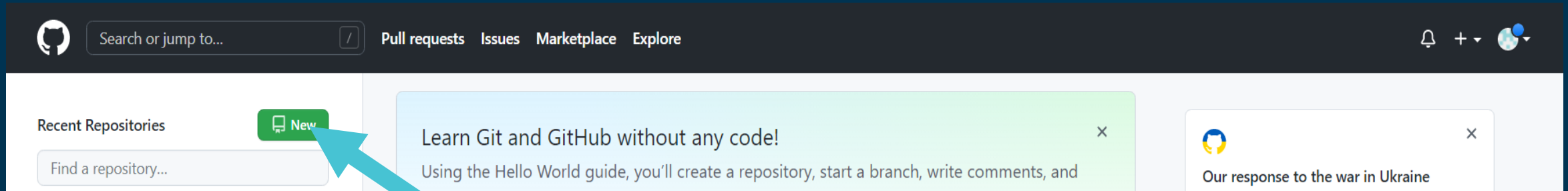
Sign up

Where the world builds software



Sign in or Sign up

Create new Repository




New Repository

Create new Repository 1 of 2

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

 kwerling-MM ▾

Repository name *

/ myName 

Great repository names are short and memorable. Need inspiration? How about **fuzzy-committing machine**?

Description (optional)

This is the comment to my new Repository

☐



Public

Anyone on the internet can see this repository and can commit to it.

☒



Private

You choose who can see and commit to this repository.

Repository name

Comment, if wished

Private or Public access

Create new Repository 2 of 2


Initialize this repository with:
Skip this step if you're importing an existing repository.


☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☒ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Java ▼

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

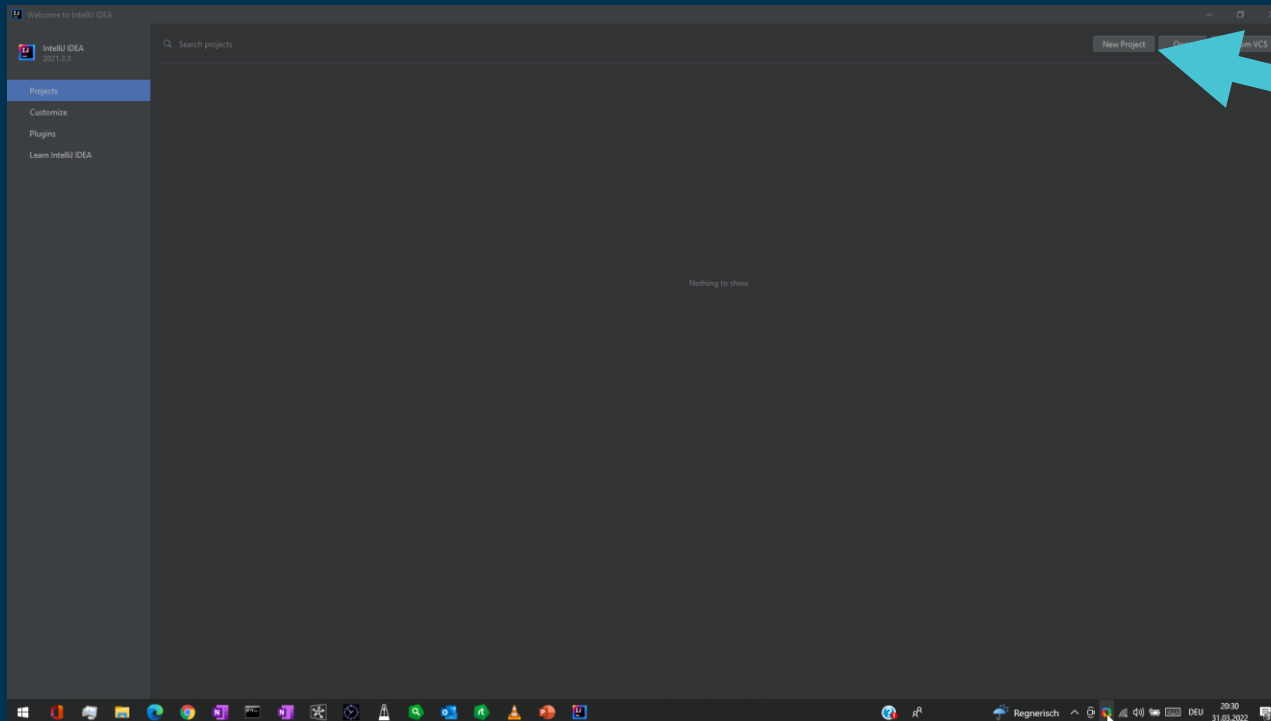
 You are creating a private repository in your personal account.

Create repository

Add .gitignore for JAVA

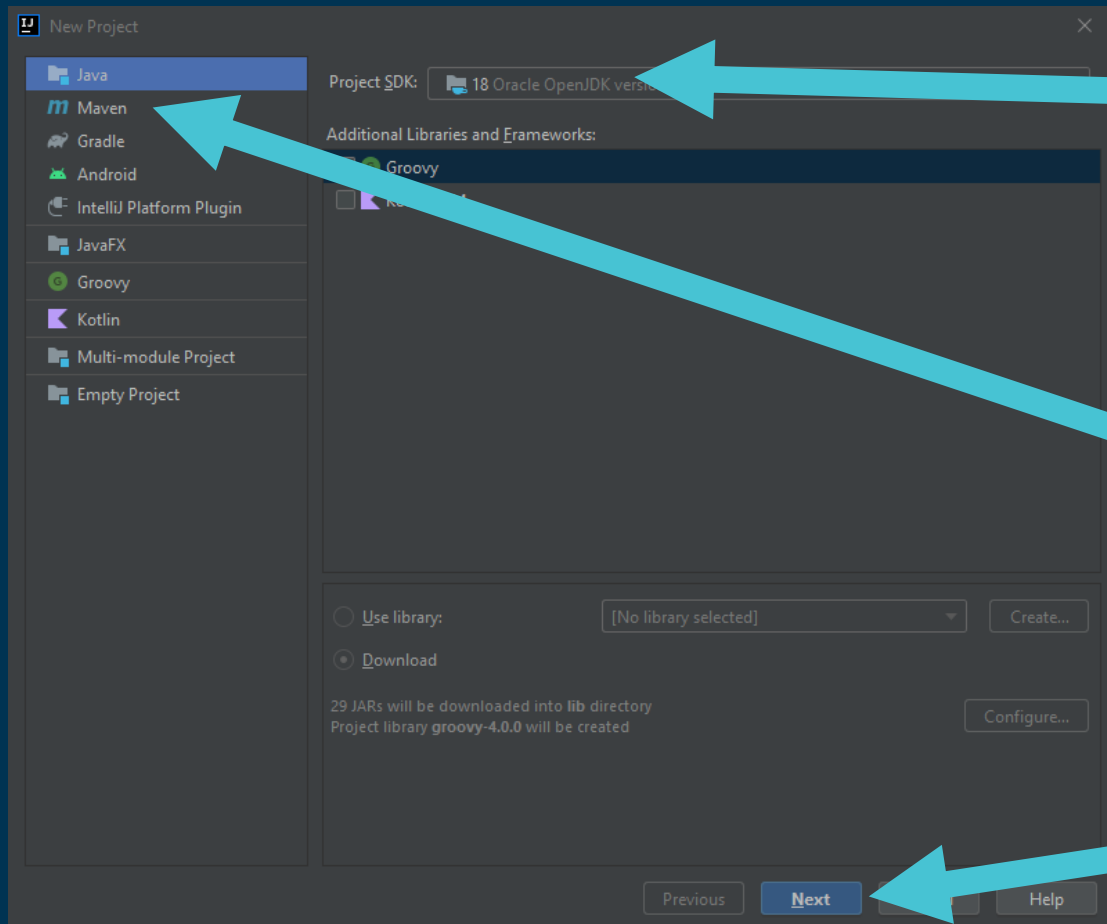
Create it

Creating a new project 1 of 5



New Project

Creating a new project 2 of 5

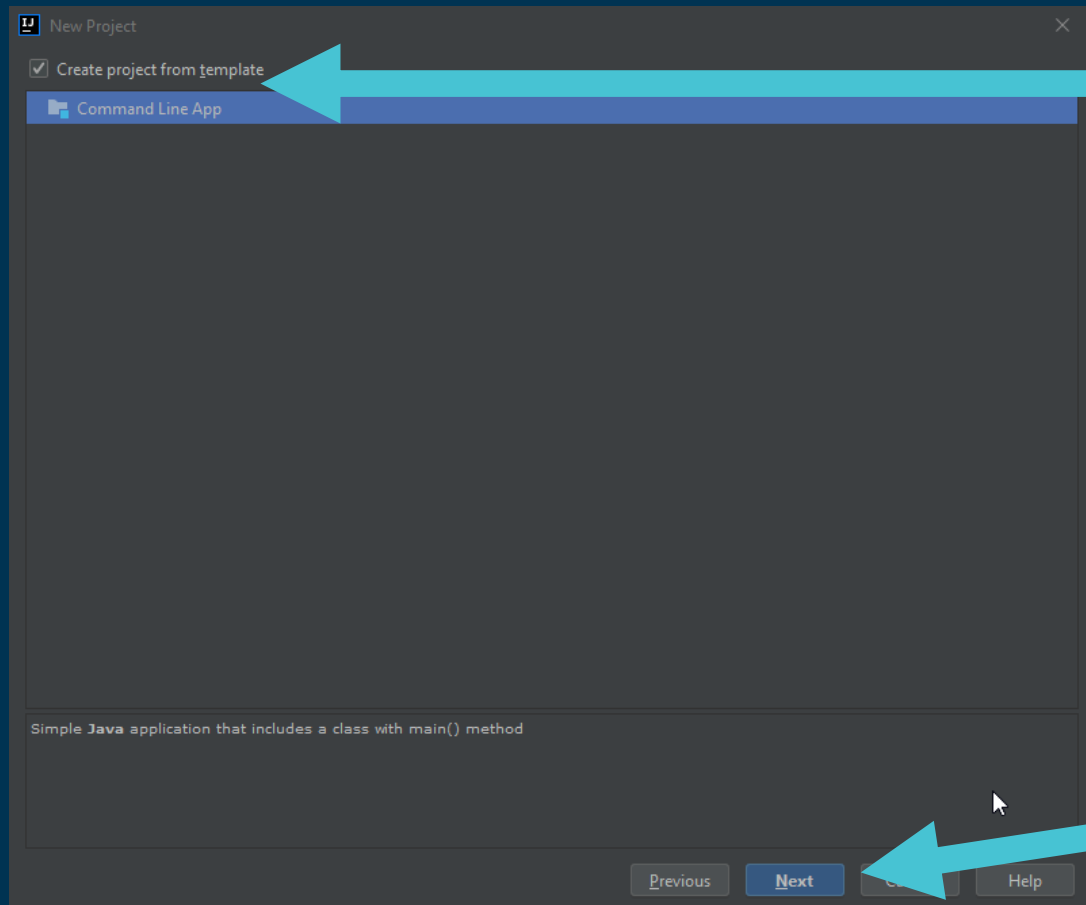


**Make sure your
JDK is listed
here**

Java

Press Next

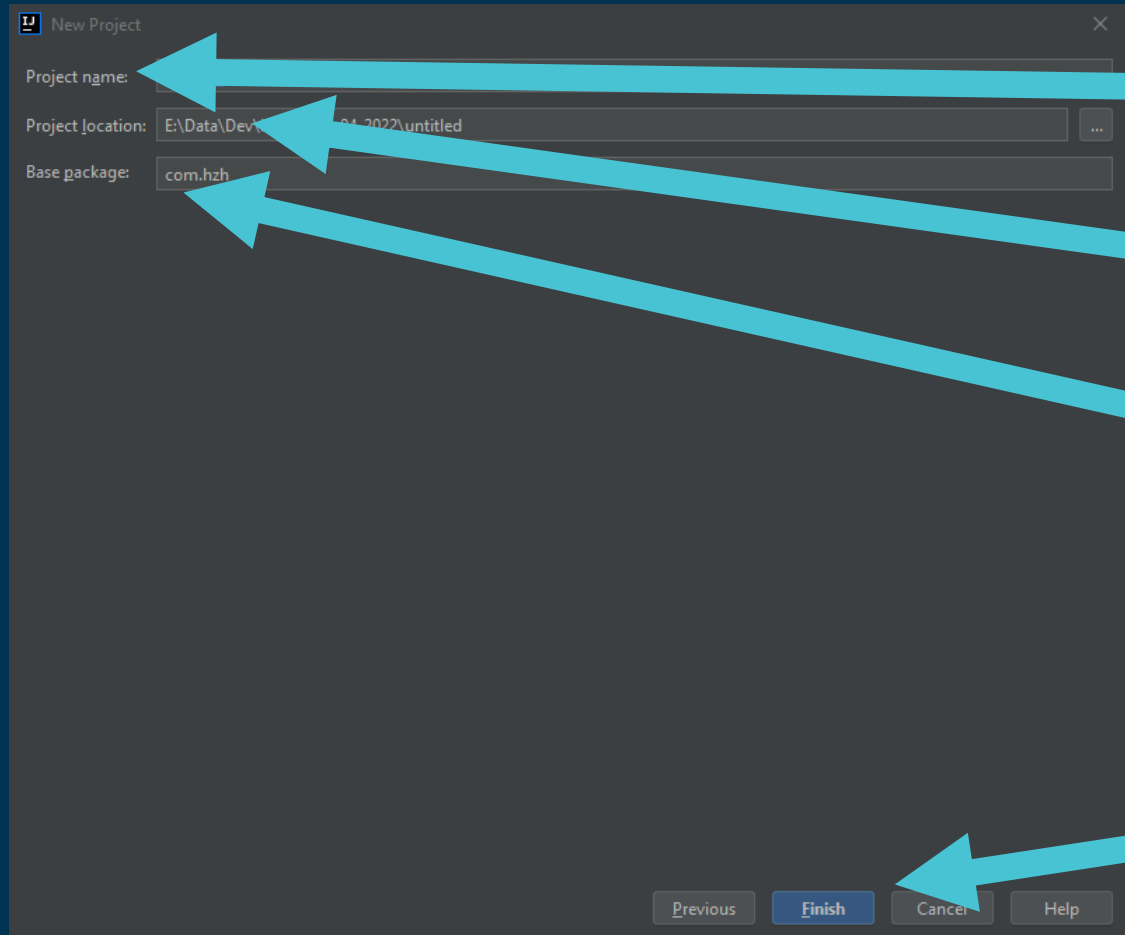
Creating a new project 3 of 5



**Choose the
Commend Line
Template to get
started**

Press Next

Creating a new project 4 of 5



The screenshot shows a 'New Project' dialog box with the following fields and buttons:

- Project name:** An empty text field.
- Project location:** A text field containing 'E:\Data\Dev\2022\untitled' with a browse button (three dots) to its right.
- Base package:** A text field containing 'com.hzh'.
- Buttons:** 'Previous', 'Finish', 'Cancel', and 'Help' at the bottom.

Four red arrows point from text labels on the right to specific parts of the dialog:

- An arrow points from 'Give it a name' to the 'Project name' field.
- An arrow points from 'Local directory' to the 'Project location' field.
- An arrow points from 'Name space' to the 'Base package' field.
- An arrow points from 'Press Next' to the 'Finish' button.

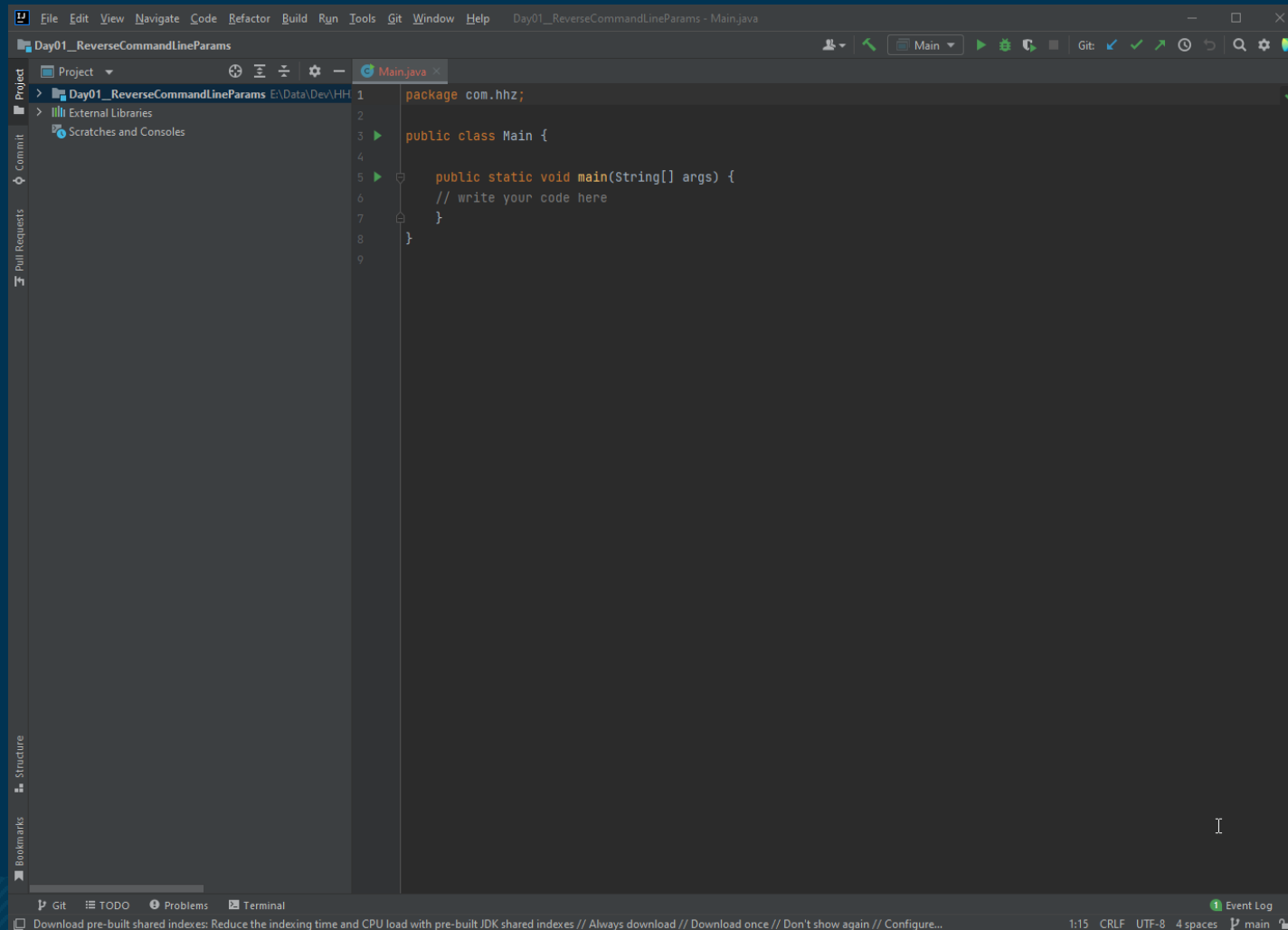
Give it a name

Local directory

Name space

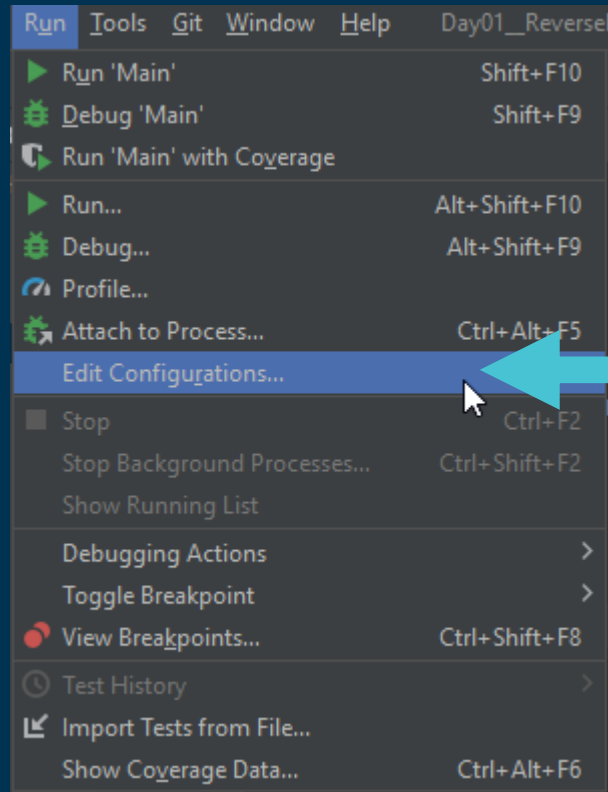
Press Next

Creating a new project 5 of 5



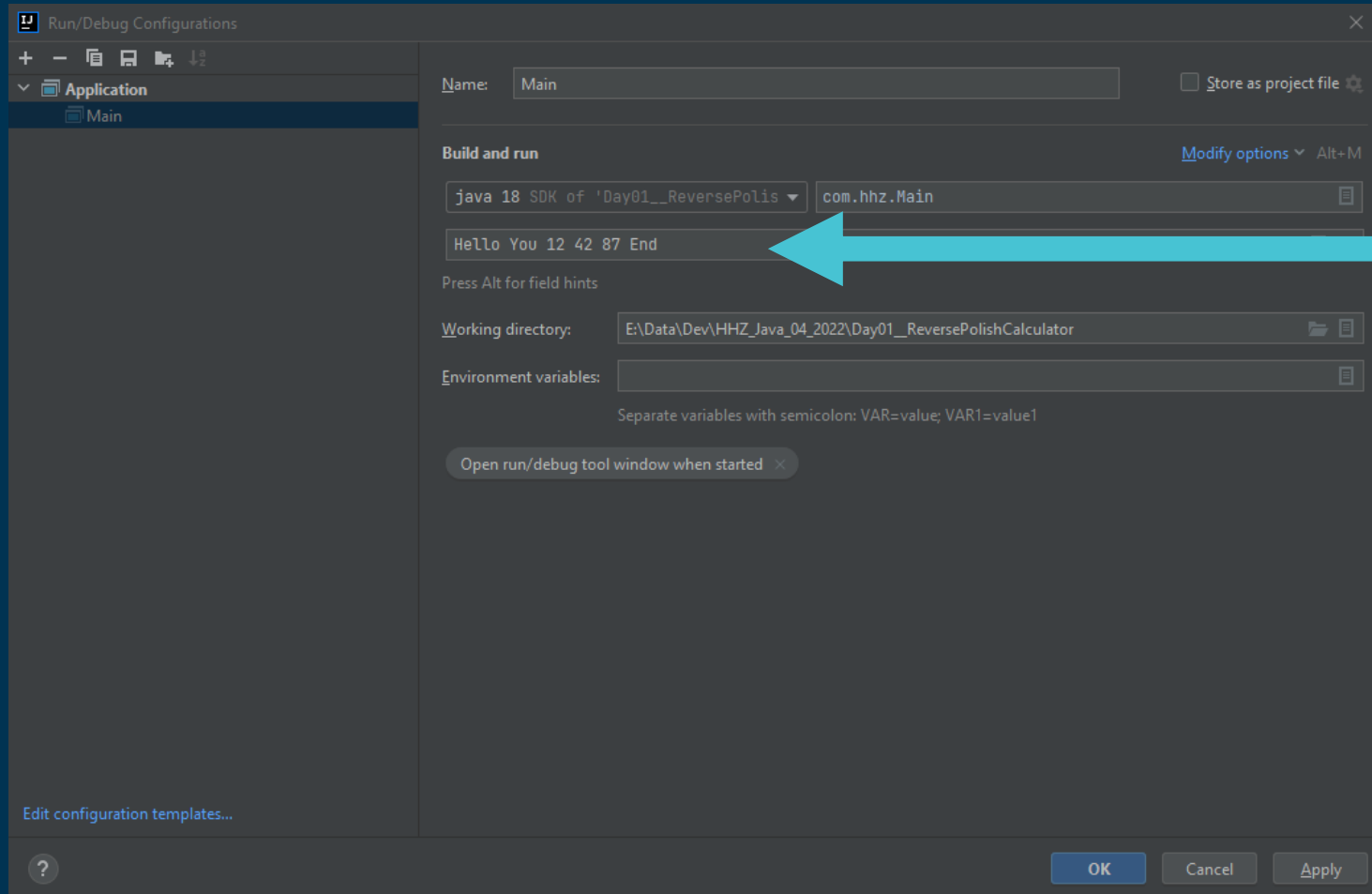
Resulting project

Run with Command Line Params 1 of 2



Adjust the configuration

Run with Command Line Params 1 of 2



Here go the command
Line params

Table of Content

- Project used in the lecture
- Day 1: Playing with Java / Deepen the knowledge
- Day 2: User Interfaces
- Day 3: Networking – From Socket to Message Bus
- Day 4: Working with Databases (SQL and No-SQL)
- Day 5: Wrap-Up and Overflow

Command line parameters

- The main function takes the command line parameters in an String array
- Each parameter is passed on as a String (String array after all)
- In case of no command line parameters the String array is empty

Exercise 01

- Write a program, which prints out the command line parameters in reverse order

Exercise 01 - Solution

```
1 package com.hhz;  
2  
3 public class Main {  
4  
5     public static void main(String[] args) {  
6         for( int i = args.length; i>0; i--) {  
7             System.out.println("Param #" + i + ": " + args[i-1]);  
8         }  
9     }  
10 }  
11
```

Some methods of the String class

| (some) String class method | Functionality |
|---|---|
| <u>String toLowerCase()</u> | It returns a string in lowercase. |
| <u>String toUpperCase()</u> | It returns a string in uppercase. |
| <u>String trim()</u> | It removes beginning and ending spaces of this string. |
| <u>int indexOf(String substring)</u> | It returns the specified substring index. |
| <u>String[] split(String regex)</u> | It returns a split string matching regex. |
| <u>boolean contains(CharSequence s)</u> | It returns true or false after matching the sequence of char value. |
| <u>int length()</u> | It returns string length. Compare to Array.length !! |
| <u>String substring(int beginIndex, int endIndex)</u> | It returns substring for given begin index and end index. |

Exercise 02 - Playing with String comparison

- Take the code on this slide
- Run it
- Explain the results



Main.java

Exercise 03 - Playing with String concatenation

- Take the code on this slide
- Run it
- Explain the results



Main.java

Integer class - parsing of text

- `int Integer.parseInt(String)`

tries to convert the String into an integer value. Throws an exception if that not possible.

| | | | |
|-----|--------------------------------------|---|------------------|
| Ex: | <code>Integer.parseInt(„411“)</code> | → | 411 |
| | <code>Integer.parseInt(„Axx“)</code> | → | Throws exception |

Try - catch - finally

- In order to control code, which might throw exceptions it is enclosed in a try-catch (-finally) construct:

```
try {  
    // Code, which might throw exeptions  
} catch( Exception ex ) {  
    // Code to run if an exeption happened  
} finally {  
    // Code, which runs wether an exeption was thrown  
}
```

Exercise 04

- Write a program, which prints out the command line parameters in reverse order.
- Add 10 to each integer value in the list before printing it out.

Class Stack

- Grows (aka Push operation) upwards
- Shrinks (aka Pop operation) downwards
- There are only the push and pop operations for accessing the stack.

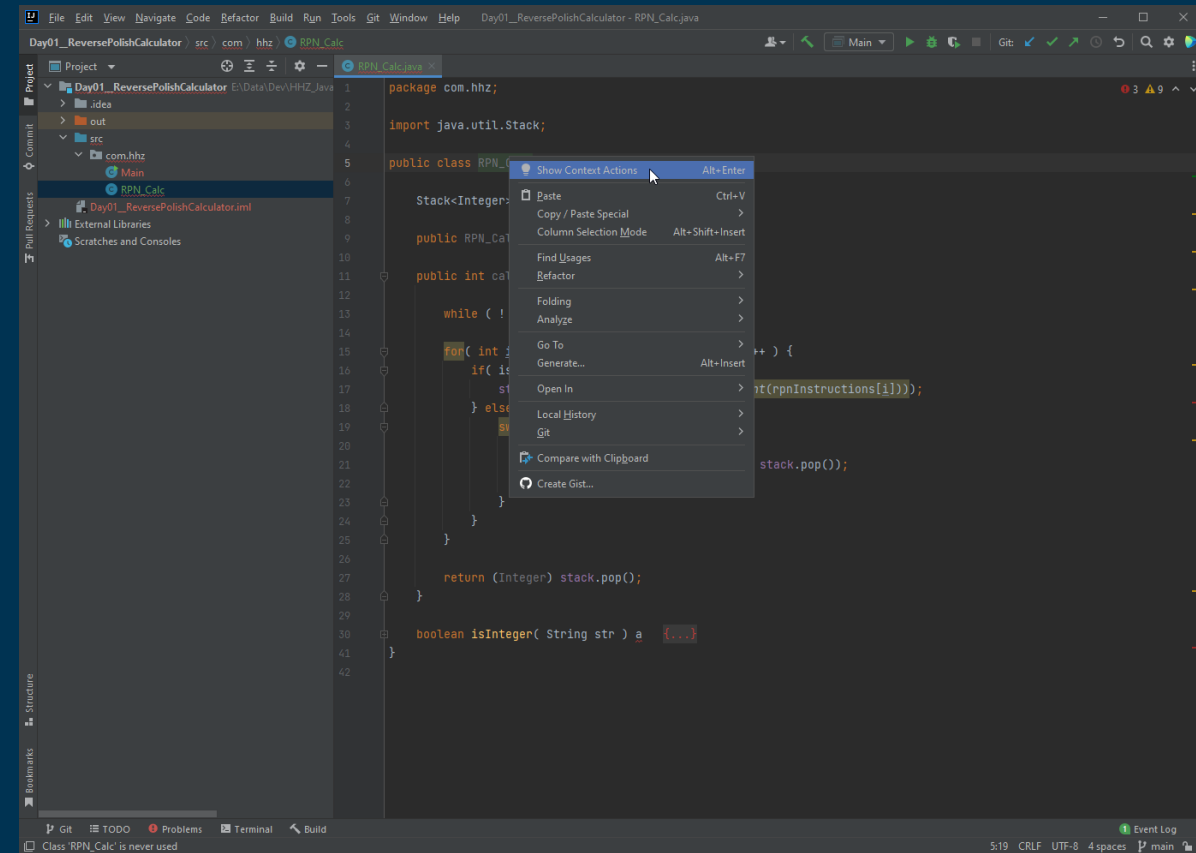


Unit testing

- Small test of parts of code
- Always test one thing and one thing only
- Expected to run fast
- YES, I know of projects where the code for testing exceeded the code under test.

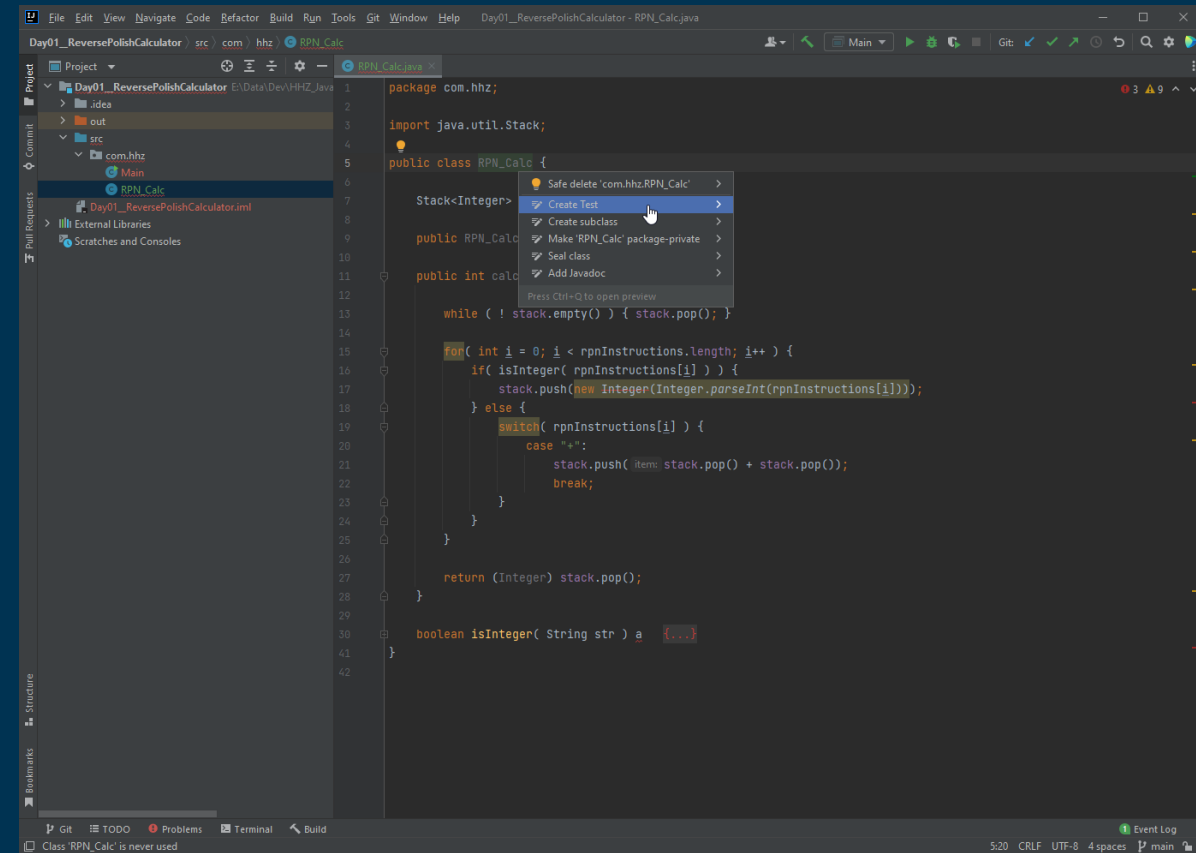
Unit testing

- Choose „Show Context Action“



Unit testing

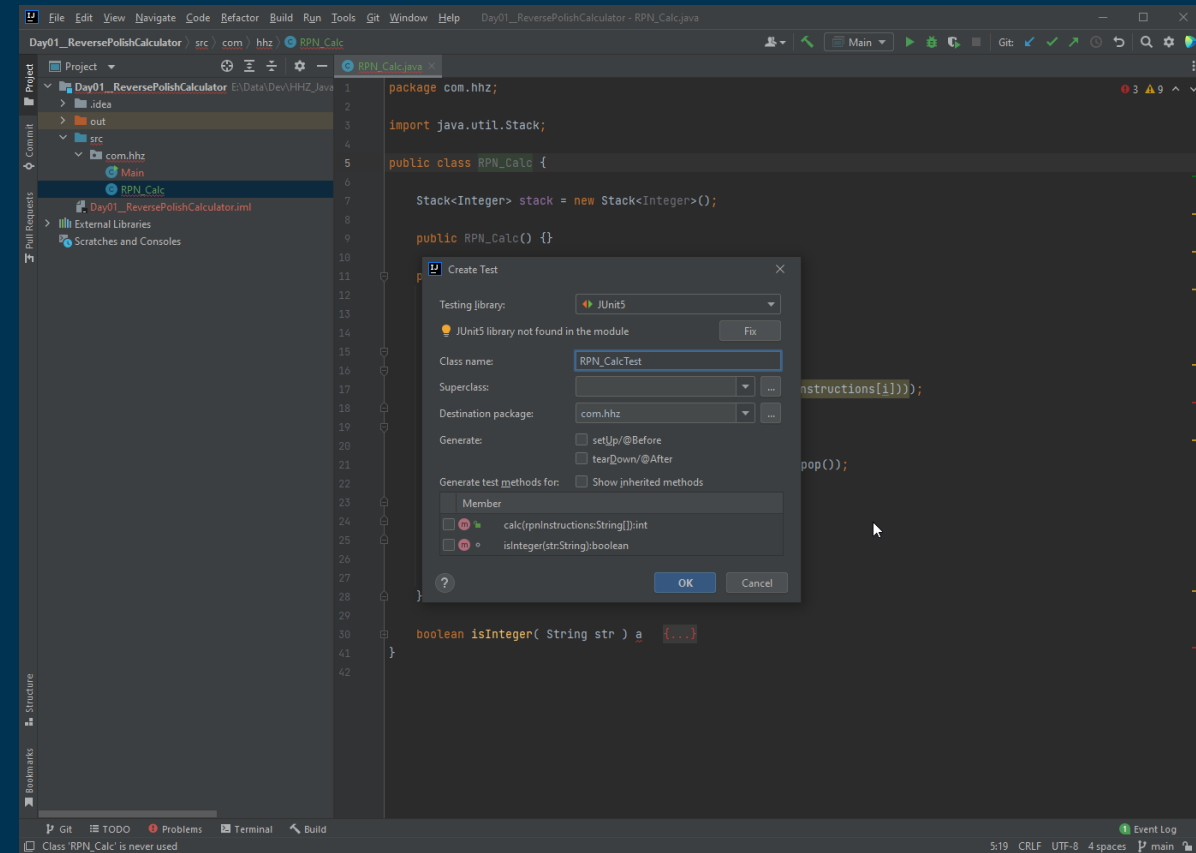
- Choose „Create Test“



Unit testing

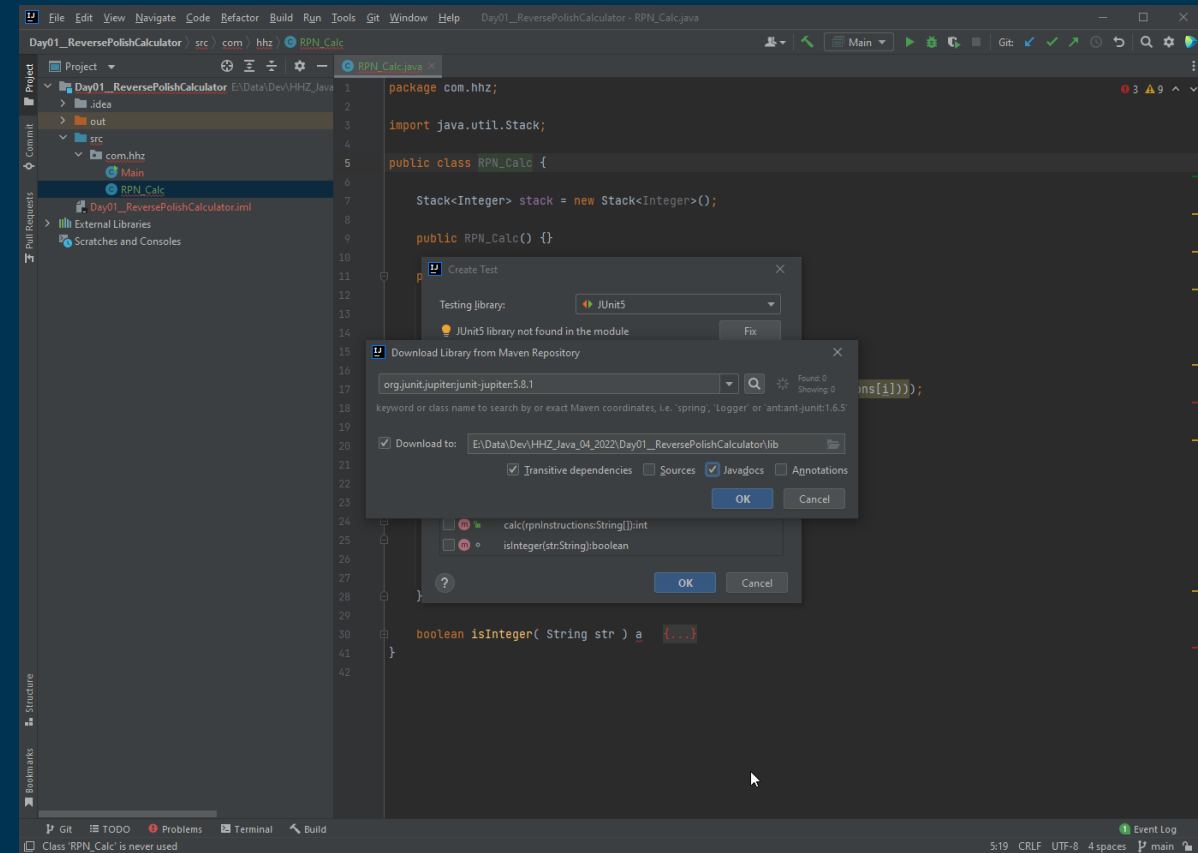
- When done for the first time the Junit jar file needs to be added to the project.

Press „Fix“



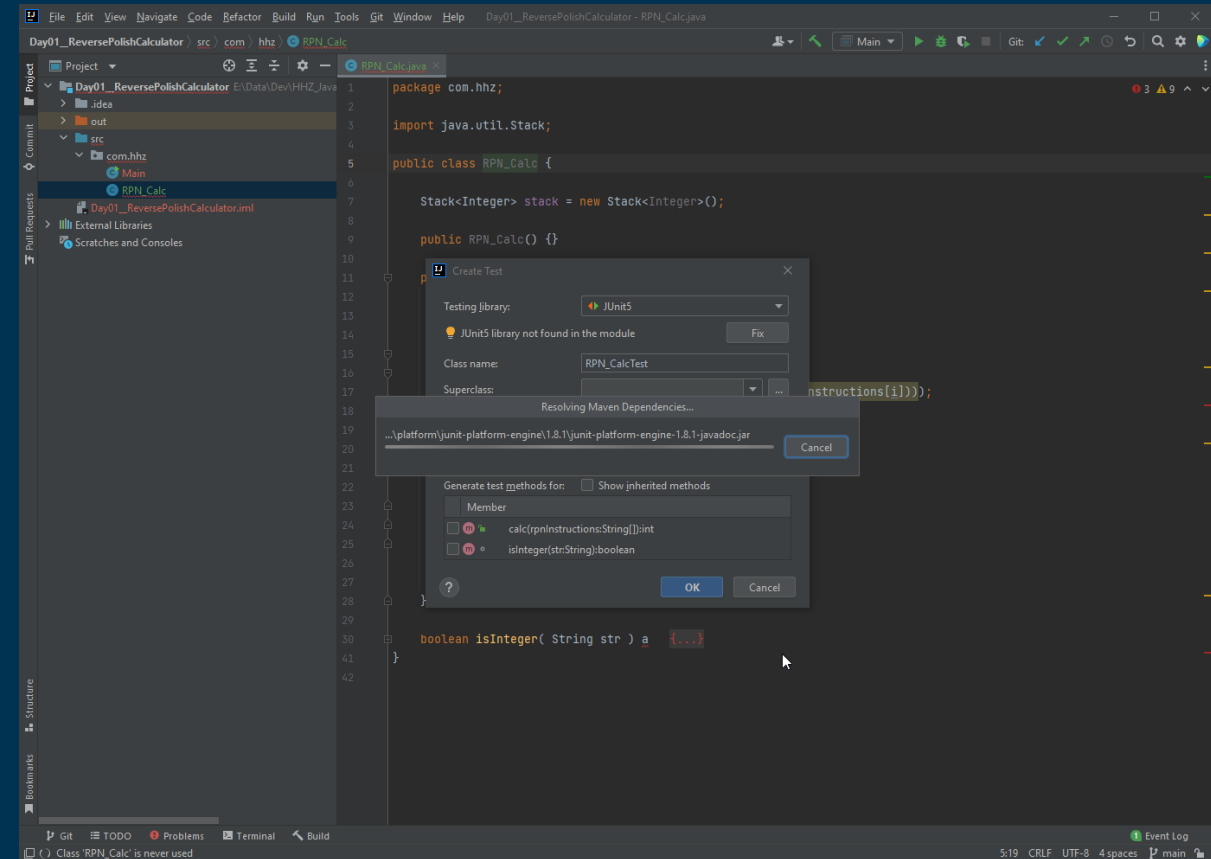
Unit testing

- Add the Junit jar file
Download Javadoc as well.



Unit testing

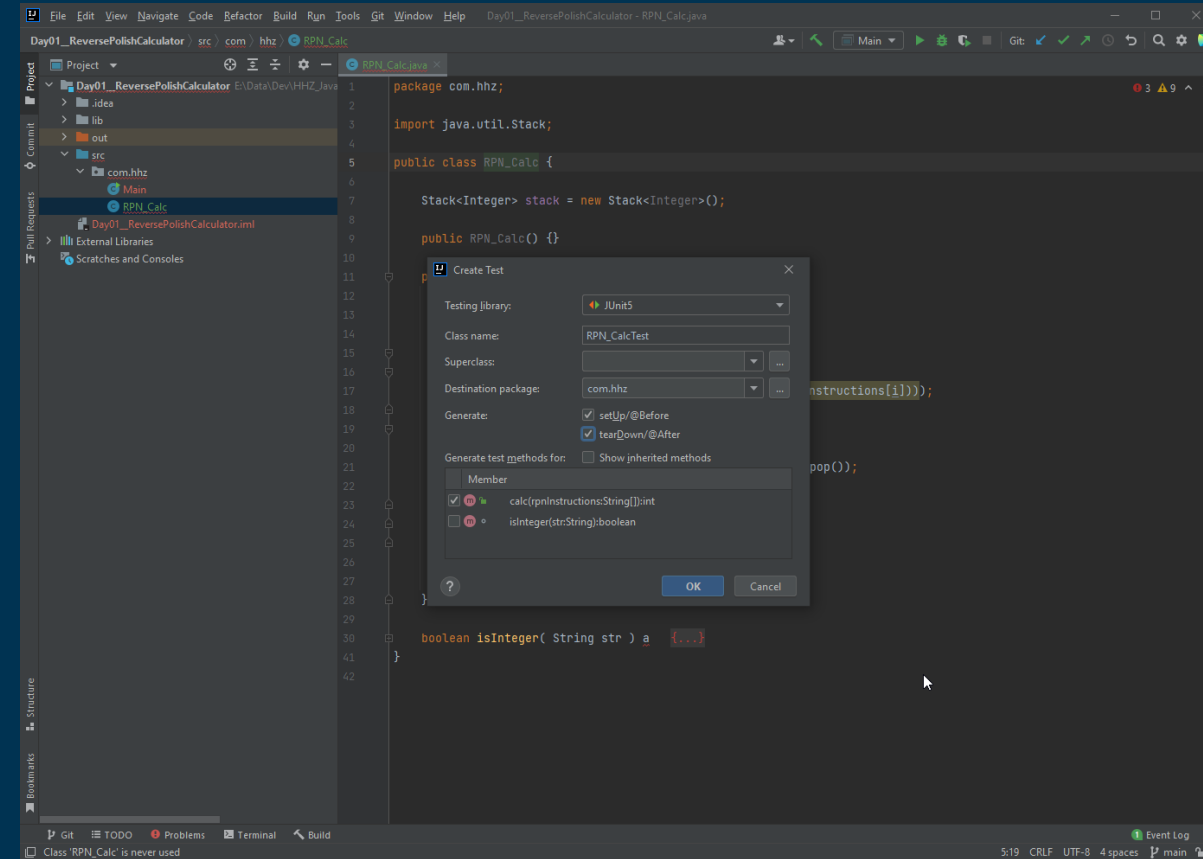
- IntelliJ downloads the required files



Unit testing

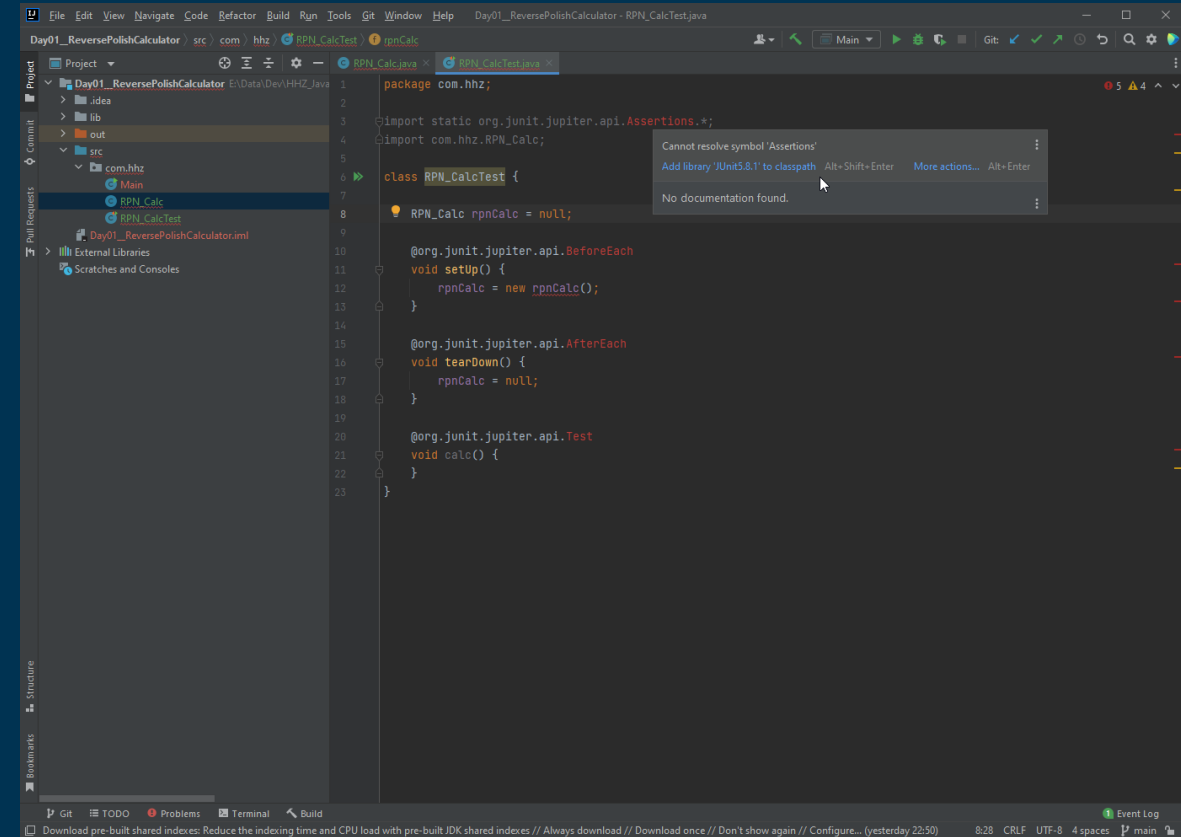
- Create the Junit test skeleton for the calc method

Create the Setup & TearDown methods



Unit testing

- Need to add the Junit library to the classpath.



Exercise

- Create a class Calc
 - With one method `int add(int a, int b)`, which returns the sum of a and b
 - Create multiple tests for this method

UPN

- Add 10 and 20:

 $10\ 20\ +$

- $3*(4+2)$

 $4\ 2\ +\ 3\ *$

Exercise 05

- Create a class RPN, which has this method:
 - `int calc(String [] rpnInstructions);`
- It takes a string array as input, which contains the operands and the operators in correct order for the calculation
- It returns the result of the calculation
- Only integer values are used in the calculation
- Add Unit testing for verification of the correctness of the class

Exercise Converter

- Create a Converter class, which is able to take in a text string in Markdown text and convert it into Latex
- For starters we need to translate the Headings first
- The first character in a line is a ,#‘, maybe followed by more of them
- The first non-‘#‘-character to the end of the line is the Header text
- All other text (Markdown tags or not) is simply copied to the Latex file.
- Create a Main-method, which takes the Markdown-Filename from the command line.
- The Latex file has the same file name as the Markdown file, just ending in ,.latex‘
- Feel free to convert other Markdown tags as well

Exercise Converter - Hints

- `ArrayList<String>`
- String class:
 - `charAt`
 - `indexOf`
 - `substring`
 - `trim`

SWING - User Interface

A simple Swing application

- JFrame is the class providing a window for an application.
- Inside of the window there will be components for user information and interaction

Excercise 1

- Build an application, which shows an empty window
- Play with it. Is there any difference to other windows from other applications?

Hints

- `JFrame.setSize(int, int)` allows to set the dimensions of the window.
- In order to end the application when the window gets closed one needs to add an event listener:

```
window.addWindowListener(new WindowAdapter() {  
    •     public void windowClosing(WindowEvent windowEvent){  
    •         System.exit(0);  
    •     }  
    • });
```

Components

- There are many different components available.
- We will start out with JButton and JLabel

Excercise 2

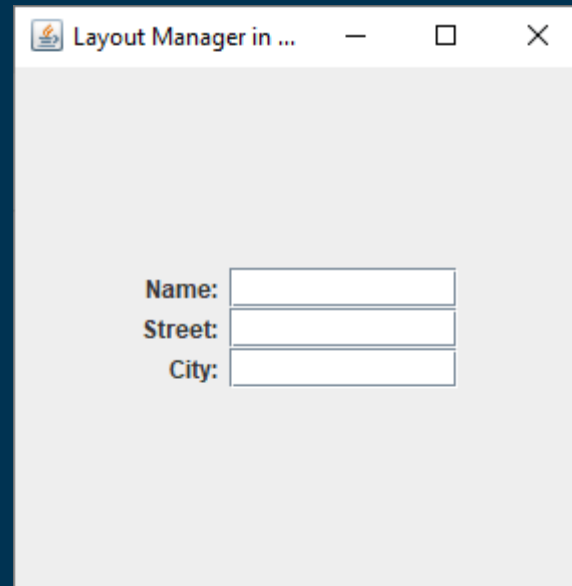
- Add a button to the window
- Anything the matter?
- Try adding a second button
- What happens then?

Layout manager

- In order to achieve control over the positioning of the different components in a window Swing offers different Layout managers.
- Layout managers can be nested.
- [The different Layout managers are shown here.](#)

Excercise 3

- Use the GridbagLayout Manager to construct the following screen with JLabels and JTextFields(width 10)



A screenshot of a Java Swing window titled "Layout Manager in ...". The window contains a form with three labels and text fields arranged vertically. The labels are "Name:", "Street:", and "City:", each followed by a text field. The text fields are empty and have a width of 10.

| | |
|---------|----------------------|
| Name: | <input type="text"/> |
| Street: | <input type="text"/> |
| City: | <input type="text"/> |

Working with events

- In order to react to the push of a button it (the button) needs to listen to such an event.

```
button.addActionListener(new ActionListener() {
```

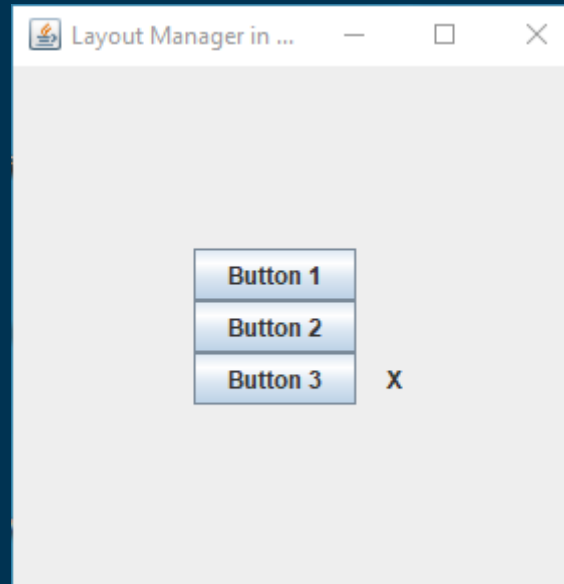
- `@Override`
- `public void actionPerformed(ActionEvent e) {`
- `System.out.println(((JButton)e.getSource()).getText());`
- `}`
- `});`

See [Different ways to implement a listener](#)

Excercise 4

- Create a little application, which shows an X next to the button, which was clicked last:

Example:



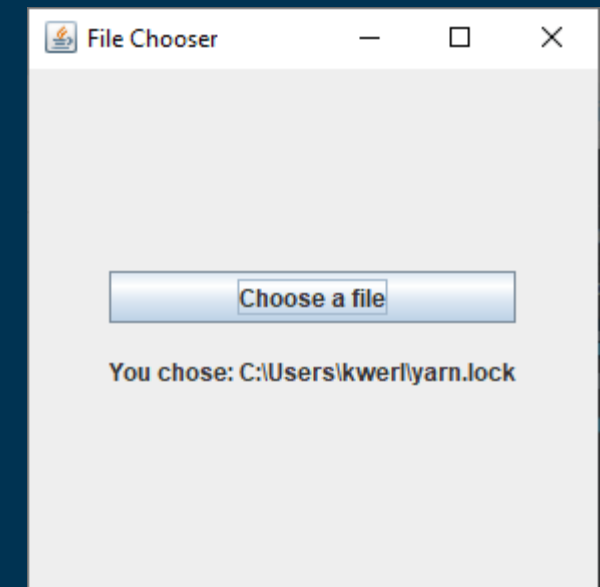
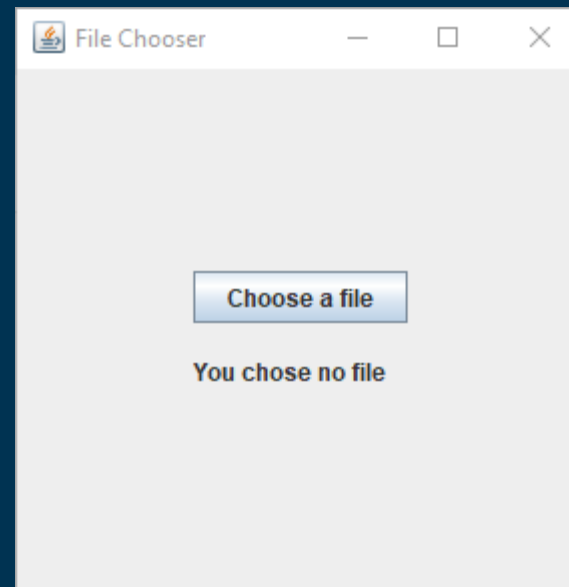
Picking a file name

- JFileChooser brings up a dialog box for choosing a file name
- ```
JFileChooser jfc = new JFileChooser(new File(System.getProperty("user.home")));
```
- ```
int result = jfc.showOpenDialog(parent_window);
```
- ```
if (result == JFileChooser.APPROVE_OPTION) {
```
- ```
    // jfc.getSelectedFile() gets the file name
```
- ```
 // Work with the file name here
```
- ```
}
```

Excercise 5

- Create an application, which shows the file name chosen from a JFileChooser component:

Example:



Excercise 6

- Write a UI, which asks the Markdown file name from the user
- Then it calls the converter code from Day 1
- The output is saved in a „.latex“ file
- If time permits do the latex to pfd translation
- Show the pdf in a viewer.
(This part we have not talked about an you will have work out how an external process can be called (and controlled) from Java) →

Exercise 6

- Run an external program from within your Java code:

```
Runtime run = Runtime.getRuntime();  
Process proc = run.exec(<<Path and Filename>>);
```

- Through the instance of the process class one can get access to stdin, stdout, and stderr of the process.
- The commands, the path to them, the expected input and output and so on are all platform dependent.

Preparation for Day 3

- Install Java 8 EE
 - Download the installer from [Java EE 8.01](#) and run it
 -
 - Set the JETTY_HOME environment variable to the directory, which contains the start.jar file
 - Go into the JETTY_HOME directory within a command prompt window.
 - Run: `java -jar %JETTY_HOME%/start.jar --add-module=server,http,deploy` (confirm the creation of start.d)

Preparation for Day 3

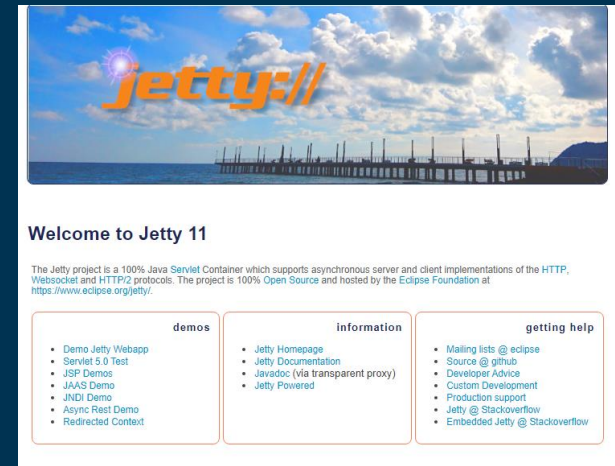
- Install the Web Server Jetty
 - Download the zip file from [Jetty Zip File Download](#). Take version 11.
 - Unpack the zip file some place.
 - Set the JETTY_HOME environment variable to the directory, which contains the start.jar file
 - Go into the JETTY_HOME directory within a command prompt window.
 - Run: `java -jar %JETTY_HOME%/start.jar --add-module=server,http,deploy`
(confirm the creation of start.d)

Preparation for Day 3

- Install the Web Server Jetty (continued)
 - To allow the verification of the Jetty installation go ahead and install the Demo App:
`java -jar %JETTY_HOME%/start.jar --add-module=demo`

- Verify the installation:

- Run Jetty: `java -jar %JETTY_HOME%/start.jar`
 - Use your web browser to go to this URL: [Verification URL](#)



Preparation for Day 3

- Install Axis2 binaries
 - Download the [Axis2 1.8.0 binaries](#)
 - Extract them to a directory
 - Set the Environment Variable AXIS2_HOME to that directory
 - Add %AXIS2_HOME%\bin to the PATH variable
- Download the [Axis2 1.8.0 WAR file](#). Unpack it.
- Move the axis2.war file to the %JETTY_HOME%\webapp directory

Preparation for Day 3

- Install Ant
 - Download the [Ant 1.10.12 binaries](#)
 - Extract them to a directory
 - Set the Environment Variable ANT_HOME to that directory
 - Add %ANT_HOME%\bin to the PATH variable

Preparation for Day 3

- Setup WSL
 - On a command line enter: `wsl --install`
 - This will setup UBUNTU as an additional OS on your machine
 - Reboot might be required
 - You will be asked for a user name and a password (these have nothing to do with your windows user or password)
 - Install windows terminal as described here: [Windows Terminal Install from MS Store](#)
 - For the not so faint at heart folks the WSL install is described [here](#).
 - Ignore the rest of the installation instructions

Preparation for Day 3

- Install Docker Desktop
 - Install Docker Desktop from [here](#).
 - Verify that everything went successful (Step 6) : Test that your installation works correctly by running a simple built-in Docker image using:

```
docker run hello-world
```

It should not matter if you run the command from a windows command prompt or from the ubuntu command prompt.

Preparation for Day 3

- Stop the installed programs
 - Exit the Docker Desktop program by right-clicking on the icon in the task bar and choose „Exit“
 - Stop Ubuntu by entering `wsl --shutdown` at the windows command prompt

Networking

- We will look into 3 technologies:
 - 1- Sockets (the basic, underlying technology)
 - 2- Restful Web Services (what the WEB is based on)
 - 3- Message Bus (Easiest way to have Applications co-operate)

Networking - Sockets

- Sockets are described as Communication End Points
- They are a logical construct, there is no physical socket representing them
- Usually an OS offers a little over 65 thousands sockets (Why?)
- The first 1024 (why?) sockets are so called well-known sockets. They support such services as ftp, telnet, ssh, mail, and so on.

Networking - Sockets

- For having two (or more) computers communicate via sockets one needs at least two different applications:
 - 1- A server, which waits for and manages connections
 - 2- A Client, which can connect to the server
- Most servers can handle more than one client connection at once.
- Of course, one can run both Application on a single machine.

Networking - Sockets

- Here is the bare server code (single, one time connection only):

```
private ServerSocket serverSocket;  
private Socket connectionSocket;  
private PrintWriter out;  
private BufferedReader in;  
  
public void start(int port) throws IOException {  
    serverSocket = new ServerSocket(port);  
    connectionSocket = serverSocket.accept();  
    out = new PrintWriter(connectionSocket.getOutputStream(), true);  
    in = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));  
    String str = in.readLine();  
    out.println("Echo: " + str);  
}
```

Networking - Sockets

- Here is the bare client code (single, one time connection only):

```
private Socket clientSocket;  
private PrintWriter out;  
private BufferedReader in;  
  
public void startConnection(String ip, int port) throws IOException {  
    clientSocket = new Socket(ip, port);  
    out = new PrintWriter(clientSocket.getOutputStream(), true);  
    in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
}  
  
public String sendMessage(String msg) throws IOException {  
    out.println(msg);  
    String resp = in.readLine();  
    return resp;  
}
```

Networking - Sockets - Exercise 01

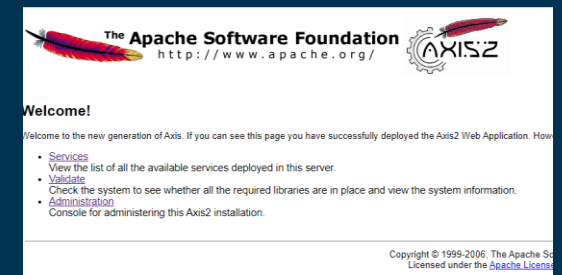
- Create two applications, one server and one client
 - (a) Run it locally on your machine
 - (b) Find another group and let your client talk to their server.
 - (c) Add logic to the server to allow for more than one message to be sent and echoed. Modify the client code accordingly.

Networking - Web Services

- There are two of Web Services:
 - SOAP (used to dominate the Web Service world)
 - REST (is dominating the Web Service world)

Networking - Web Services

- In order to start out with Soap you will install a tomcat/axis2 docker image and run it:
 - Run (on the command line): `docker pull angig/axis2`
 - Start it with the docker desktop app. Set the container port 8080 to your local port 8080 (assuming it is not yet used).
 - In your web browser go to: [Axis2 Welcome Page](#)
 - Admin Console: User: admin Password: axis2



Networking - Web Services

- For the complete Axis Soap Web Service section we will use IntelliJ as an editor, but the compile of the Web Service and the Client reaching out to it will happen through the use of the `ant` command on the command line.
- Update your Git repository to get access to the first Soap project :

Day03__SimpleSoapPojo

- We are going to look into the project and discuss it

Networking - Web Services

- Use ant to construct the web service aar file
- Upload it to the tomcat/axis2 and restart the container
- Check the logs – what happened? Any idea why?

Networking - Web Services

- Next project:

Day03__SimpleSoapClient+Webservice

- Update your git repository
- We gonna discuss the service and the client + the WSDL file
- Compile both, upload the service, restart the container, then run the client

Networking - Web Services

- For the second version of Web services we are only looking into a client and use a freely available Rest Web Service on the internet:

<https://postman-echo.com>

- We are going to use a Java http client included in the JDK

Networking - Web Services

- What is REST?
 - Http based
 - Stateless
 - Rest Request
 - Http verb: GET, POST, PUT, DELETE (most common ones)
 - Header: Accept field (response type), Authorization, ...
 - Path to Resource (e.g.: mysrv.com/Customers/9/request/54)
 - (Optional) Message Body for Data

Networking - Web Services

- The Http-Client uses the builder pattern, meaning that there are multiple methods, which are used in combination to construct the http-Request:

```
HttpRequest request = HttpRequest.newBuilder()  
.uri( new URI("https://postman-echo.com/get") )  
.GET()  
.build();
```

This request is then handled by the HttpClient.

Networking - Web Services

- This is the way to send the request via HttpClient:

```
HttpClient client = HttpClient.newHttpClient();
```

```
HttpResponse<String> response = client.send(request,  
BodyHandlers.ofString());
```

`response.body()` contains the result.

Networking - Web Services

- Create a Java program
 1. Which retrieves the response of the Postman Get web service and prints it out.
 2. Which also calls the Postman IP web service, which returns only the IP Address of the caller in its answer. Print that response as well.
 3. If time permits try use the Postman POST web service. That requires reading up on the HttpClient and the Postman details on the web.

Networking - Message Bus

- Install the docker image for RabbitMQ (the message bus we are using)

Run this command on the command line:

```
docker pull rabbitmq
```

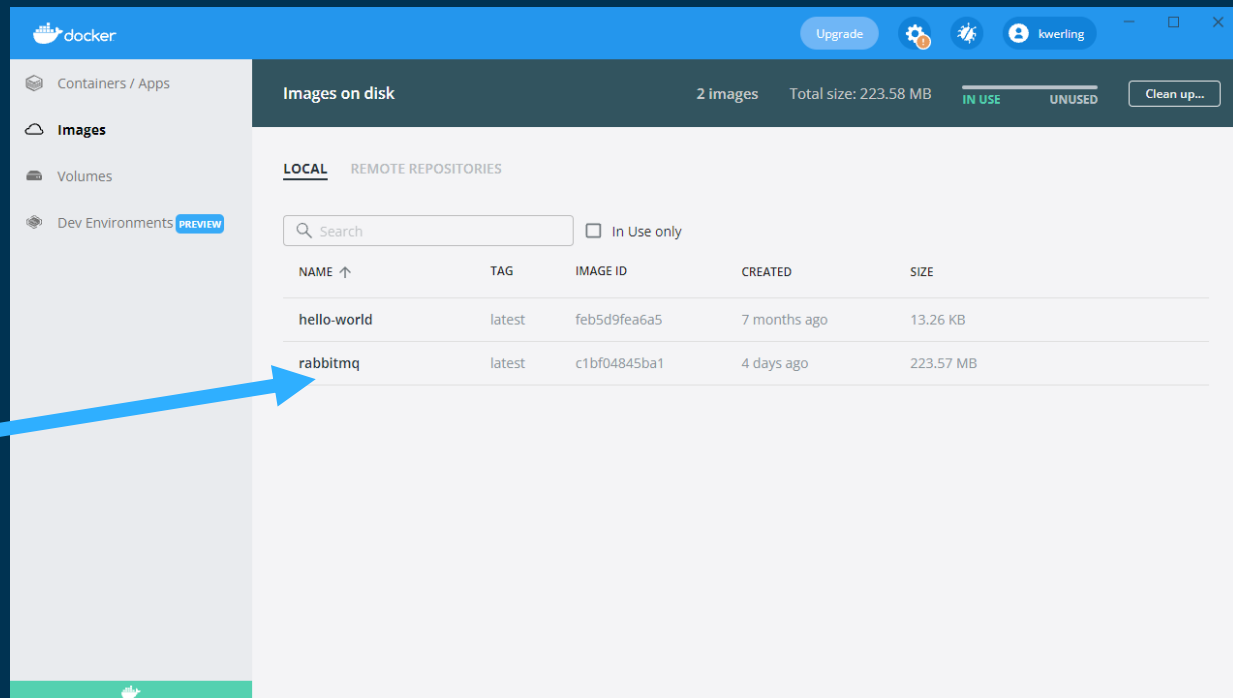
This will take a little while

Networking - Message Bus

- Start the MessageMQ image with Docker Desktop:

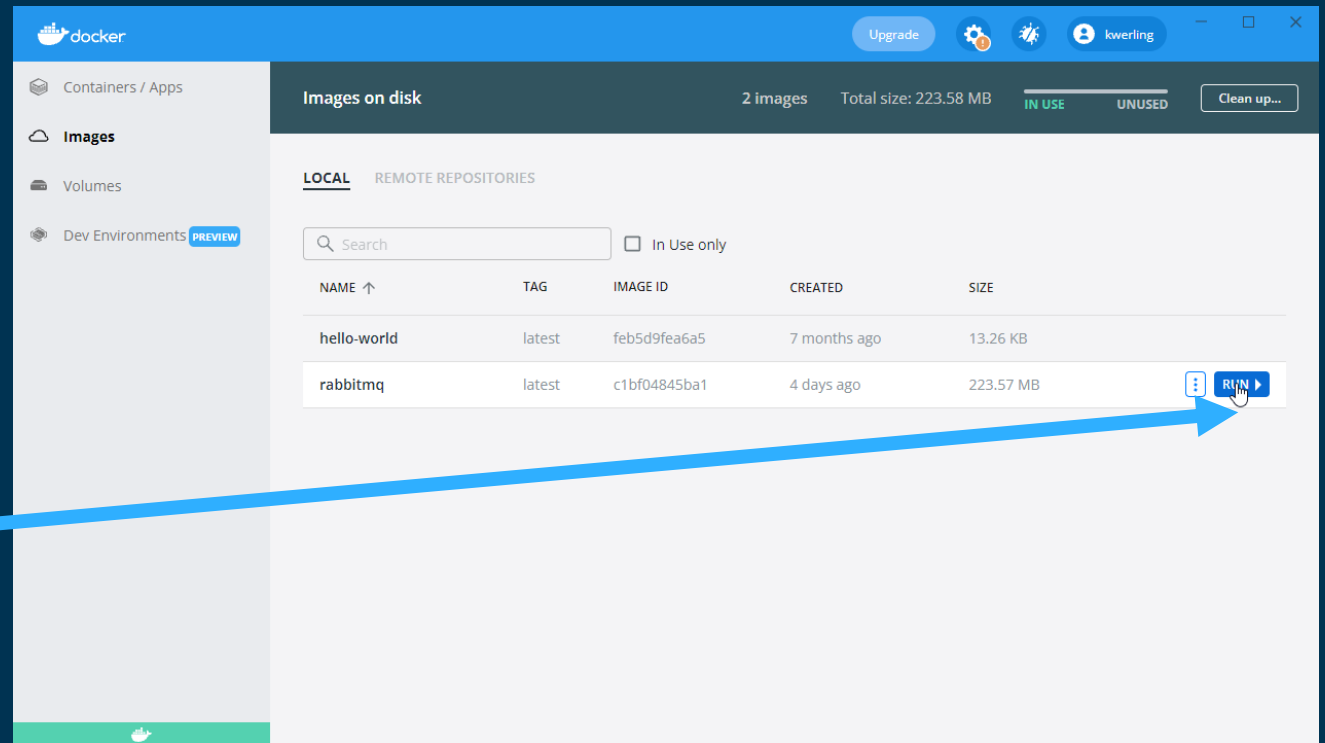
You should see
a window similar
to this one ----->

Rabbitmq is the image
we will start now



Networking - Message Bus

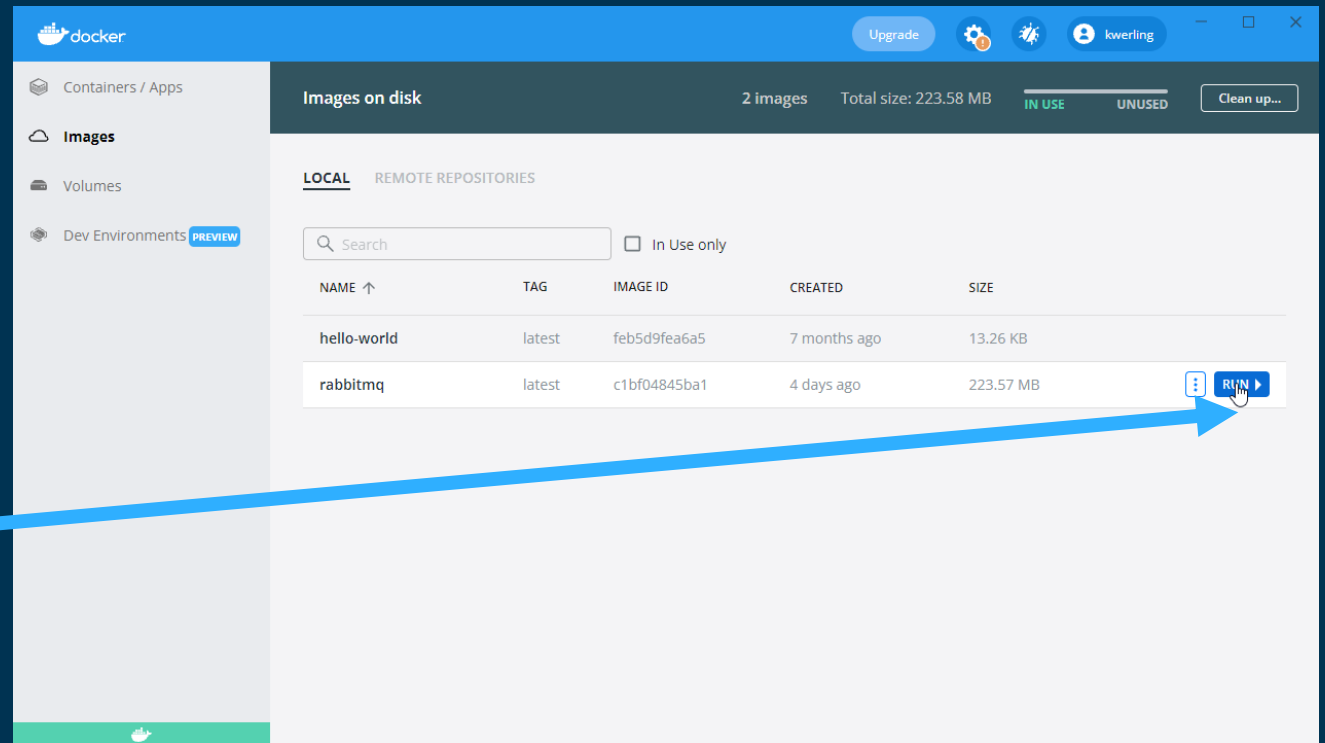
- Start the MessageMQ image with Docker Desktop:



Click on RUN

Networking - Message Bus

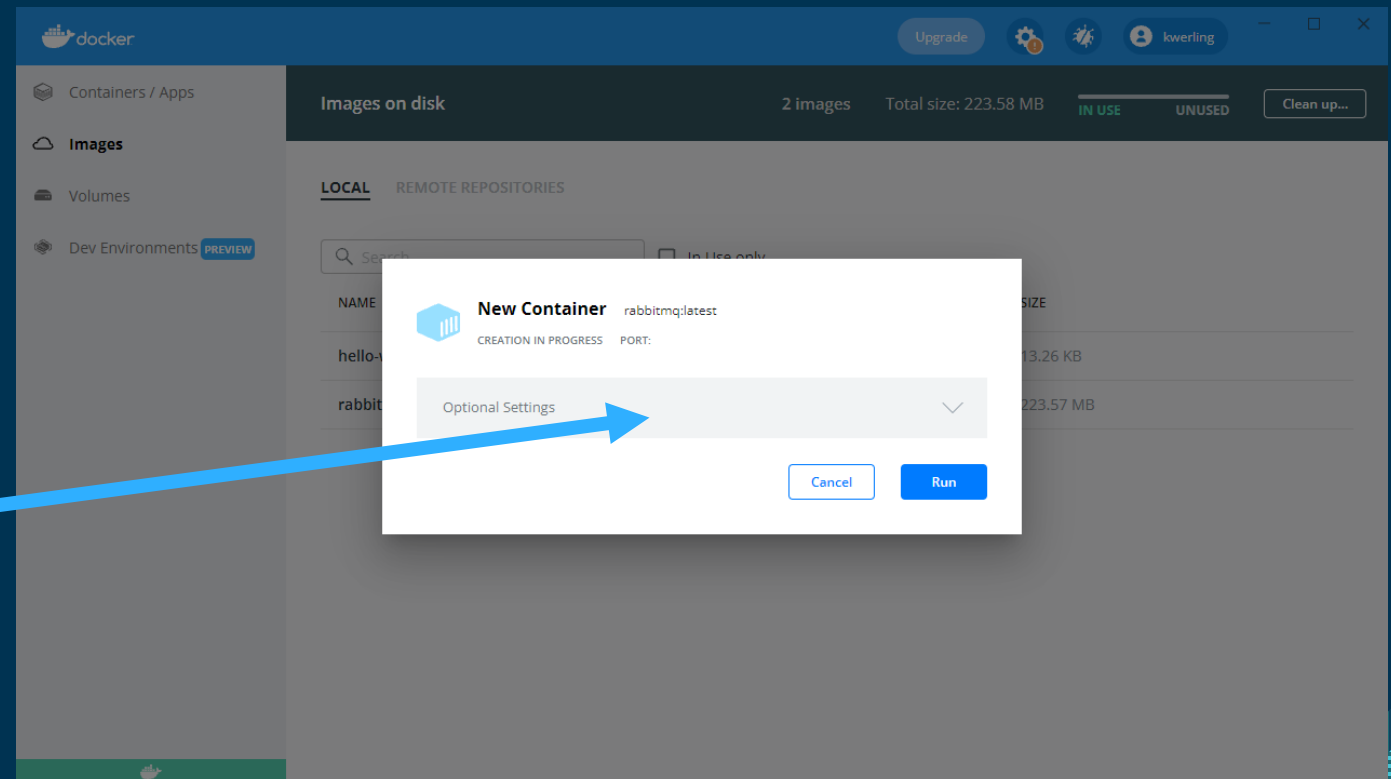
- Start the MessageMQ image with Docker Desktop:



Click on RUN

Networking - Message Bus

- Start the MessageMQ image with Docker Desktop:



Click on
OPTIONAL SETTING

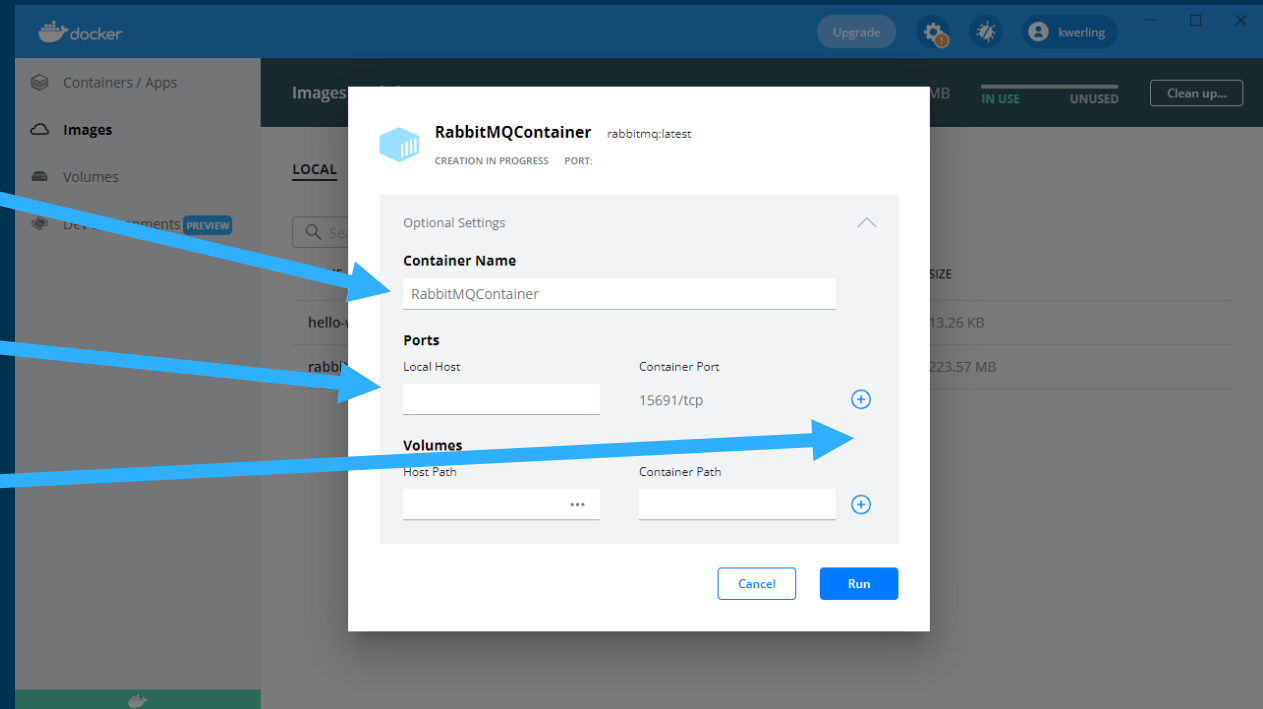
Networking - Message Bus

- Start the MessageMQ image with Docker Desktop:

Give it a name

Fill in the same
LOCAL HOST PORT
as you can see for
the CONTAINER PORT

Click the Plus



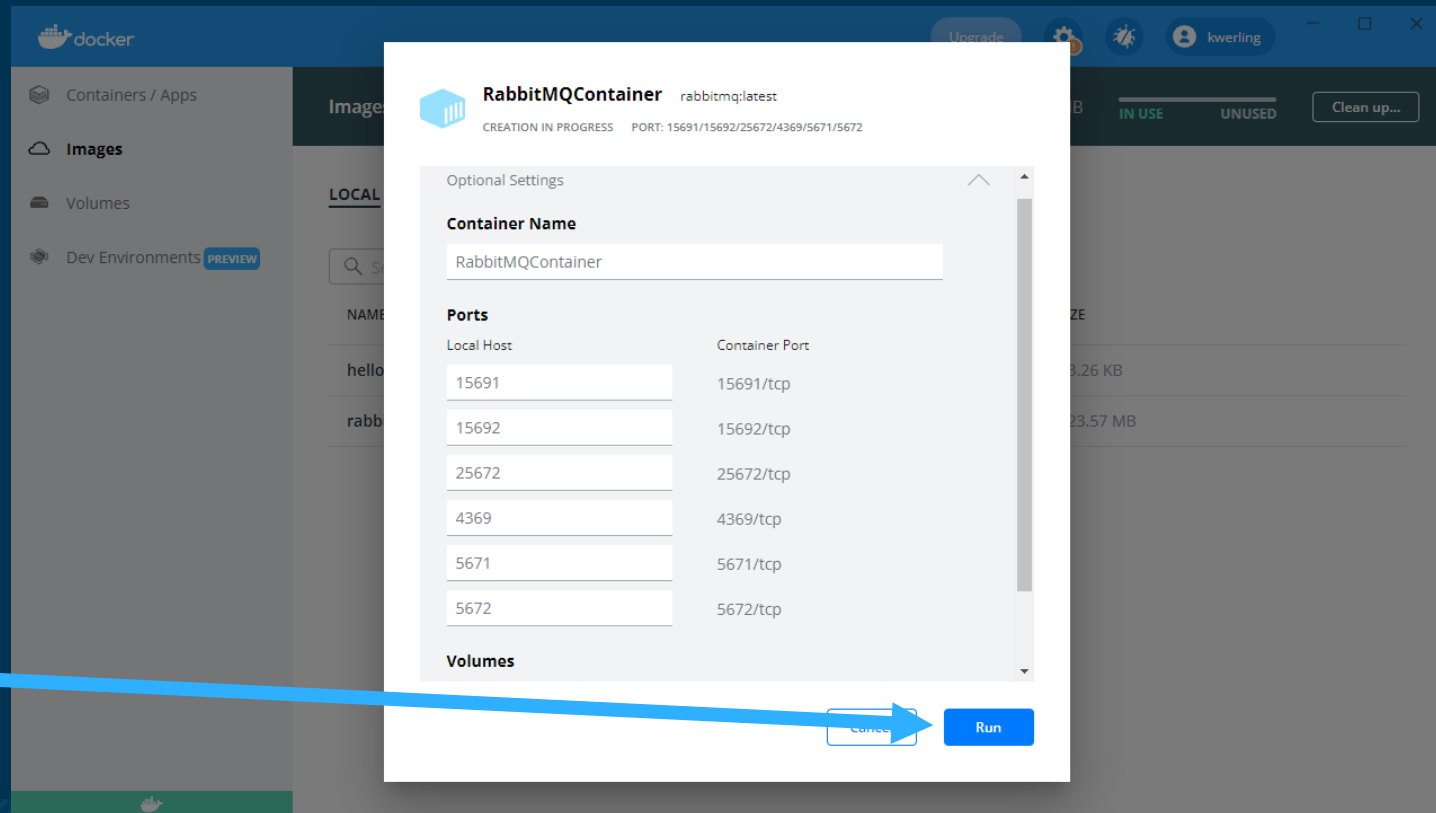
Networking - Message Bus

- Start the MessageMQ image with Docker Desktop:

It should look similar to this:

Leave the **VOLUMES** empty

Click run:

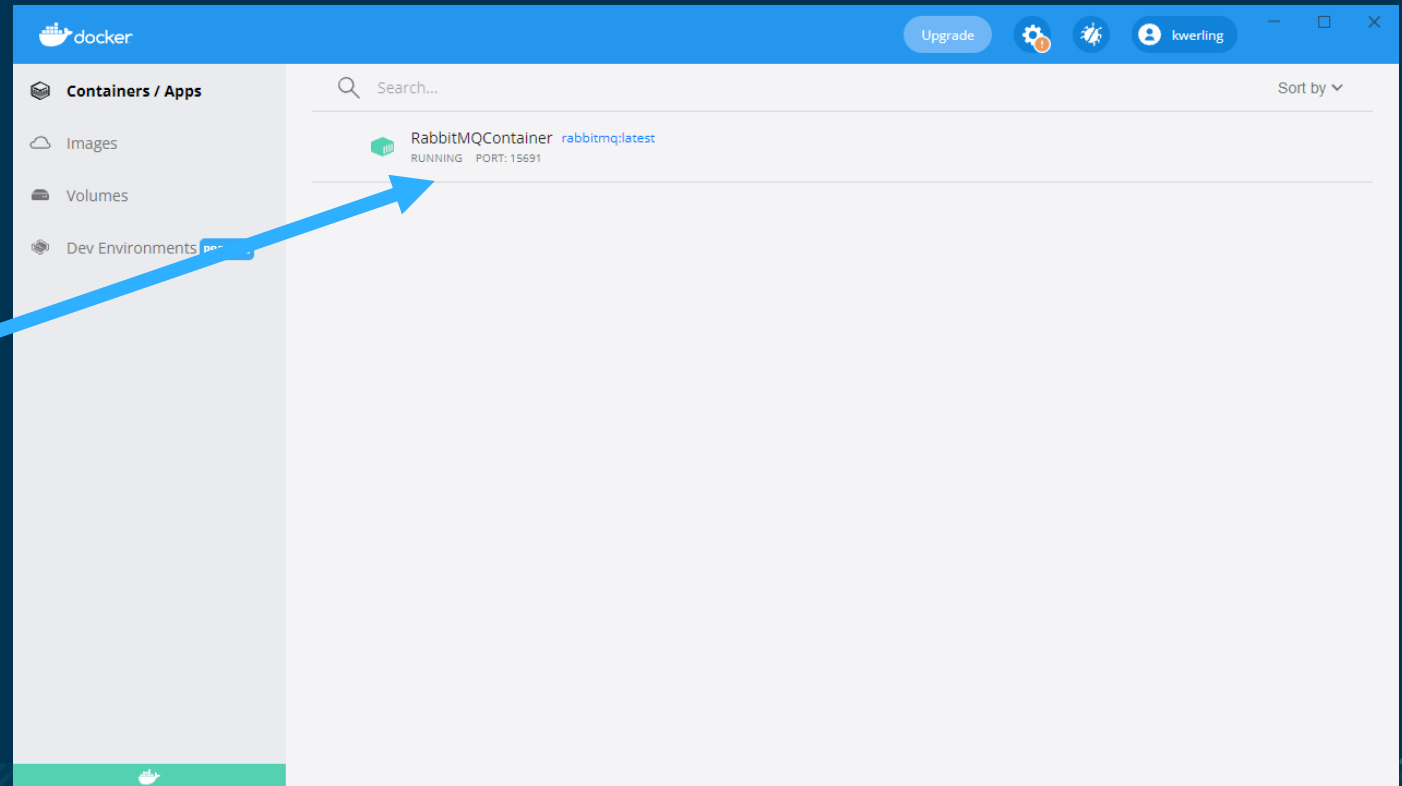


Networking - Message Bus

- Start the MessageMQ image with Docker Desktop:

It should look similar to this:

It says RUNNING

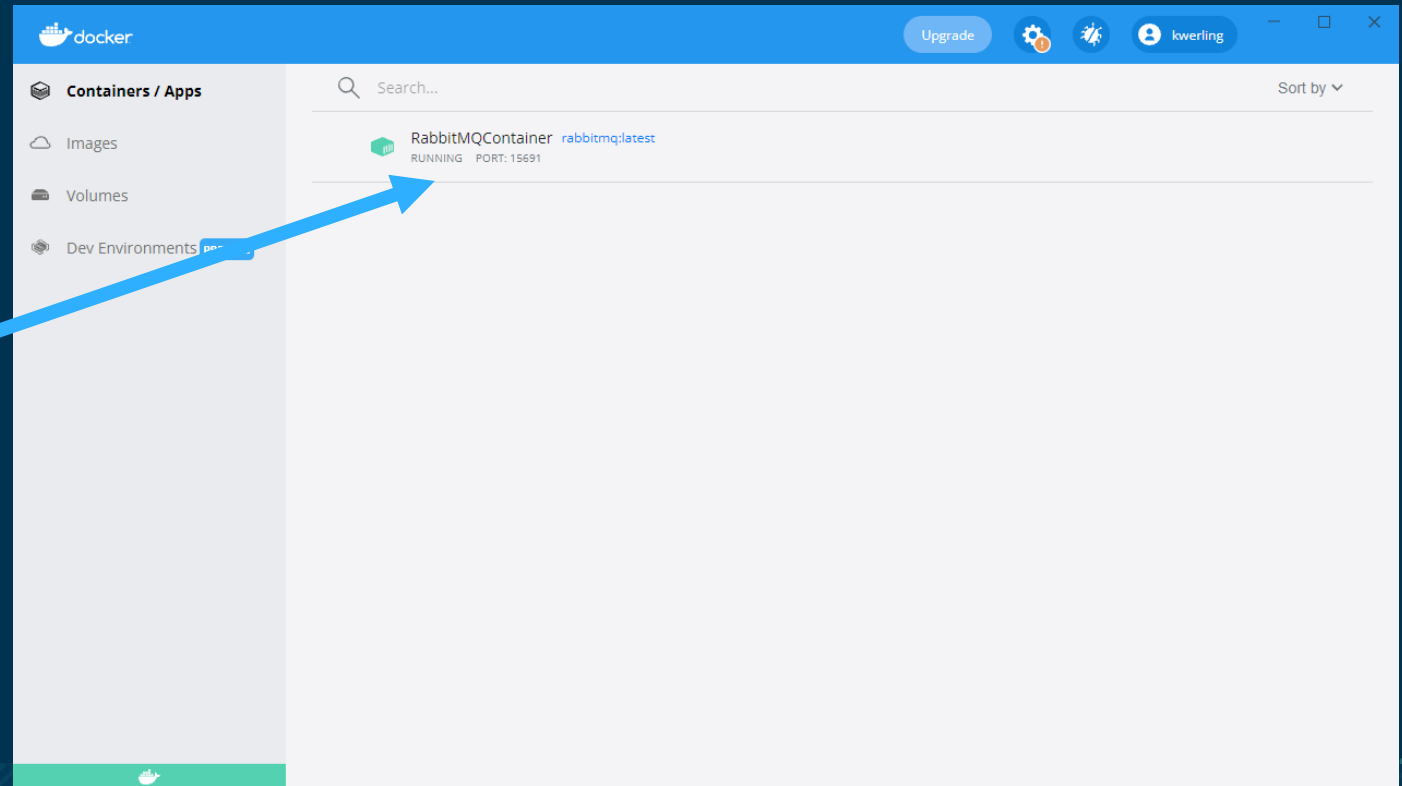


Networking - Message Bus

- Start the MessageMQ image with Docker Desktop:

It should look similar to this:

It says RUNNING



Networking - Message Bus

- What have we done?
- Run UBUNTU as a sub-system in Windows; It is a virtual machine, a separate system in its own right
- We run docker containers on top of Ubuntu – again, separate systems in their own right
- In order to being able to work with the docker images we map the ports of the Docker Container System to the windows system.

Networking - Message Bus

- How to make use of RabbitMQ?
- Here is the Producer & the Consumer code:



Main.java



Main.java

- Make two projects, a producer & a consumer project.

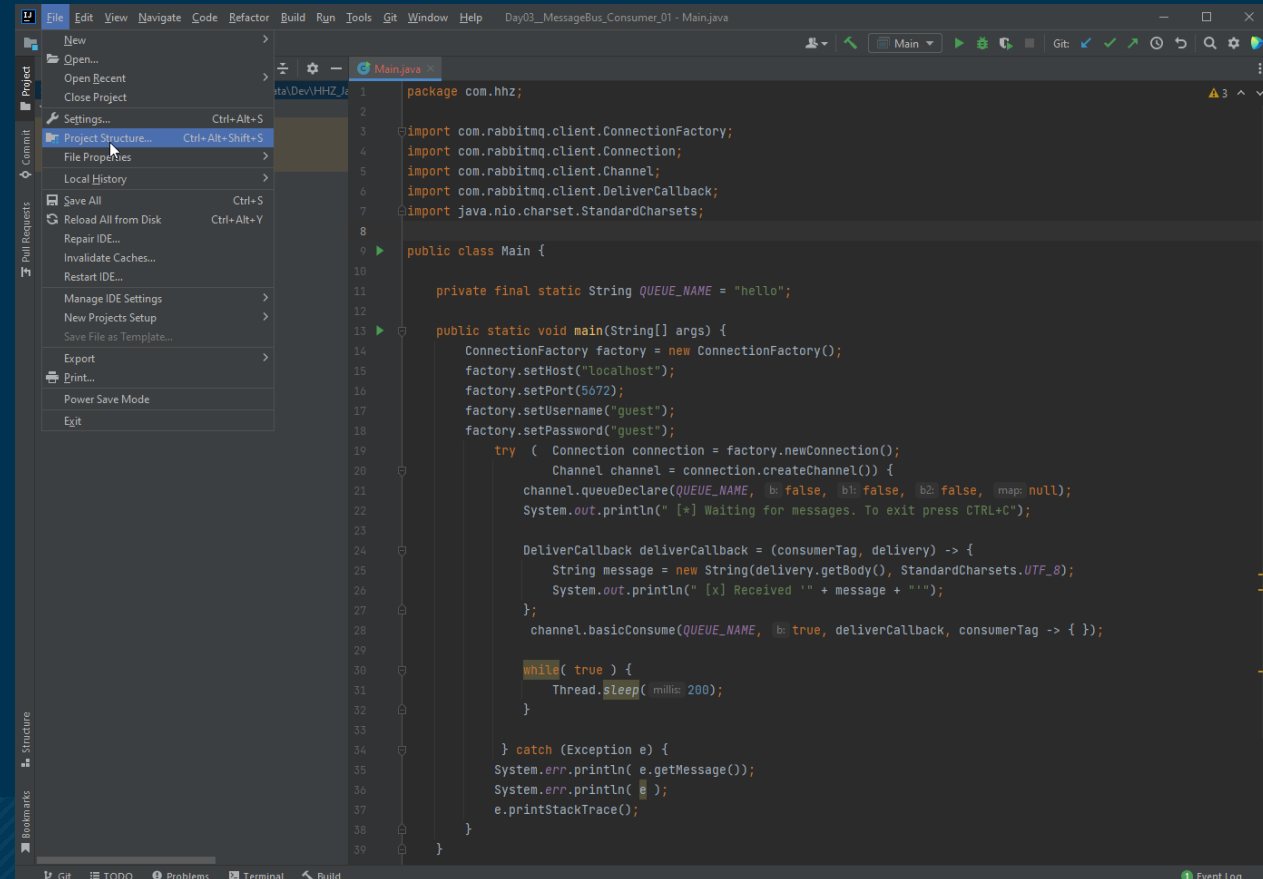
Networking - Message Bus

- In addition, you will need to have access to the following JAR files:
 - Ampqp-client (to work with RabbitMQ)
- Update your Git Lecture exercises directory
- Add the library files to the two projects created (see next slide).
Doing the following steps in the IDE is similar to adding the jar / class files to the classpath.

Networking - Message Bus

- Add library Jar files from a directory to a project:

File --> Project Structure



```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help Day03_MessageBus_Consumer_01 - Main.java
New
Open...
Open Recent
Close Project
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Repair IDE...
Invalidate Caches...
Restart IDE...
Manage IDE Settings
New Projects Setup
Save File as Template...
Export
Print...
Power Save Mode
Exit

package com.hhz;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.DeliverCallback;
import java.nio.charset.StandardCharsets;

public class Main {

    private final static String QUEUE_NAME = "hello";

    public static void main(String[] args) {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("localhost");
        factory.setPort(5672);
        factory.setUsername("guest");
        factory.setPassword("guest");
        try {
            Connection connection = factory.newConnection();
            Channel channel = connection.createChannel();
            channel.queueDeclare(QUEUE_NAME, false, false, false, null);
            System.out.println(" [*] Waiting for messages. To exit press CTRL+C");

            DeliverCallback deliverCallback = (consumerTag, delivery) -> {
                String message = new String(delivery.getBody(), StandardCharsets.UTF_8);
                System.out.println(" [x] Received '" + message + "'");
            };
            channel.basicConsume(QUEUE_NAME, true, deliverCallback, consumerTag -> {});

            while( true ) {
                Thread.sleep( millis: 200);
            }

        } catch (Exception e) {
            System.err.println( e.getMessage());
            System.err.println( e );
            e.printStackTrace();
        }
    }
}
```

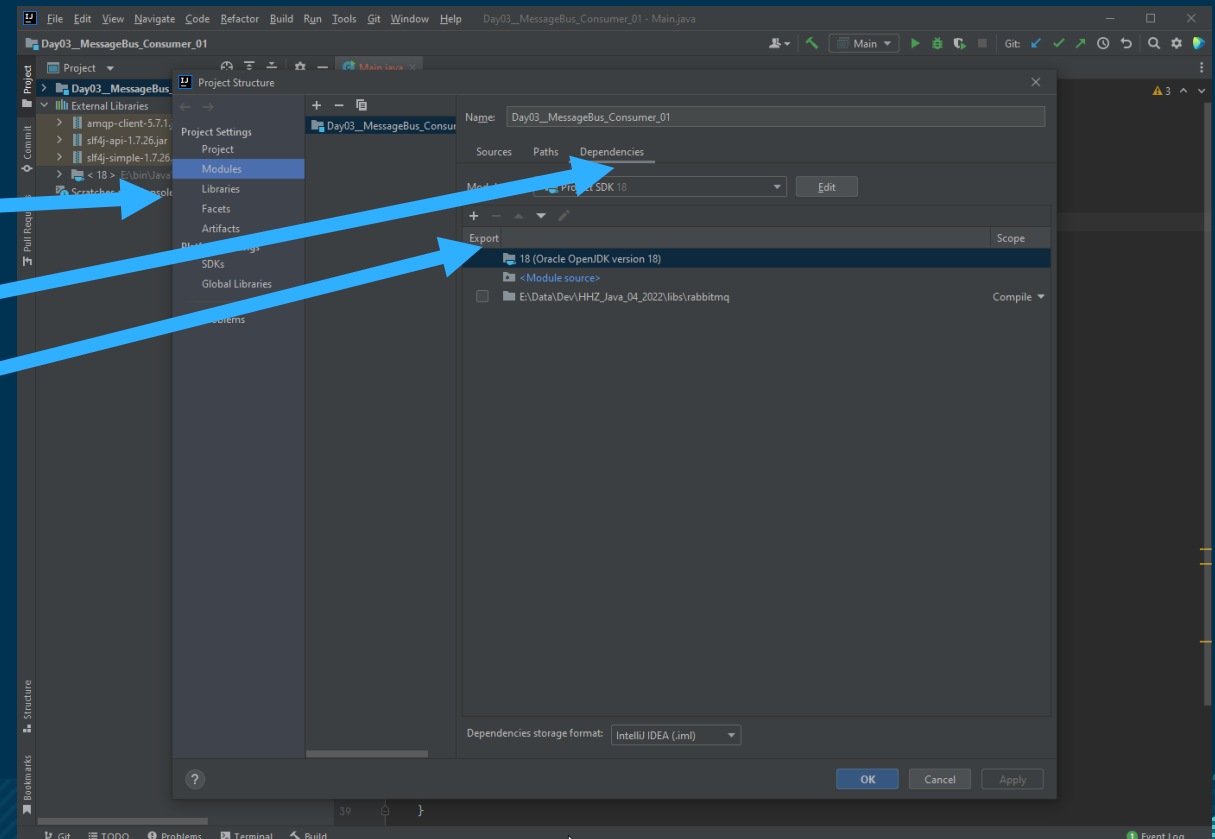
Networking - Message Bus

- Add library Jar files from a directory to a project:

Modules

Dependencies

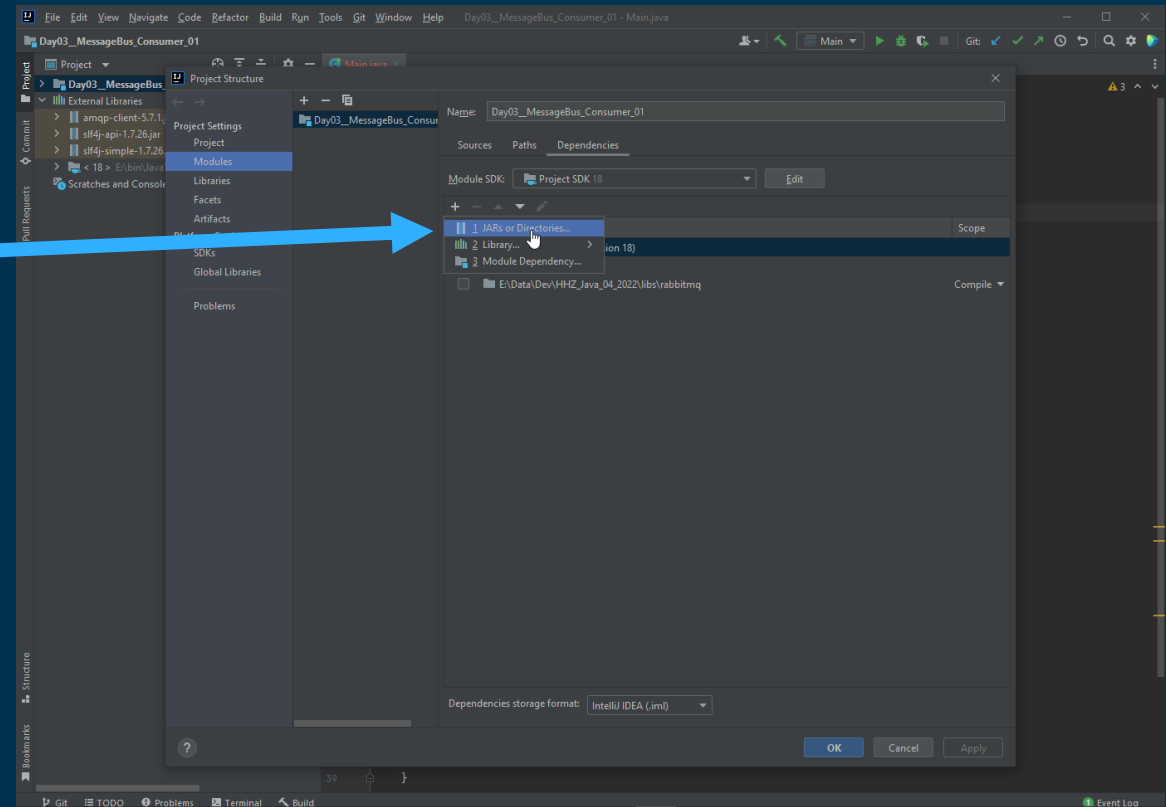
+Icon



Networking - Message Bus

- Add library Jar files from a directory to a project:

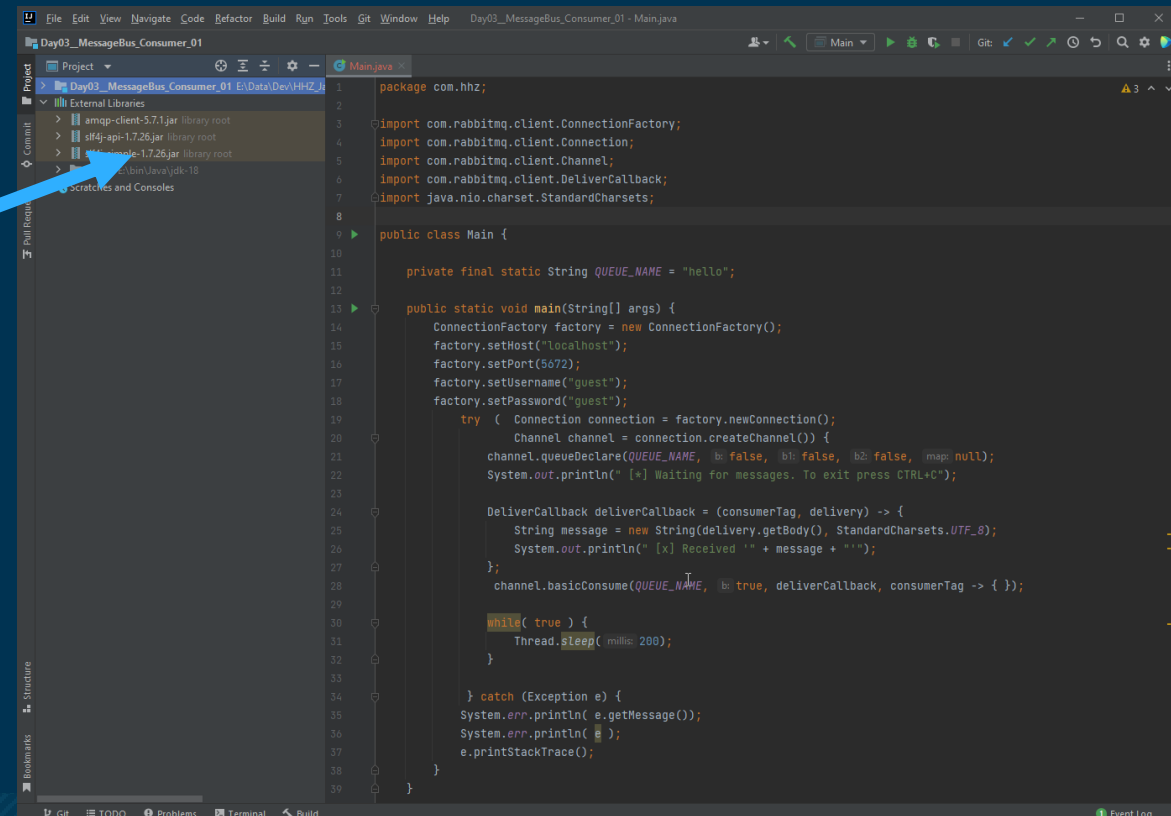
Jars or Directories



Networking - Message Bus

- Add library Jar files from a directory to a project:

They will show up
under External
Libraries



Networking - Message Bus

- Now you are ready to run the projects.
- Whenever you run the Producer the Consumer will show the message sent.
- BTW: I cannot (yet) explain why the `Thread.sleep` endless loop is required to prevent to have the consumer ending.

Databases

- Today, there are two types of Databases in use:
 - Relational Databases, which mostly is a synonym for an SQL base database.
 - Non-SQL databases

Databases -Relational Databases

- Today, there are two types of Databases in use:
 - Relational Databases, which mostly is a synonym for an SQL based databases.
 - No-SQL databases
 - Key-Value Databases
 - Document Databases
 - Graph Databases
 - ...

Databases -Relational Databases

- We start out with a relational DB. There are many freely available relational DBs available. Here are some of them:
 - HSQLDB (the one we are going to use)
 - Derby / Java DB (used to be bundled with the JDK)
 - H2 (derived from HSQLDB)
 - And many, many others.

[Here](#) is a list of relational and non-sql java databases.

Databases -Relational Databases

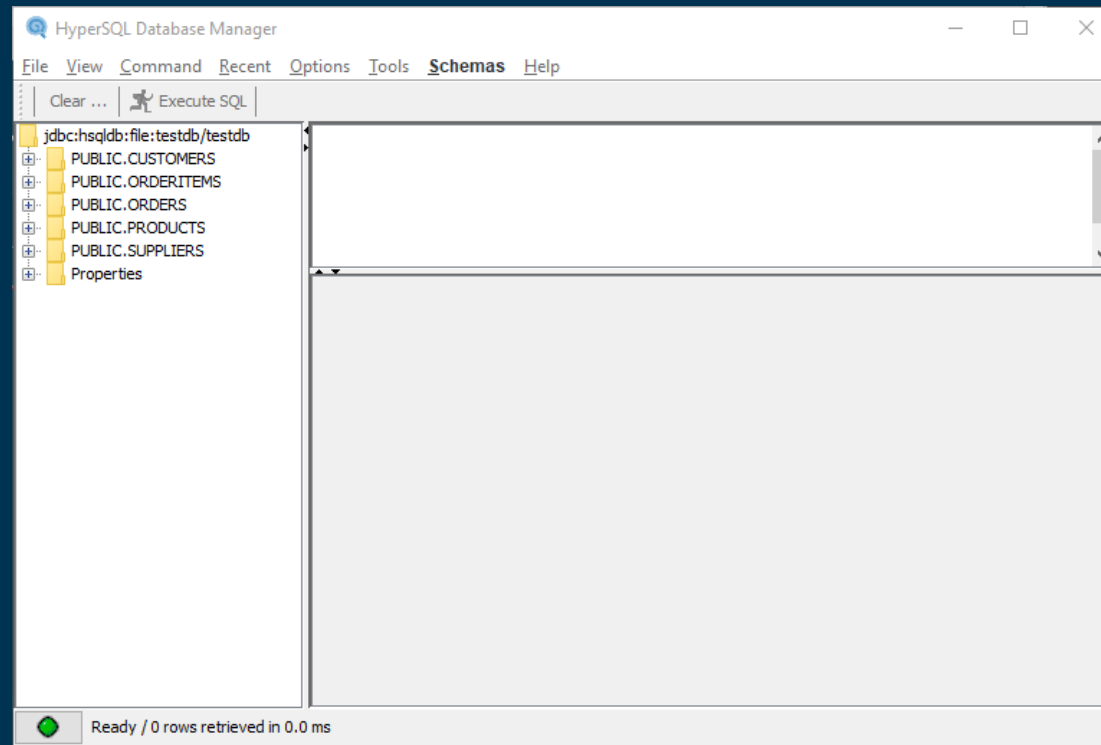
- Preparation:
 - Either Download HSQLDB from [here](#)
or update the repository and use it from the lib directory
 - Load the Day04__CreateRelationalIDBinHSQLDB project into IntelliJ
 - Add the hsqldb.jar file to the project.
 - Compile and run it.

Databases -Relational Databases

- Open the command line
 - Go into the project directory
 - Run this command: `java -jar <location of hsqldb.jar file> --URL jdbc:hsqldb:file:testdb/testdb`
- <location of hsqldb.jar file> ==> Where ever the hsldb.jar is located in your file system.

Databases -Relational Databases

- You were successful when you see this:



Databases -Relational Databases

- ER Diagram of DB used:

| Suppliers | | |
|-------------|---------|----|
| Id | Integer | PK |
| CompanyName | String | |
| ContactName | String | |
| City | String | |
| Country | String | |
| Phone | String | |
| Fax | String | |

| Products | | |
|----------------|---------|----|
| Id | Integer | PK |
| ProductName | String | |
| SupplierId | Integer | FK |
| UnitPrice | Double | |
| Package | String | |
| IsDiscontinued | Bit | |

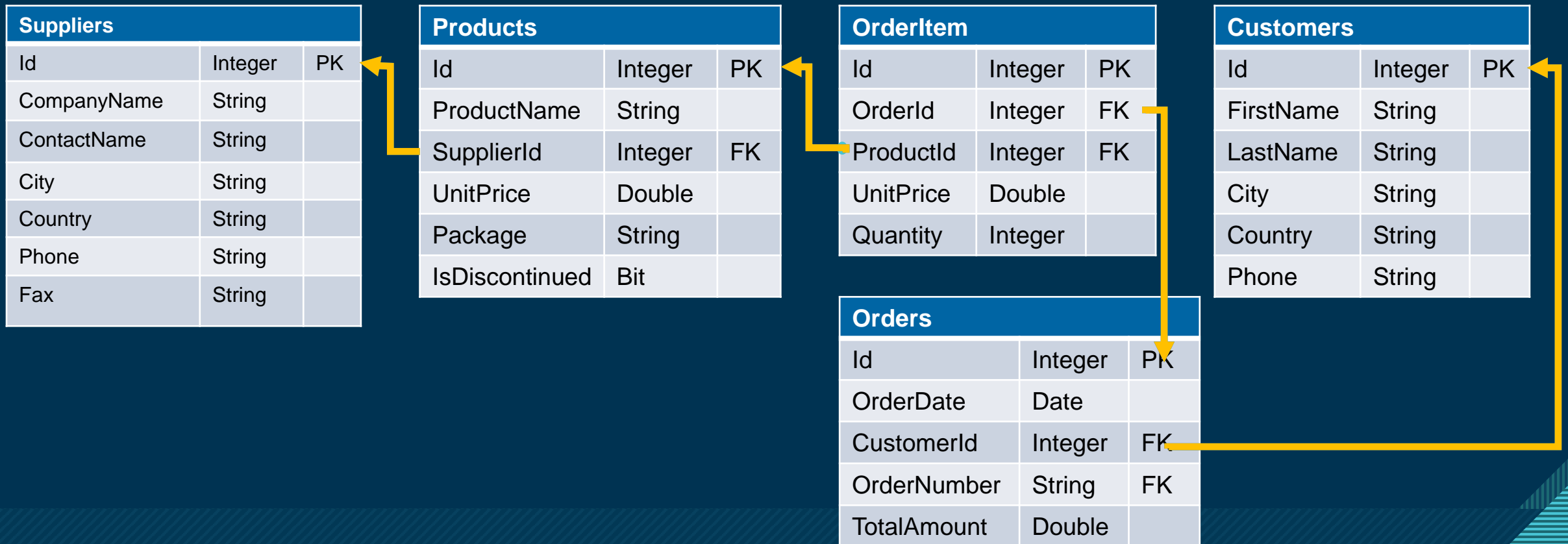
| OrderItem | | |
|-----------|---------|----|
| Id | Integer | PK |
| OrderId | Integer | FK |
| ProductId | Integer | FK |
| UnitPrice | Double | |
| Quantity | Integer | |

| Customers | | |
|-----------|---------|----|
| Id | Integer | PK |
| FirstName | String | |
| LastName | String | |
| City | String | |
| Country | String | |
| Phone | String | |

| Orders | | |
|-------------|---------|----|
| Id | Integer | PK |
| OrderDate | Date | |
| CustomerId | Integer | FK |
| TotalAmount | Double | |

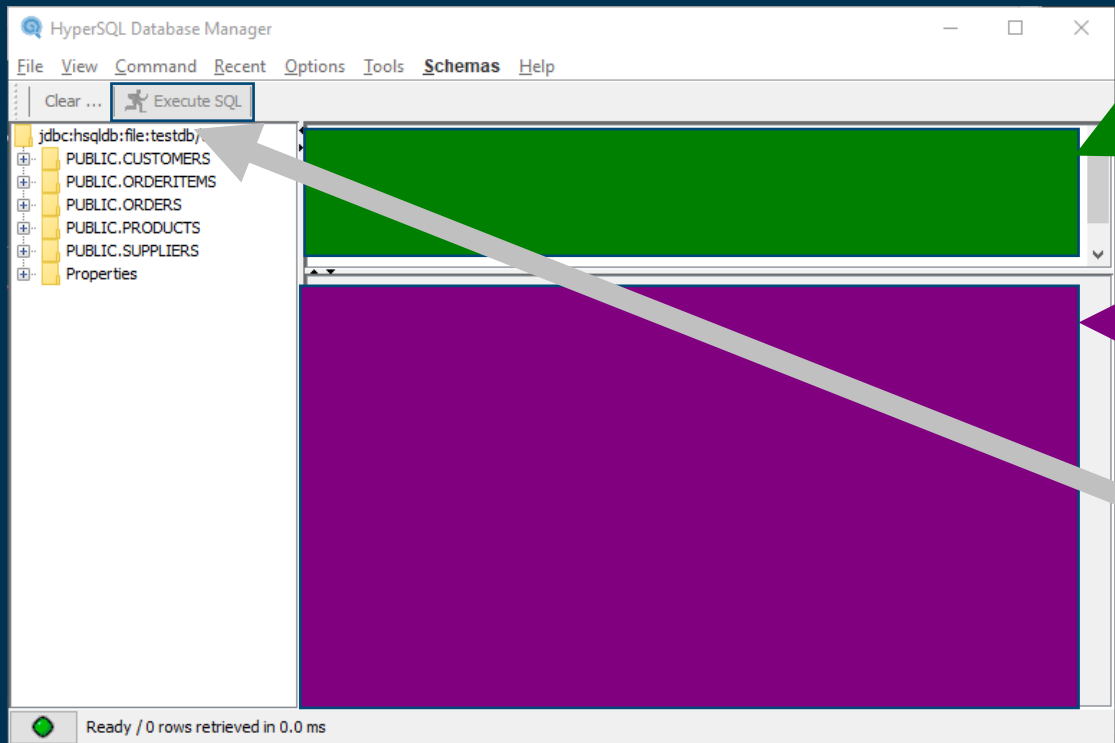
Databases -Relational Databases

- ER Diagram of DB used:



Databases -Relational Databases

- How to use this console:



Enter the SQL command here

See the results of the command here (after execution)

Databases -Relational Databases

- Learning by example: Extracting Data
 - Get all the records in a table:
select * from <table name>;
Ex: select * from suppliers;
 - Get all the records with just some of the columns in a table:
select <column name>,<column name> from <table name>
Ex: select id, companyname from suppliers;

Databases -Relational Databases

- Learning by example: Extracting Data
 - Get only the records fulfilling some criteria
select * from suppliers as s
where s.country = 'USA' and s.fax is not null
 - Get a count of the the resulting records
select count(*) from suppliers as s
where s.country = 'USA'
 - Compare to: select count(fax) from suppliers where country = 'USA'

Databases -Relational Databases

- Learning by example: Extracting Data

- Extracting data from more than one table:
`select * from suppliers as s, orders as o`

What happens here? Can you find a solution to the problem?

Hint:

```
select count(*) from suppliers as s;  
select count(*) from orders as o;  
select count(*) from suppliers as s, orders as o;
```

Databases -Relational Databases

- Learning by example: Extracting Data
 - Using aggregate functions(sum, avg, min, max, count, ...):

```
select p.supplierid, count(*) from products as p  
group by p.supplierid
```

- Getting only the suppliers with more than 3 products:

```
select p.supplierid, count(*) from products as p  
group by p.supplierid  
having count(*) > 3
```

Databases -Relational Databases

- Learning by example: Extracting Data
 - Using aggregate functions(sum, avg, min, max, count, ...):

```
select p.supplierid, count(*) from products as p  
group by p.supplierid
```

- Getting only the suppliers with more than 3 products:

```
select p.supplierid, count(*) from products as p  
group by p.supplierid  
having count(*) > 3
```

Databases -Relational Databases

- Learning by example: Extracting Data
 - Suppliers by name with more than 3 products 1st try:

```
select s.companyname, p.supplierid, count(*)  
from products as p, suppliers as s  
group by p.supplierid  
having count(*) > 3
```

That did not work out. Why?

Databases -Relational Databases

- Learning by example: Extracting Data

- Suppliers by name with more than 3 products 2nd try (corelated subqueries):

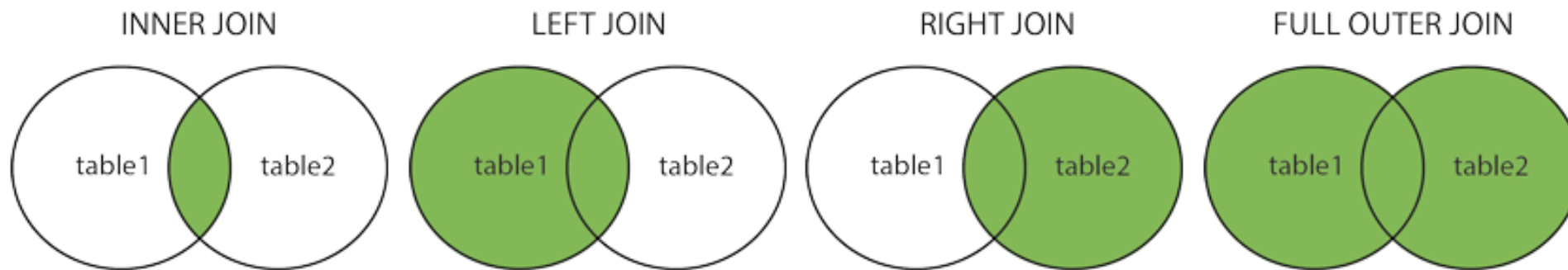
```
select s.companyname, (      select count(*) as c
                             from products as p
                             where p.supplierid = s.id
                             group by p.supplierid
                             as PRODCOUNT
                        )
from suppliers as s
where 3 < (      select count(*) as c
                from products as r
                where r.supplierid = s.id
                group by r.supplierid
            )
```

Databases -Relational Databases

Different Types of SQL JOINS

Here are the different types of the JOINS in SQL:

- **(INNER) JOIN** : Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN** : Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN** : Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN** : Returns all records when there is a match in either left or right table



Databases -Relational Databases

- Examples for different kind of joins:
 - `select * from Suppliers as s inner join Products as p on s.id = p.supplierid`
 - `select * from Suppliers as s left join Products as p on s.id = p.supplierid`
 - `select * from Suppliers as s right join Products as p on s.id = p.supplierid`
 - `select * from Suppliers as s full join Products as p on s.id = p.supplierid`

Databases -Relational Databases

- How to run SQL commands in Java - JDBC:
 - Preparation:
 - `final String DB_URL = "jdbc:hsqldb:mem:memdb";`
 - `final String USER = „SA“;`
 - `final String PASS = "";`
 - `final String QUERY = "SELECT * FROM Products where ProductName like 'C%';";`
 - `final String INSERT = "INSERT INTO Products (Id, ProductName, SupplierId, UnitPrice, Package, IsDiscontinued) VALUES(2002, 'Cup, model 32a67', 14, 4.34, '12 per box', true)";`

Databases -Relational Databases

- How to run SQL commands in Java - JDBC:

- Opening of the database

- `try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {`

`<<<< DB code here >>>>`

```
} catch ( Exception ex ) {  
    System.err.println("ERROR: " + ex.getMessage());  
}
```

Databases -Relational Databases

- How to run SQL commands in Java - JDBC:
 - try(Statement stmt = conn.createStatement();
 ResultSet rs = stmt.executeQuery(QUERY);
) {
 while(rs.next()){
 System.out.println("Name: " + rs.getString("productname"));
 System.out.println("Price: " + rs.getDouble("unitprice"));
 }

 stmt.execute(INSERT); // New SQL command – no resultset
 }

Databases -Relational Databases

- Using PreparedStatement:

| Numbers | | |
|---------|---------|----|
| Id | Integer | PK |
| Number | Integer | |

- `String stm = "INSERT INTO Numbers (Id, Number)VALUES(?,?)";`

```
try( PreparedStatement preparedStatement = conn.prepareStatement(stm) ) {  
  
    for (int i = 0; i < 788; i++) {  
        preparedStatement.setInt(1, i);  
        preparedStatement.setInt(2, 4 * i);  
  
        preparedStatement.addBatch();  
    }  
  
    preparedStatement.executeBatch();  
}
```

Databases -Relational Databases

- Using PreparedStatement:

- String stm = „Select * from Products where SupplierId = ?;“;

Here also the question mark is used with `setInt(<intValue>`

- Compare to:

```
void select_Supplier_by_name( String Name ) {  
    ...  
    Stmt.executeQuery( “Select * from Suppliers as s  
                       where s.CompanyName = “ + Name + “;“);  
    ...  
}
```

Databases -Relational Databases

- Using PreparedStatement:

- What if it is called like this:

```
void ( String Name ) {
```

```
...
```

```
String str = " a"; Delete from Suppliers where 1=1 or CompanyName='a';  
select_Supplier_by_name( str );
```

```
...
```

Databases -Relational Databases

- Transactions:
 - A set of SQL statements, which need to be successfully executed onto a consistent DB or not executed at all with no consequences for the Data in the DB
 - JDBC drivers are often set to AUTO-COMMIT
 - Transactions are ended by : `conn.commit();`
 - Usually, transactions are started implicitly.

Databases -Relational Databases

- Exercises - Write programs, which do the following:
 - List all the products a supplier supplies. The supplierId is passed into the program via the command line.
 - List the supplier names and the number of products they provide
 - Verify that the TotalAmount in Orders is correct according to the orderItems entries

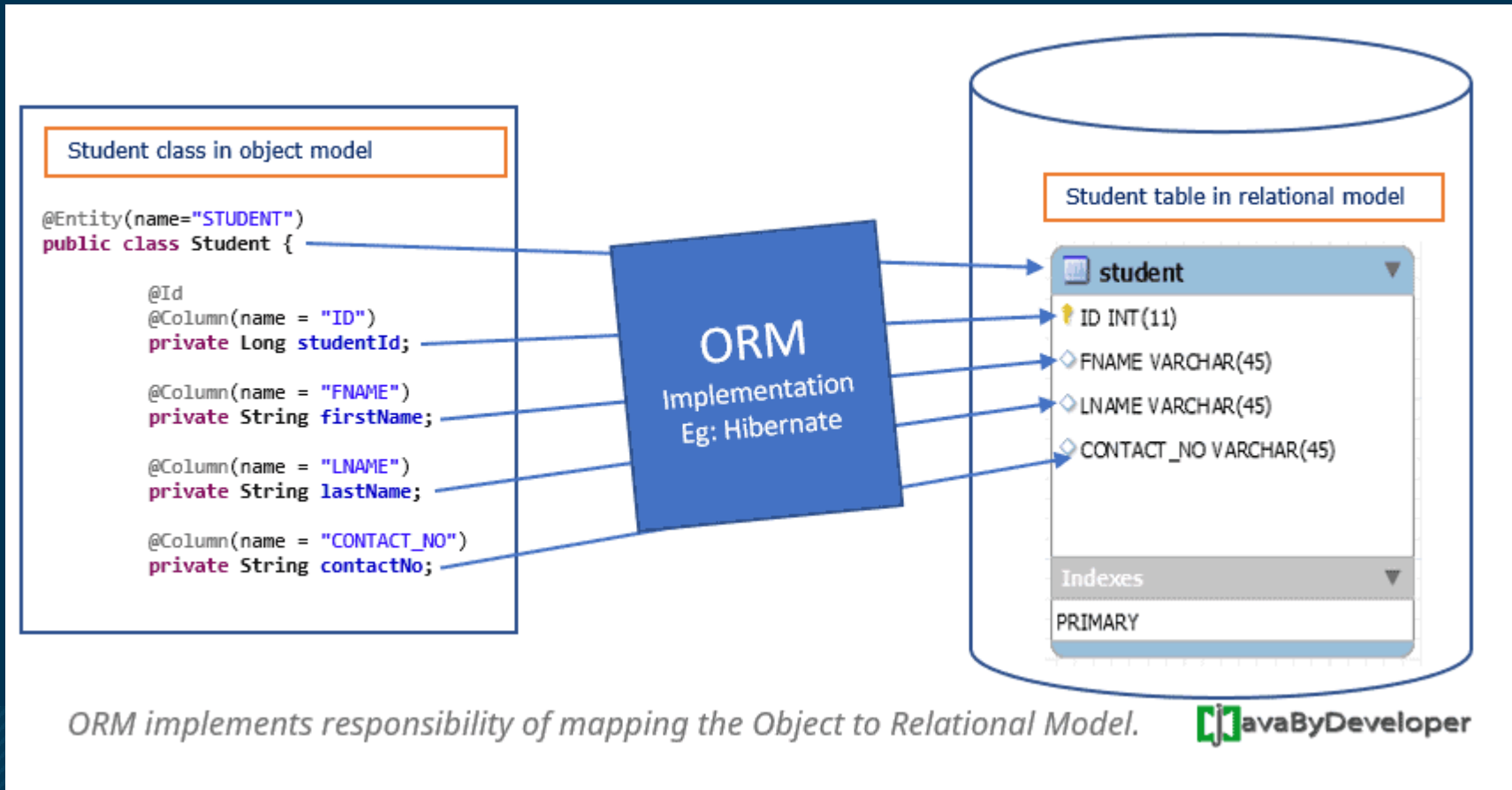
(Discuss the impact of the field TotalAmount in Orders)

Databases -Relational Databases

- ORM -- Object Relational Mapping
 - There are plenty of them!
 - ORMs use JDBC under the hood.
 - They abstract the DB away as good as possible.
 - The goal is to only work with the Instances of classes in Java and not worry about the attribute mapping either from the program into the DB or the other way around.

Databases -Relational Databases

- ORM -- Object Relational Mapping



Databases -Relational Databases

- ORM -- Object Relational Mapping
 - em being the entity manager for the Students table (simplified):
 - `Student st = new Student(33, "Ralf", "Schuster", "0160-2381-1");`
`em.persist(st);`
 - `Student st = em.find(Student.class, 33);`

Databases -Relational Databases

- ORM -- Object Relational Mapping
 - em being the entity manager for the Students table and cb being the criteria builder (simplified pseudo code):
 - `Query q = cb.createQuery(Student.class);`
`q.where(cb.greaterThan "studentID", 423);`
`List<Student> lSt = q.execute();`

Databases - No-Sql: Key-Value DB

- One type of No-Sql databases are key-value databses.
- There is a plethora of different implementations of it available.
- I provide you with a Memory based version, called MemKV.

Databases - No-Sql: Key-Value DB

- When to use a KV DB?
 - Cache for applications (Logged-in users accessing a web site)
 - When very high performance is required in accessing a comparable small set of data

Databases - No-Sql: Key-Value DB

- MemKV methods:

- Add data to MemKV:

| | | |
|--------------------------|--|-------------------|
| Generic Object add: | <code>put(key, object)</code> | |
| Java Class instance add: | <code>putString (key, stringObject)</code> | |
| | <code>putInteger(key, integerObject)</code> | <code>....</code> |

- Retrieve data from MemKV:

| | | |
|--------------------------------|--------------------------------|--------------------|
| Generic Object retrieval: | <code>get(key);</code> | |
| Java Class instance retrieval: | <code>getString (key)</code> | |
| | <code>getDouble(key)</code> | <code>.....</code> |

Databases - No-Sql: Key-Value DB

- MemKV methods:
 - Save data in memory to disk:
`persist(<< fileName >>)`
 - Load a MemKV data file into memory:
`load(<< fileName >>)`

Databases - No-Sql: Key-Value DB

- MemKV methods:

- Number of entries in DB:
- Is DB empty?
- Remove all data from DB
- Delete an entry
- Does key exist?
- Get copy of DB
- List of all keys
- Add data from other MemKV DB
- Loop through data

size()
isEmpty()
clear()
delete(key) or remove(key)
containsKey(key)
getCopyOfDB()
getListOfKeys()
addOtherDb(db)
forEach(function)

Databases - No-Sql: Key-Value DB

- MemKV remarks:
 - The put method either creates a new entry or overwrites an existing one
 - If the program aborts before the data was persisted to the disk then all the data is lost.

Databases - No-Sql: Key-Value DB

- MemKV exercises:
 - Write a program, which gets its data from the command line in the form of <user name> < bonus points >
 - The program can be run multiple times, one after the the other with different data provided
 - Make sure that multiple bonus points can be stored per user.
 - Every time the program ends running it prints out the data like so:

 << user name >> : << sum of all points of that user >> (<< list of indivually achieved points >>)

Databases - No-Sql: Key-Value DB

- MemKV exercises:

- Example:

- Run 1: Peter 32 Andrea 8 Thorsten 12
 - Run 2: Mark 7
 - Run 3: Thorsten 3 Andrea 54

- Output at the end of Run 3: Order of names does not matter.

```
Peter :      32 ( 32 )
Andrea:      62 (  8, 54 )
Thorsten:    15 ( 12, 3 )
Mark:        7  ( 7 )
```

Databases - No-Sql: Key-Value DB

- MemKV has one feature not all of the KV DBs have:

With the methods:

```
MemKV findEntriesByKey(String searchPatter )
```

```
MemKV findEntriesByKey(String searchPatter, boolean caseInsensitiveSearch )
```

one can search the keys with Dos Wildscards (* and ?). It returns another MemKV with the matching keys and their values.

Example: `findEntriesByKey("Andre* Schmidt"`)

Databases - No-Sql: Key-Value DB

- MemKV exercises:
 - Write a program, which contains the following keys (Values do not matter for this exercise):
 - [Customer1][Order17][Item39]
 - [Customer1][Order17][Item71]
 - [Customer1][Order24][Item165]
 - [Customer2][Order25][Item134]
 - [Customer2][Order25][Item181]
 - [Customer4][Order67][Item232]
 - [Customer9][Order94][Item145]

Databases - No-Sql: Key-Value DB

- MemKV exercises:
 - Query for the following criteria:
 1. All orders for Customer1
 2. The Customer behind Order25
 3. All Orders, which contain items with a number in the range between 100 to 199
 - Print the results.