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 execise 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 execises
- 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	*This text is italicized*	- Unordered List Item 2
#### Heading level 4	~~Strikethrough text~~	- Unordered List Item 3
##### Heading level 5	***Bold and italics text***	1. Ordered List Item 1
###### Heading level 6	**Bold and *nesting italics* text**	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 \textbf{bold text}
Italic	This is \textit{text in italic}
Strikethrough	
Unordered List	\begin{itemize} \item Item 1 \item ... \end{itemize}
Ordered List	\begin{enumerate} \item Item 1 \item ... \end{enumerate}
Link	Use the hyperref package

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

Latex

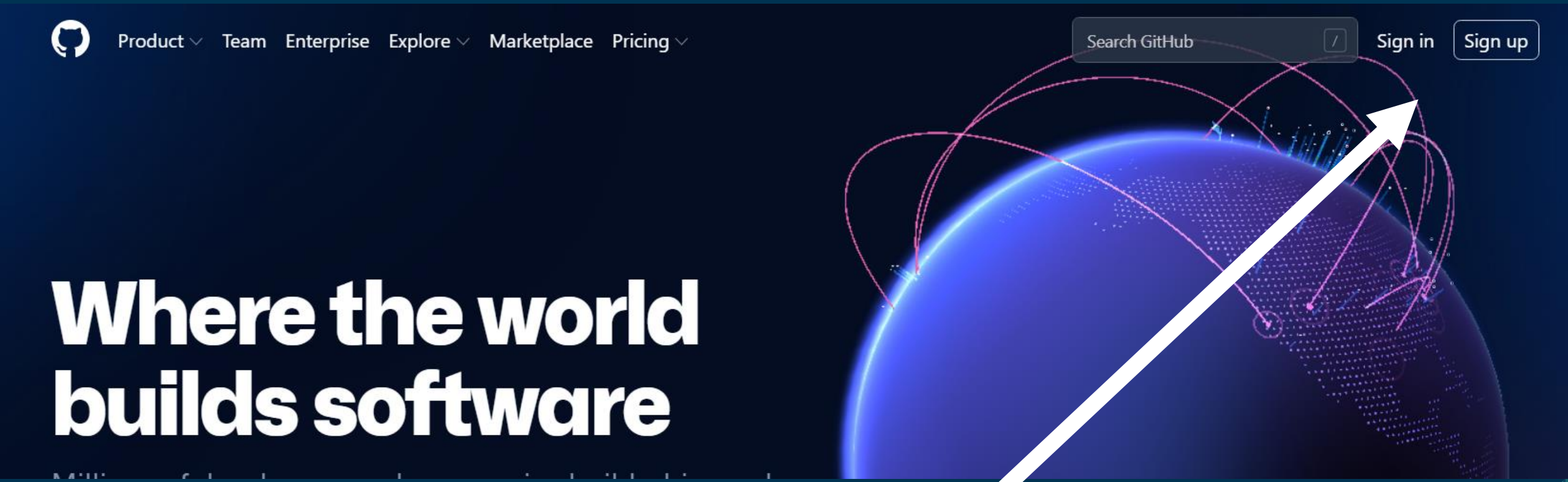
Heading 1	**\section{section}**
Heading 2	**\subsection{subsection}**
Heading 3	**\subsubsection{subsubsection}**
Heading 4	**\paragraph{paragraph}**
Heading 5	**\subparagraph{subparagraph}**

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

What we need to get started

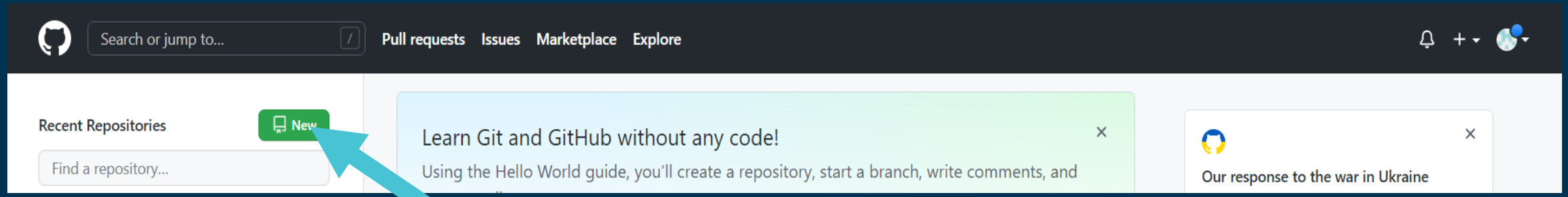
- IntelliJ Community Edition (Download here)
- MiKTex (or any other Tex that can process Latex) (Download here)
- OpenJDK Java 18 (Download here)
- Markdown Viewer (For example: Windows Markdown Viewer)
- Git for Windows (Download here)

Github.Com Account



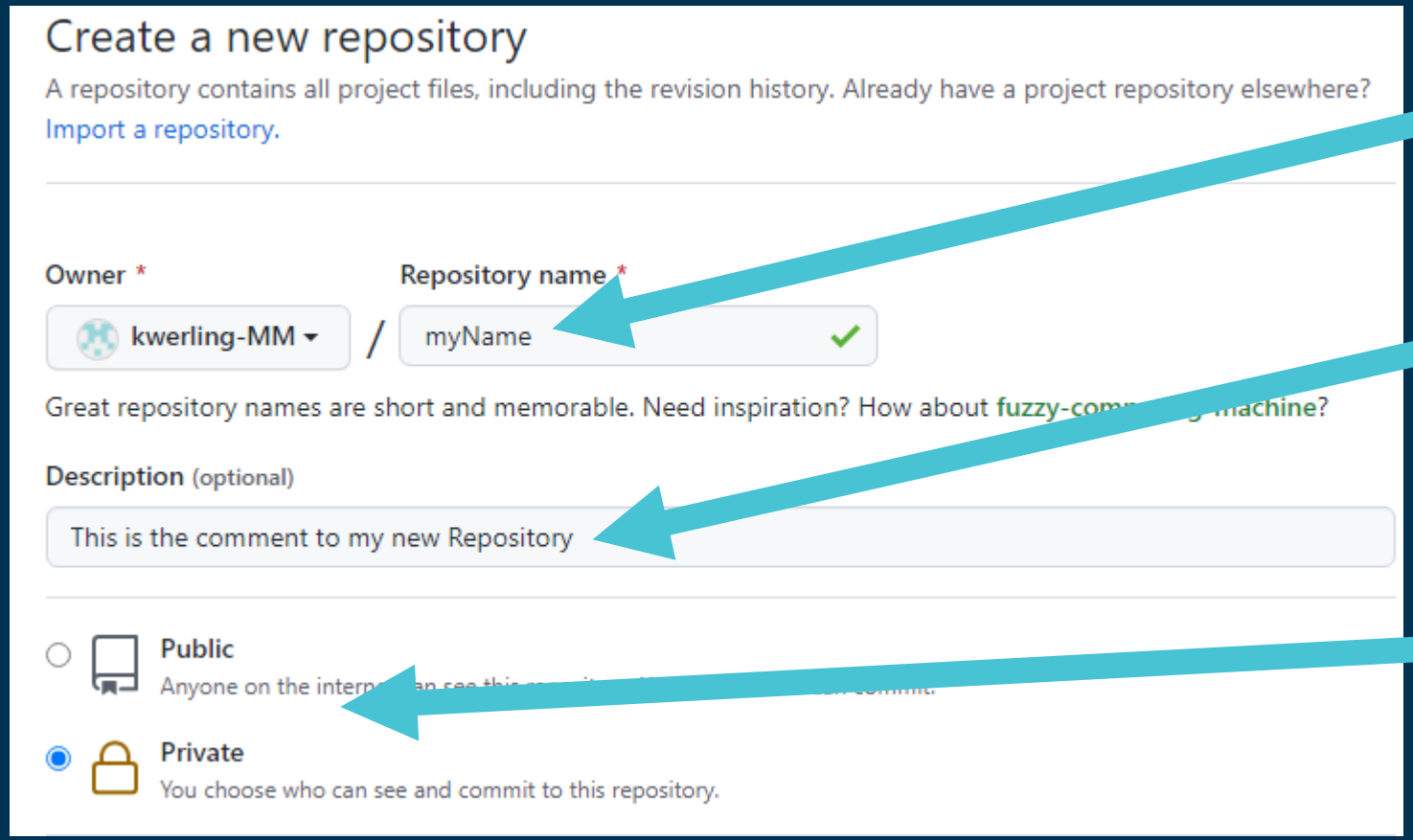
Sign in or Sign up

Create new Repository



New Repository

Create new Repository 1 of 2

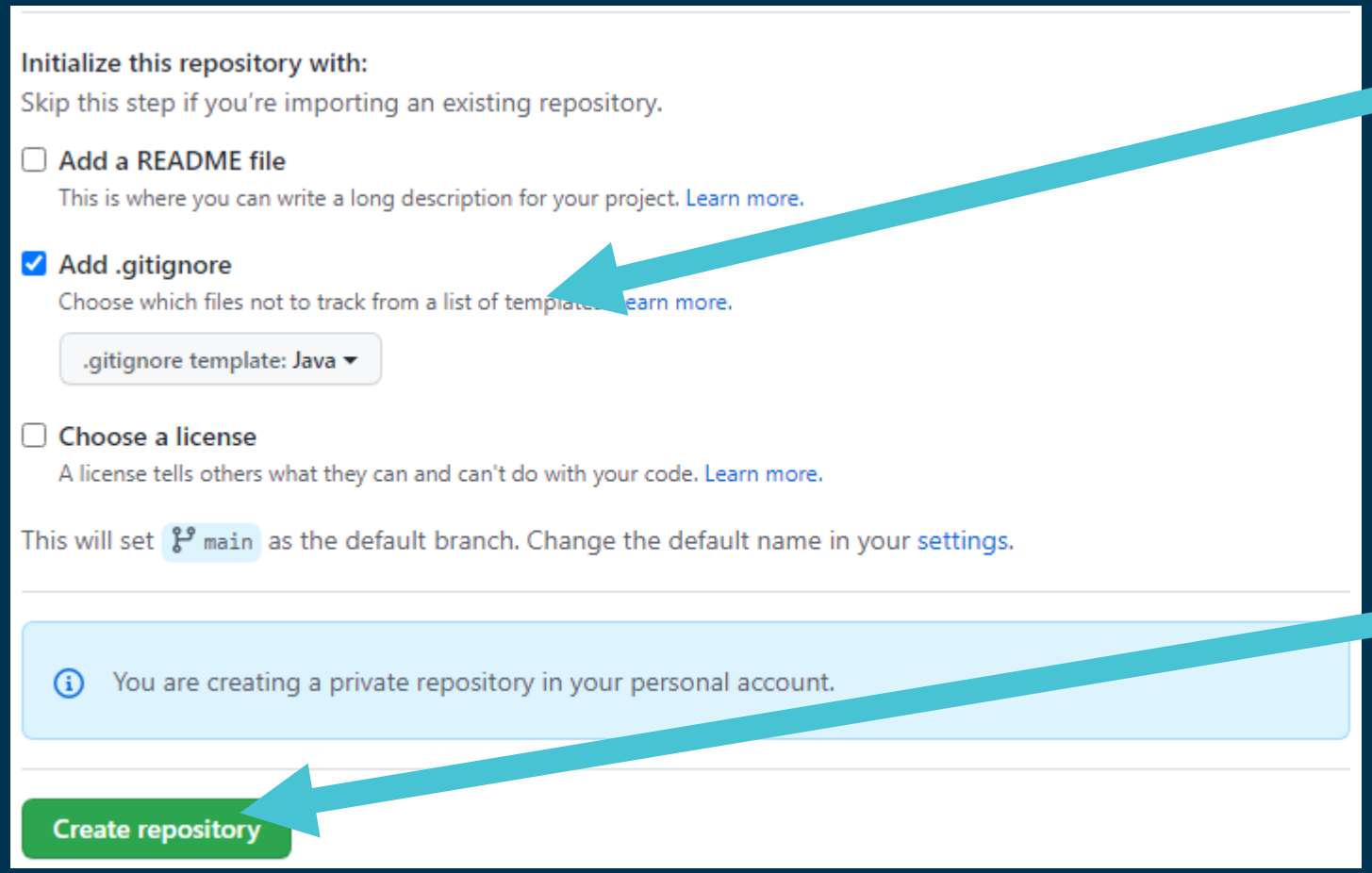


Repository name

Comment, if wished

Private or Public access

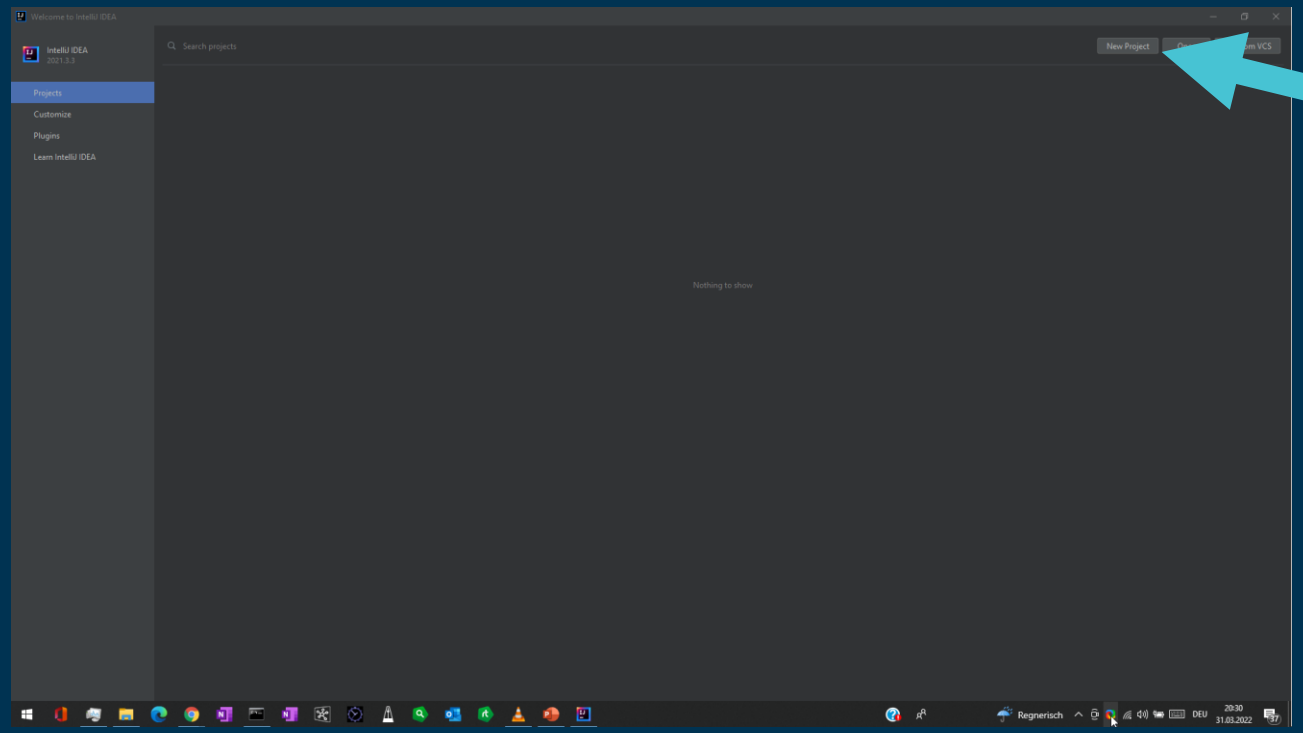
Create new Repository 2 of 2



Add .gitignore for JAVA

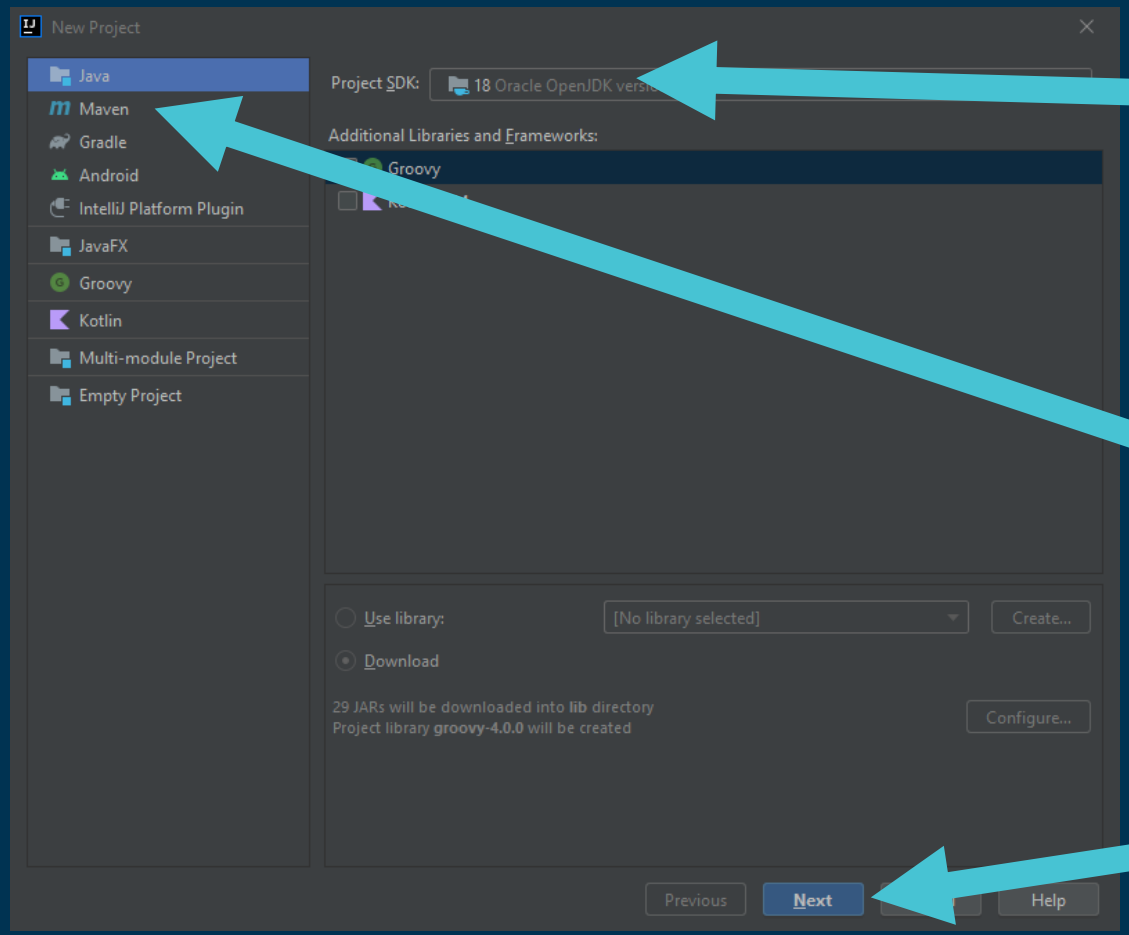
Create it

Creating a new project 1 of 5



New Project

Creating a new project 2 of 5

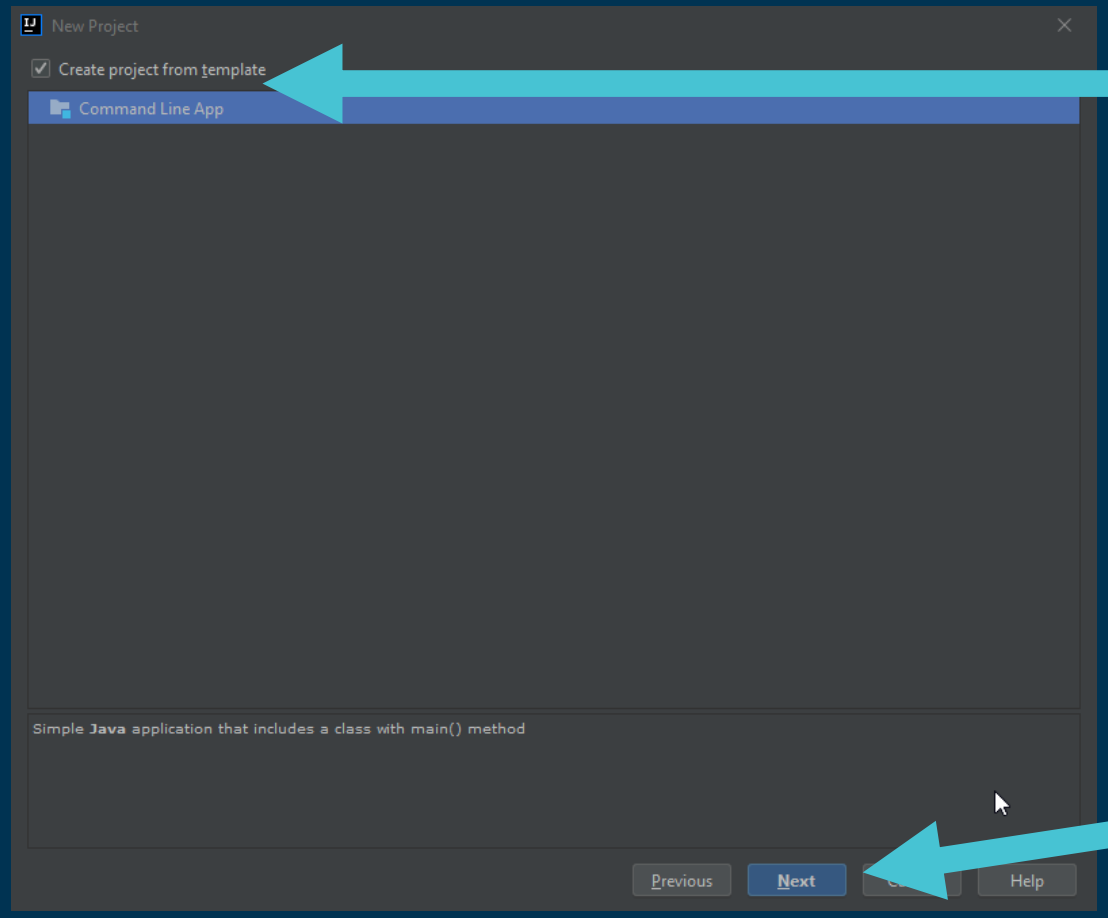


Make sure your JDK is listed here

Java

Press Next

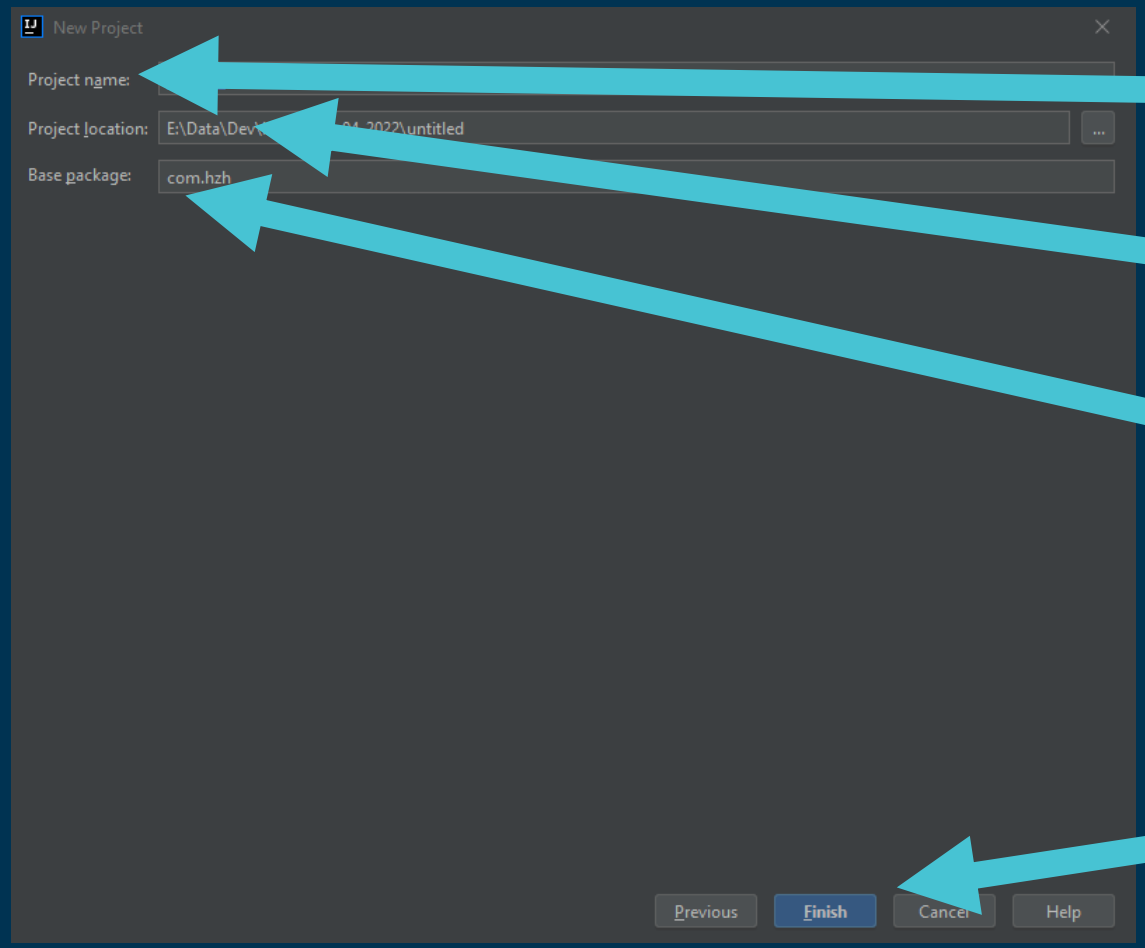
Creating a new project 3 of 5



Choose the Commend Line Template to get started

Press Next

Creating a new project 4 of 5



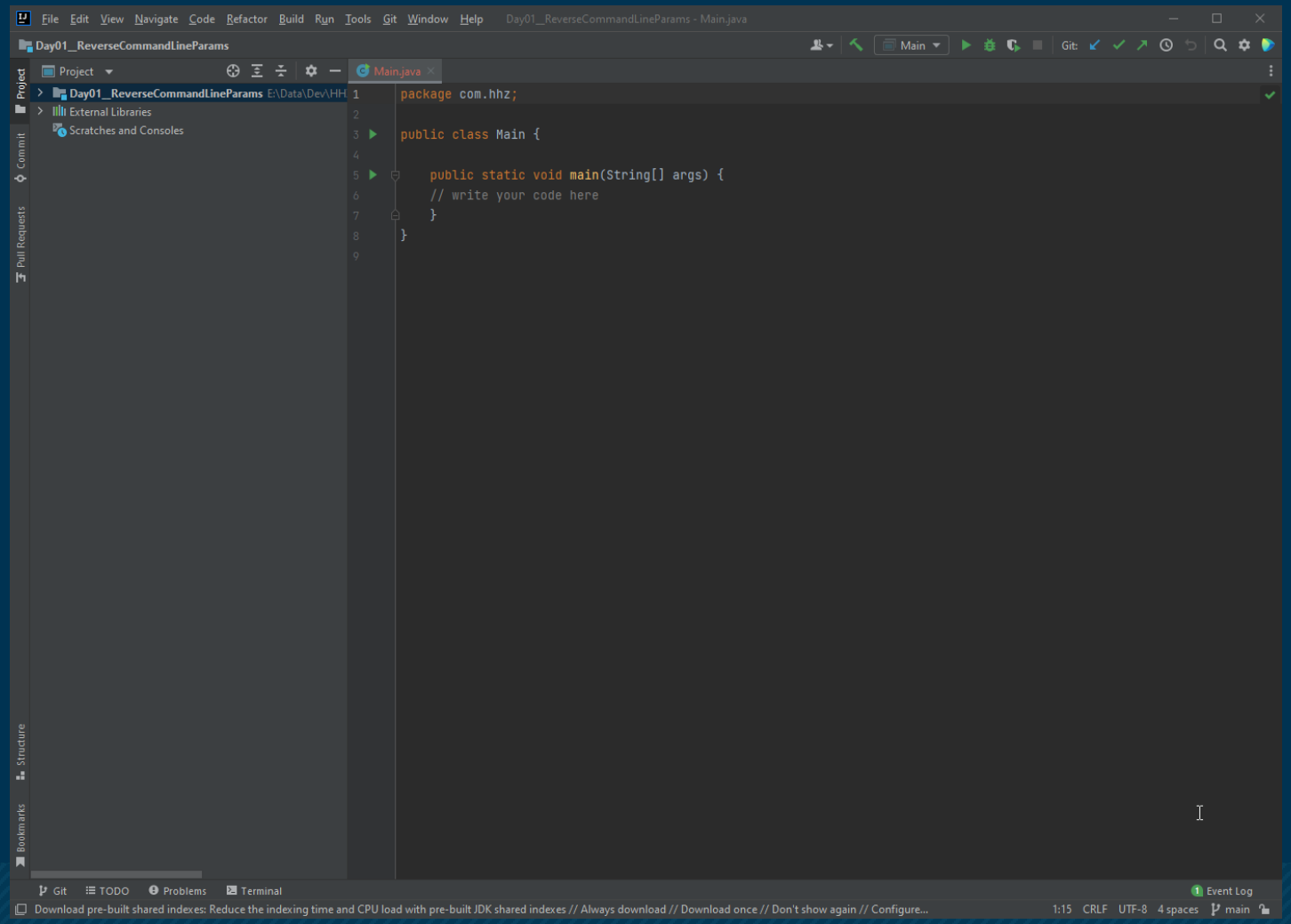
Give it a name

Local directory

Name space

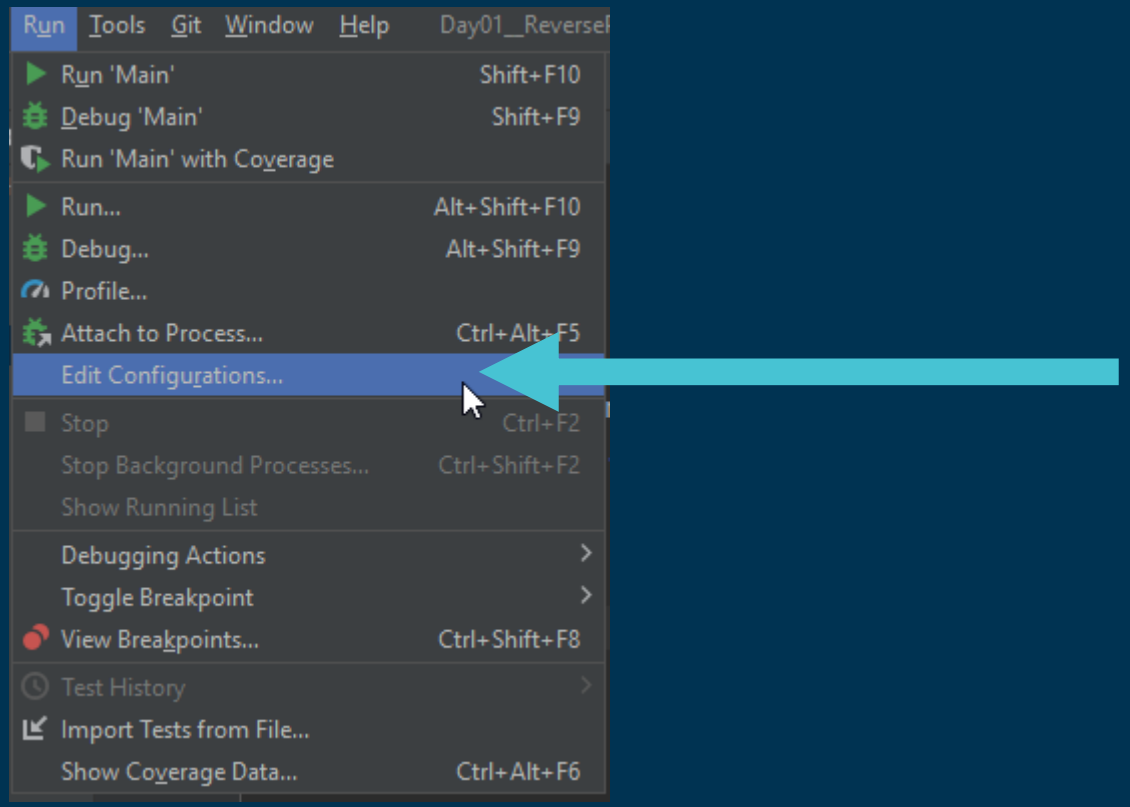
Press Next

Creating a new project 5 of 5



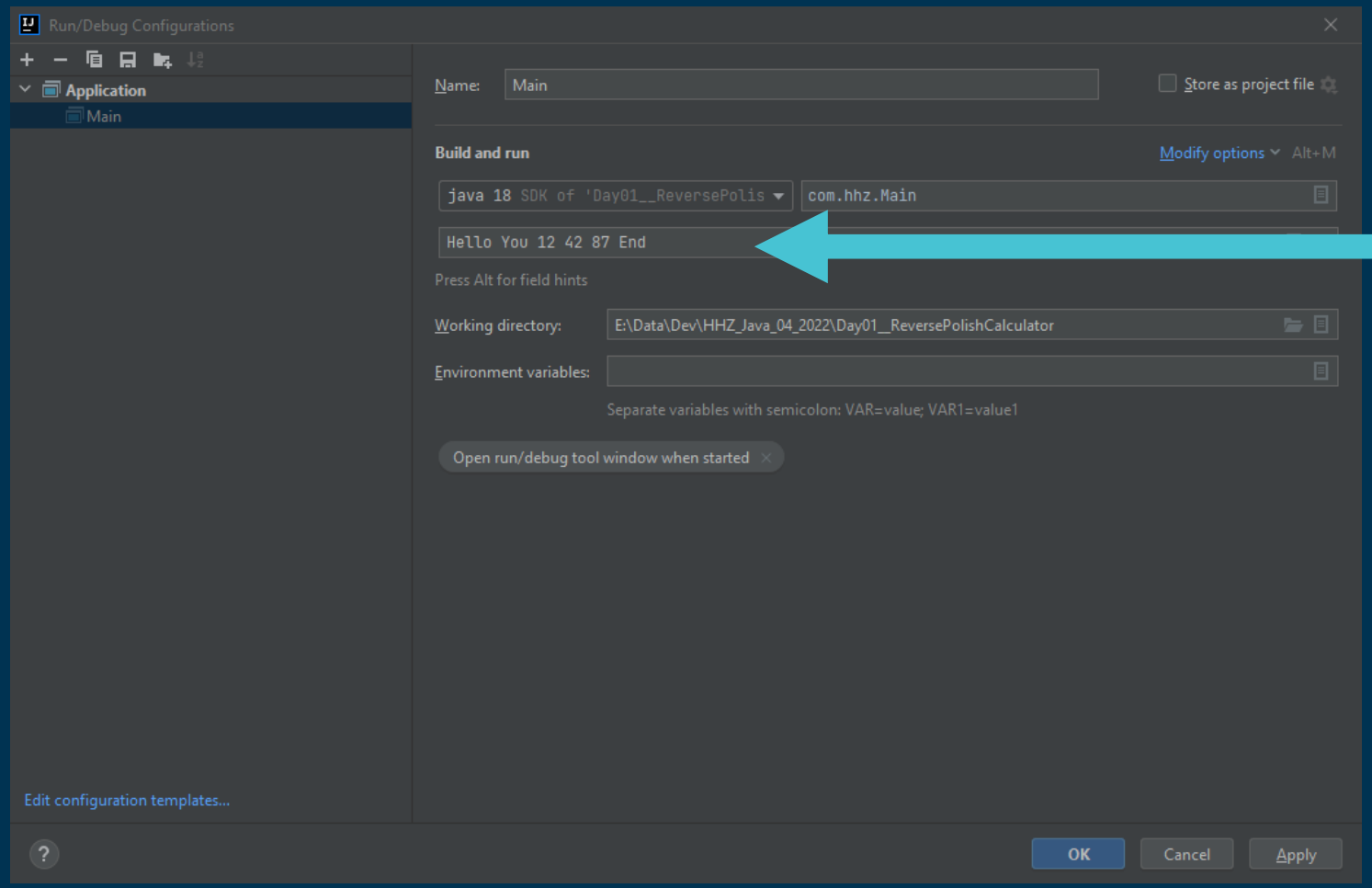
Resulting project

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

```java
package com.hhz;

public class Main {

    public static void main(String[] args) {
        for( int i = args.length; i>0;  i--) {
            System.out.println("Param #" + i +":    " + args[i-1]);
        }
    }
}
```

Some methods of the String class

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

- 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: Integer. parseInt(„411“) → 411
  Integer. parseInt(„Axx“) → Throws exception

Try - catch - finally

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

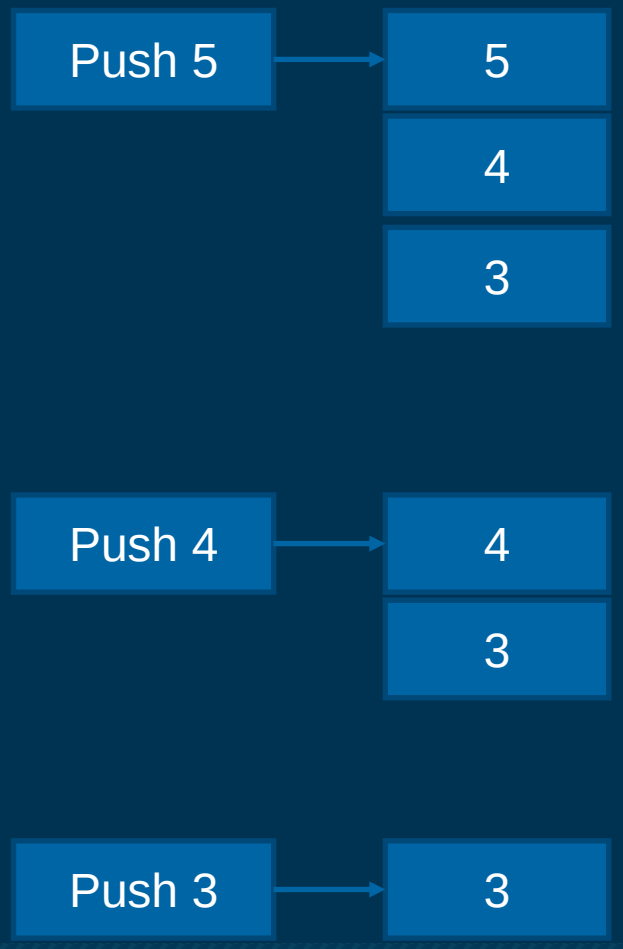
```
try {
        // Code, which might throw exeptions
} catch( Exception ex ) {
        // Code to run if an exeption happened
} finally {
        // Code, which runs wether an exeption was thrown
}
```

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

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

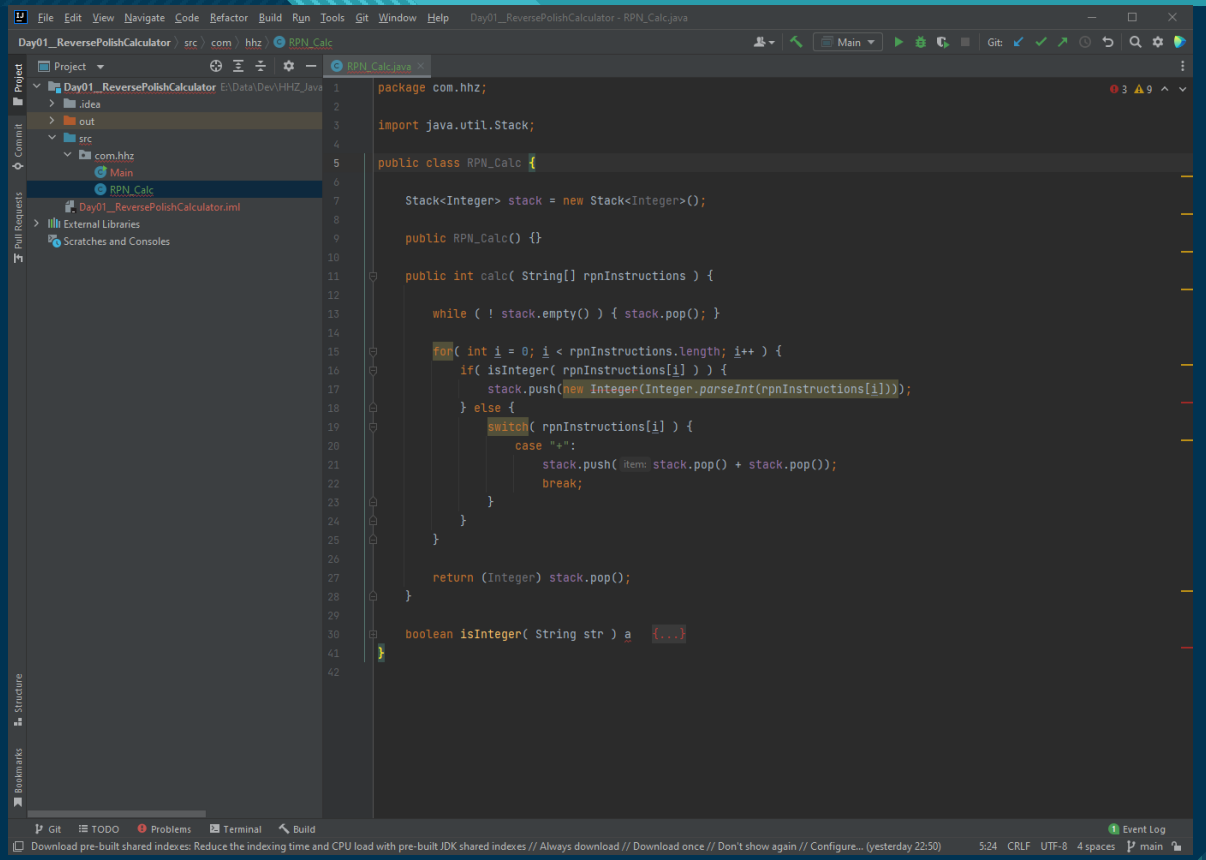


Unit testing

- 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.

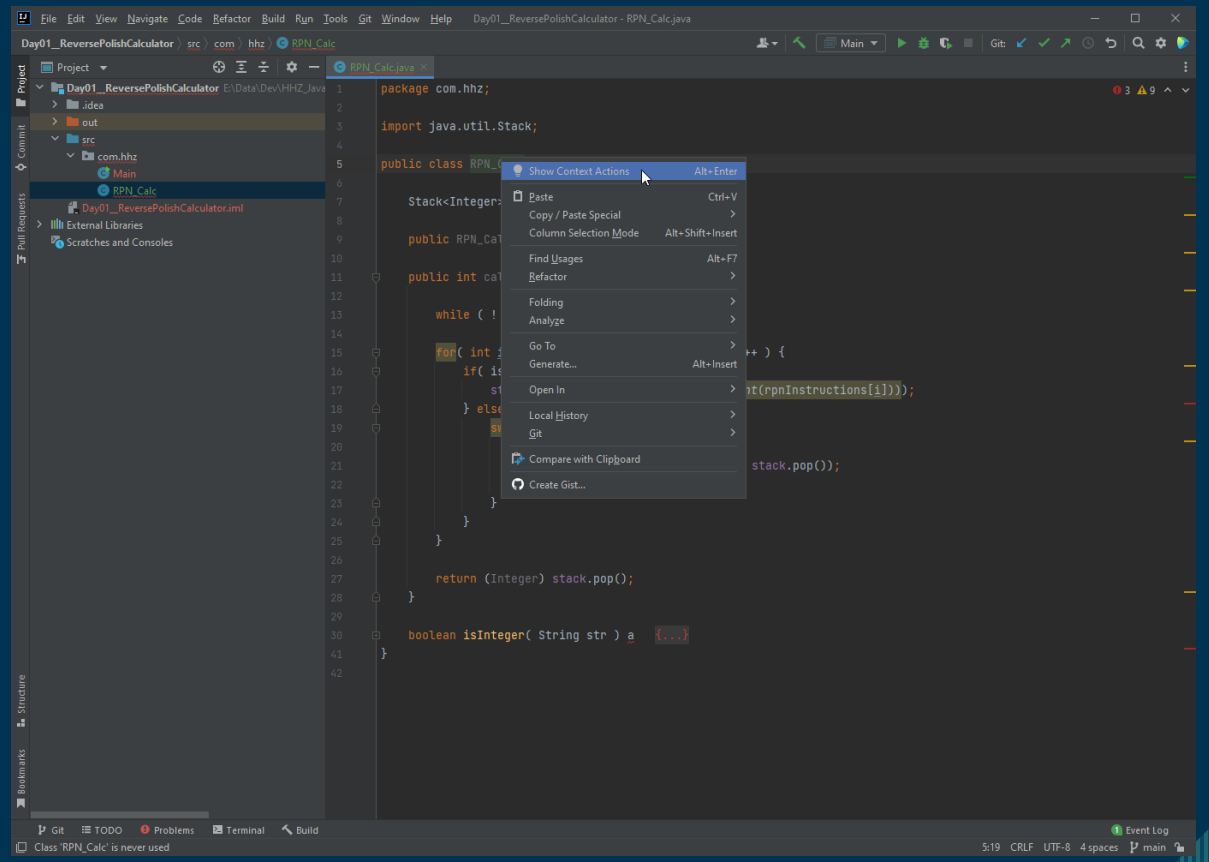
Unit testing

- In order to create Unit tests move the cursor over the class name and press the right mouse button → context menu



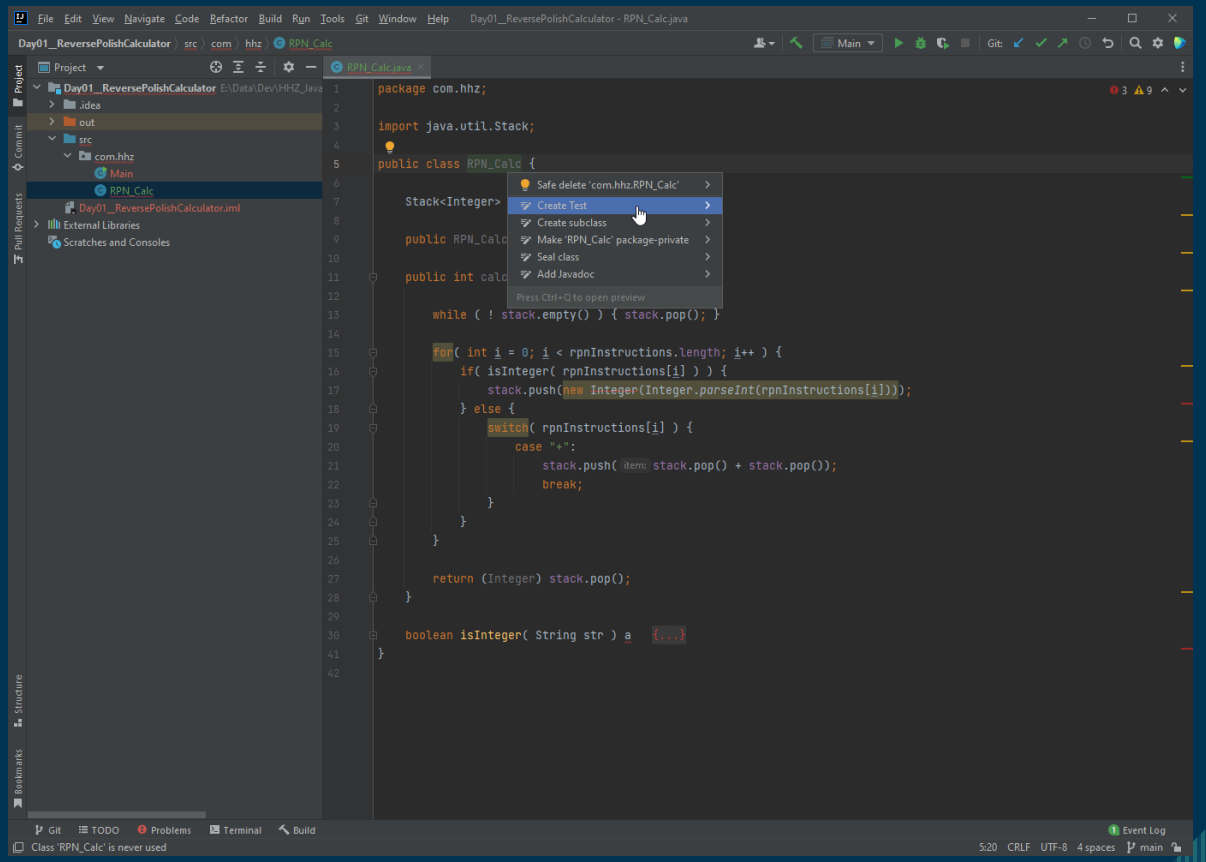
Unit testing

- Choose „Show Context Action“



Unit testing

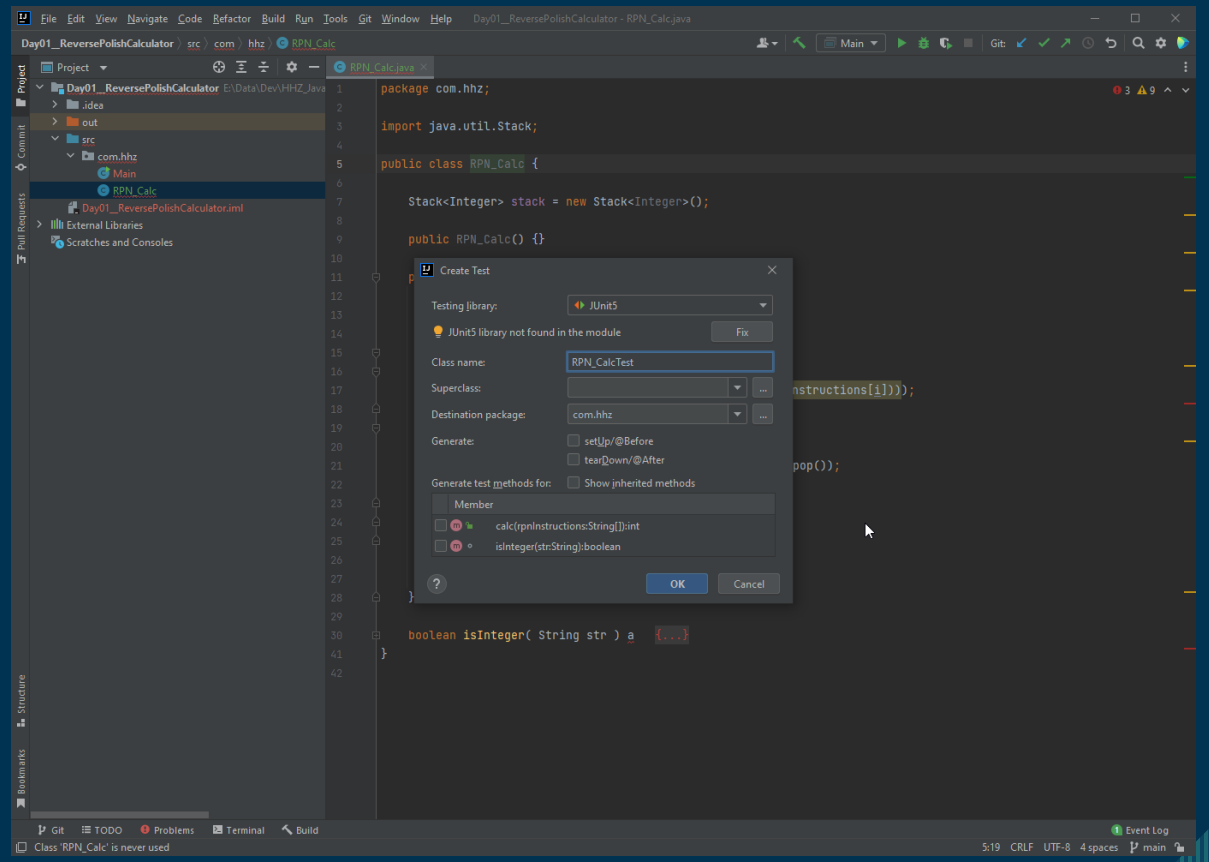
- Choose „Create Test“



Unit testing

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

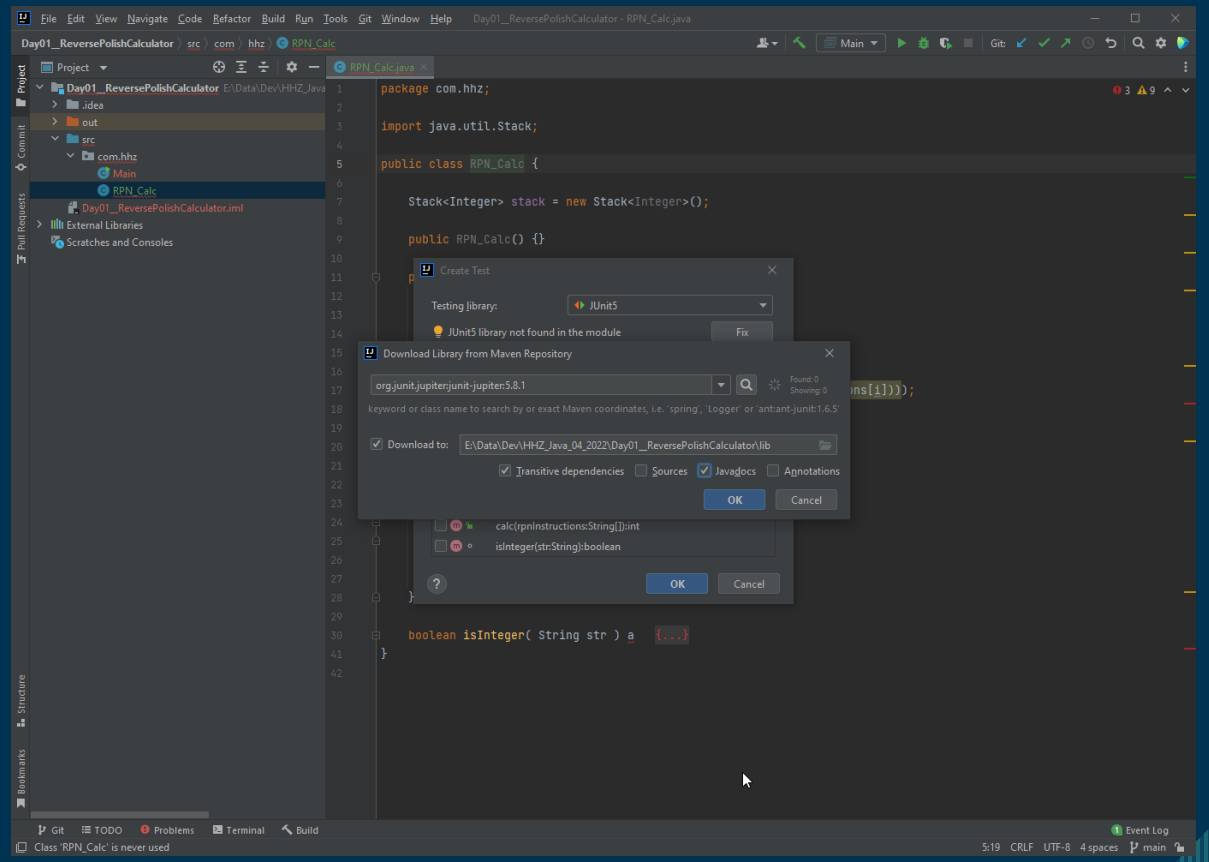
  Press „Fix“



Unit testing

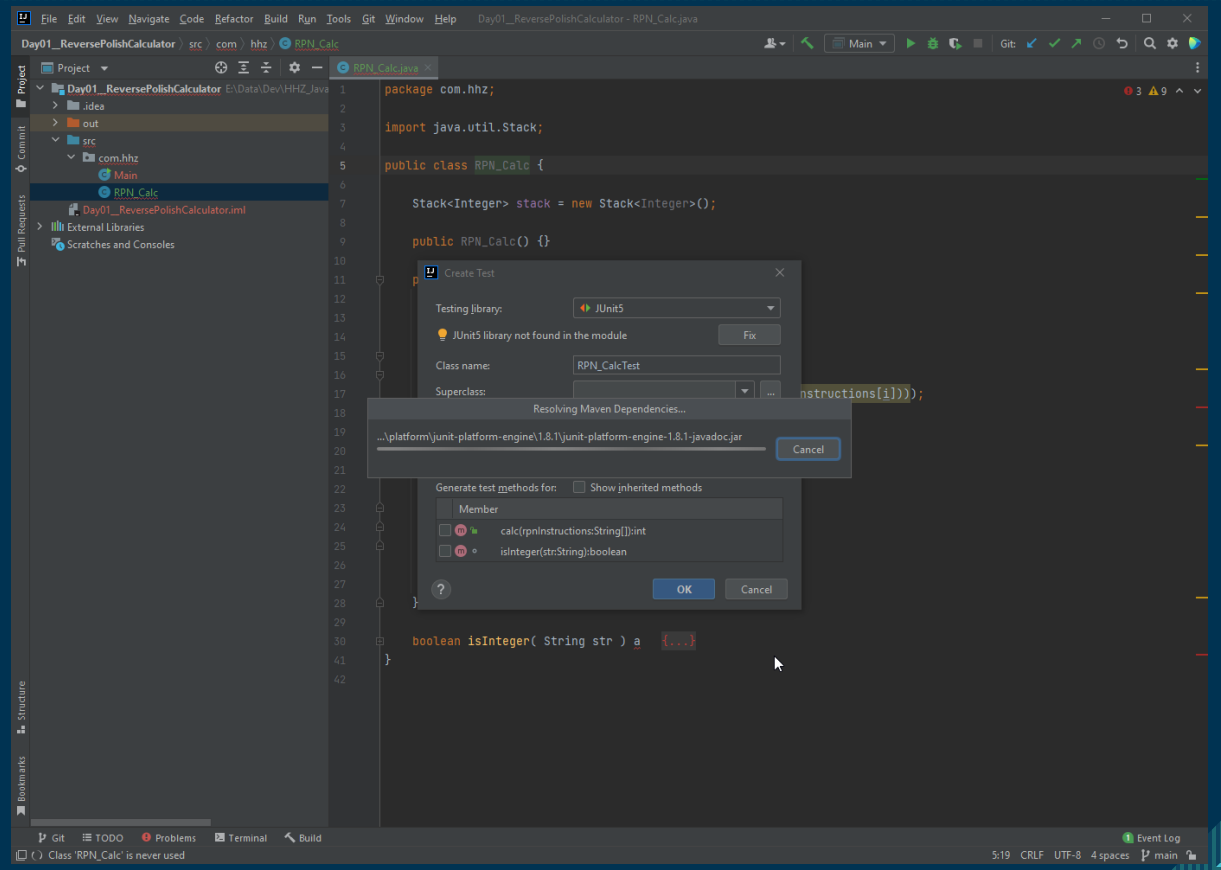
- Add the Junit jar file

  Download Javadoc as well.



Unit testing

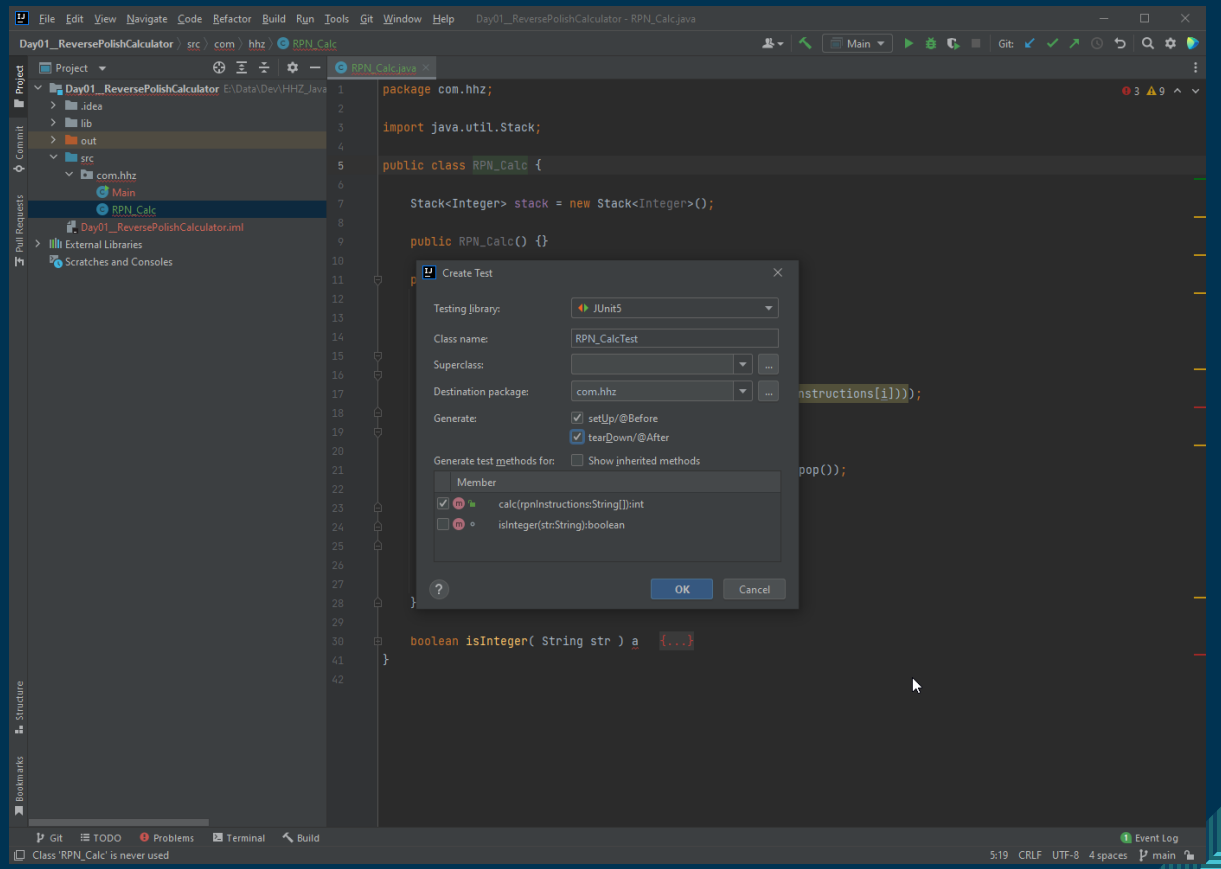
- IntelliJ downloads the required files



Unit testing

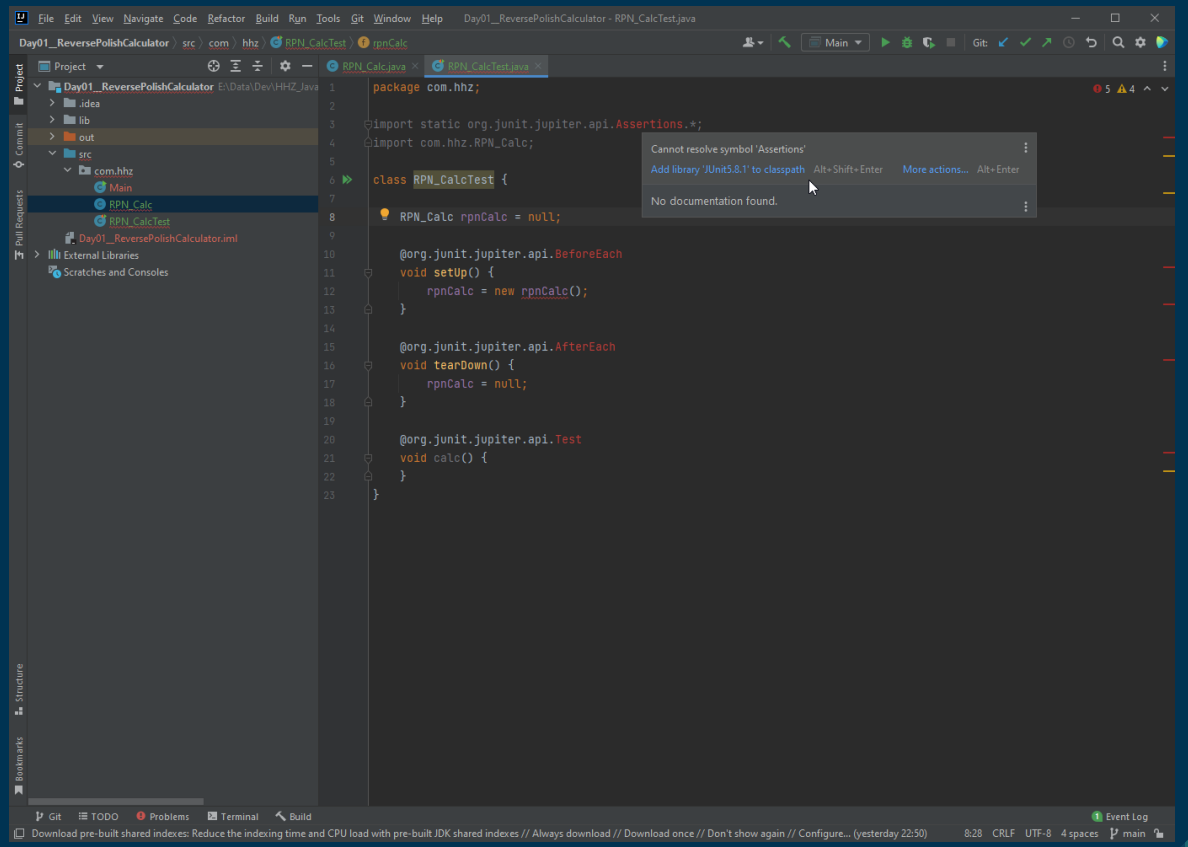
- Create the Junit test skeleton for the calc method

  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 Makrdown tags as well

Exercise Converter - Hints

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

SWING - User Interface

A simpe 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 JLable

Excercise 2

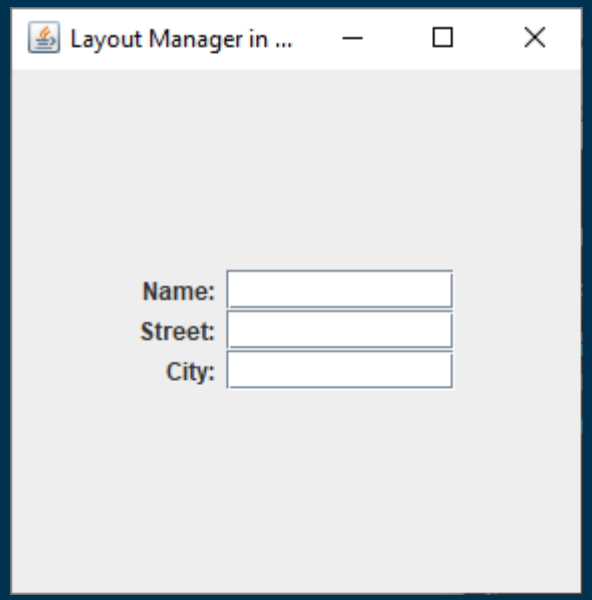
- Add a button to the window
- Anything the matter?
- Try adding a second button
- What happens then?

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 manages are shown here.

Excercise 3

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



Working with events

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

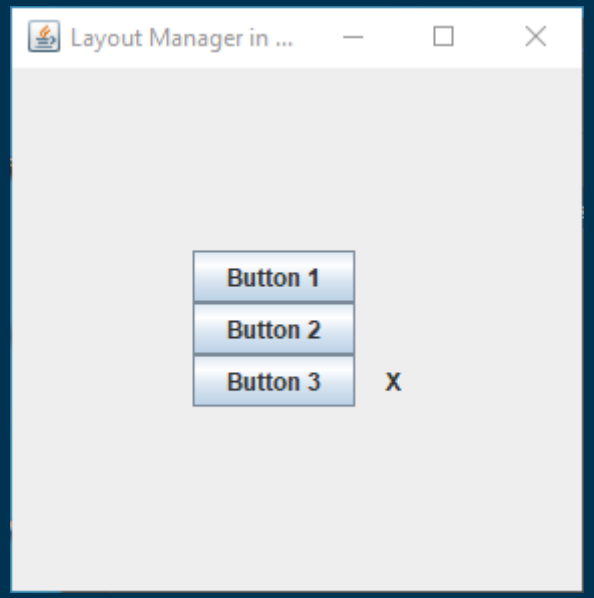
```
button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println( ((JButton)e.getSource()).getText() );
        }
    });
```

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

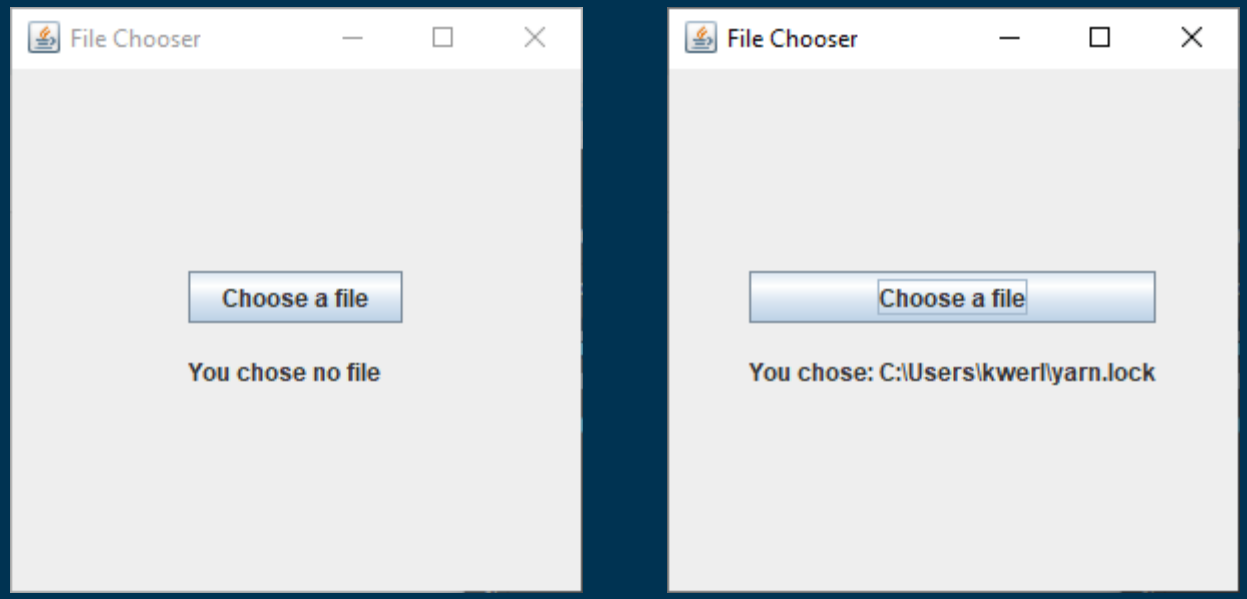
- JFileChooser brings up a dialog box for choosing a file name

```
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) →

Excercise 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.

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__CreateRelationalDBinHSQLDB project into IntelliJ
  - Add the hsqldb.jar file to the project.
  - Compile and run it.

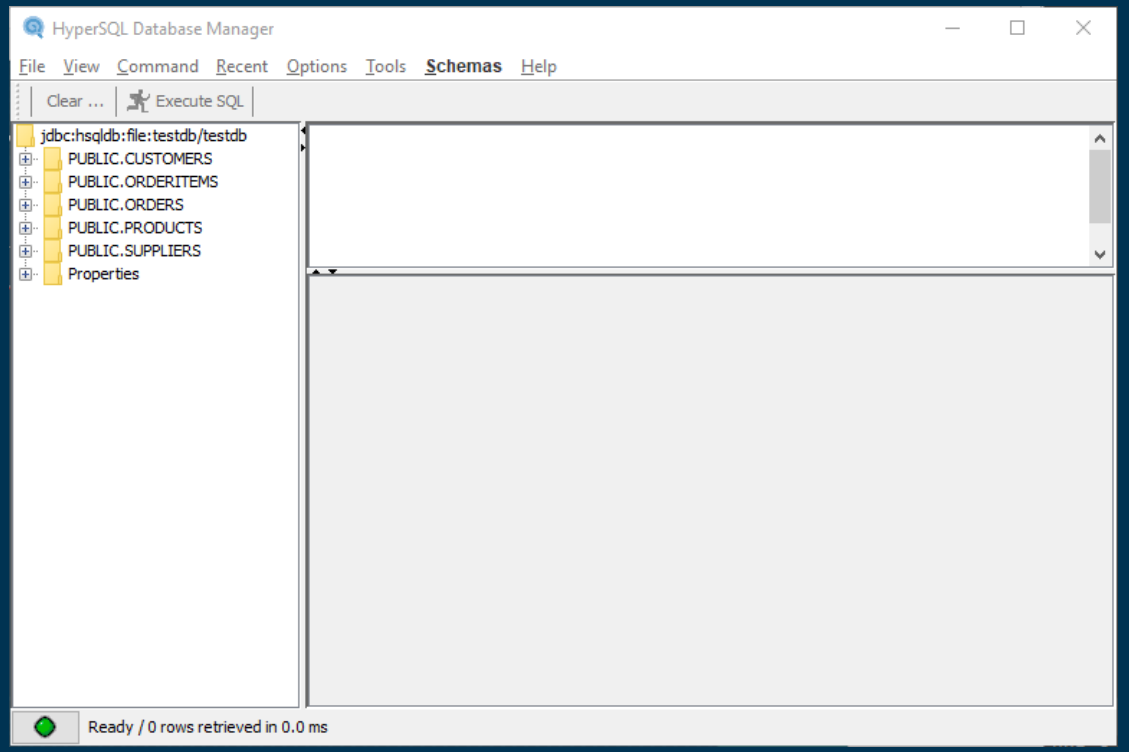
Databases -Relational Databases

- 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	

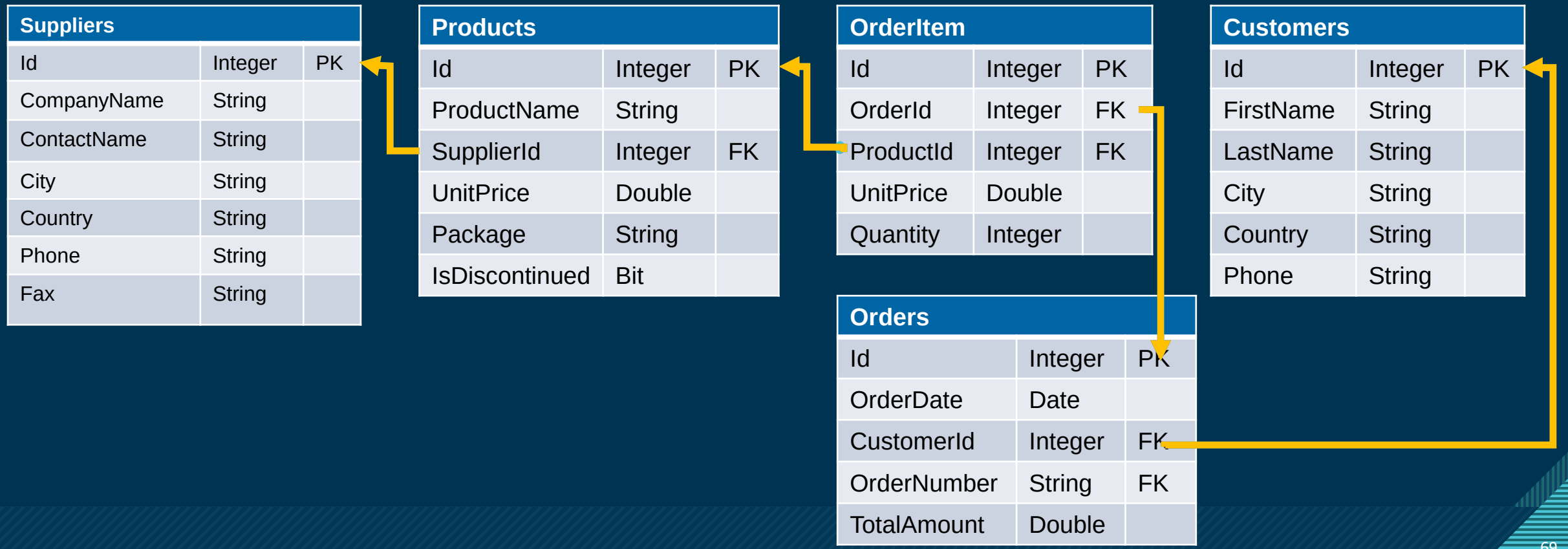
OrderItem		
Id	Integer	PK
OrderId	Integer	FK
ProductId	Integer	FK
UnitPrice	Double	
Quantity	Integer	

Orders		
Id	Integer	PK
OrderDate	Date	
CustomerId	Integer	FK
TotalAmount	Double	

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

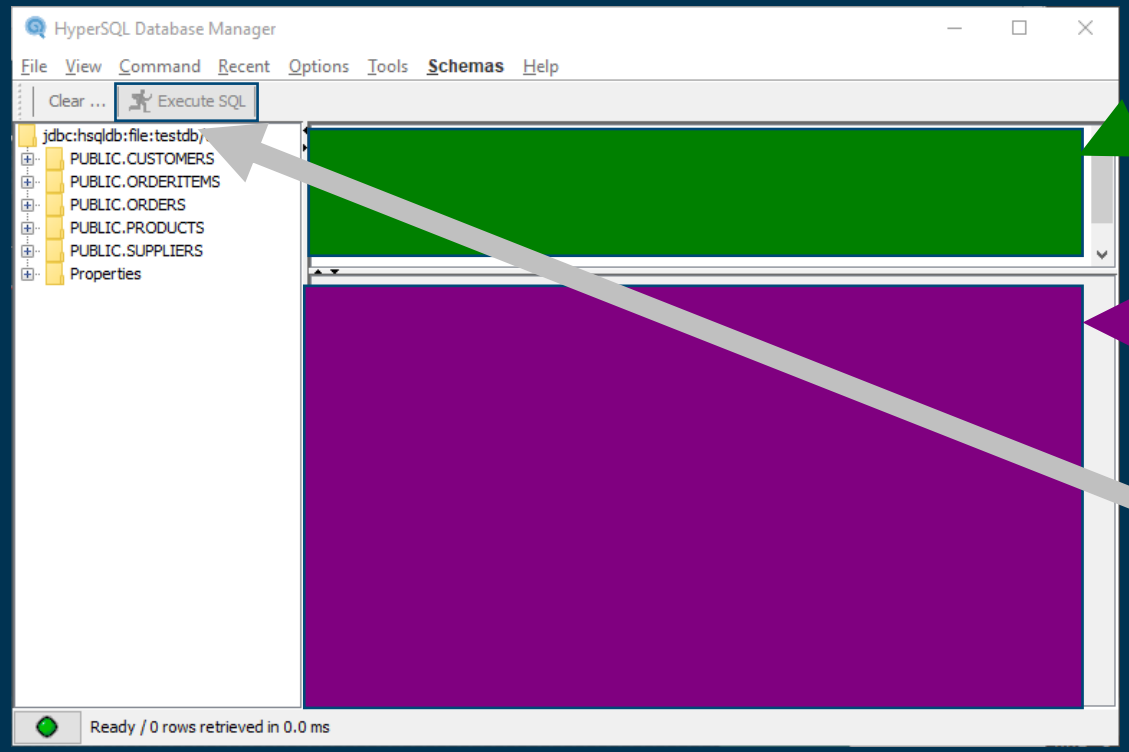
Databases -Relational Databases

- 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 date 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 (correlated subqueries):

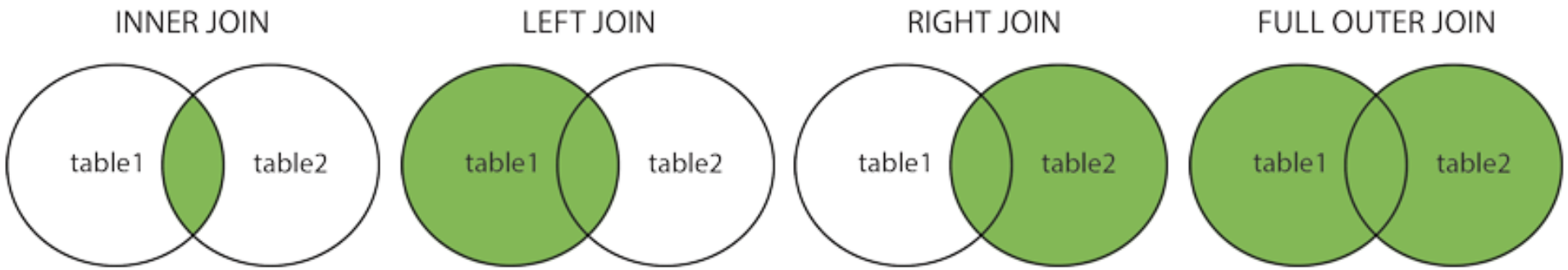
```
select s.companyname, (     select count(*) as c
                            from products as p
                            where p.supplierid = s.id
                            group by p.supplierid
                        )   as PRODCOUNT
from suppliers as s
where 3 < (     select count(*) as c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



URL

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

Numbers		
Id	Integer	PK
Number	Integer	

- Using PreparedStatement:

- 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 ) {
   ….
   Stm.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 succesfully 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 implicitely.

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

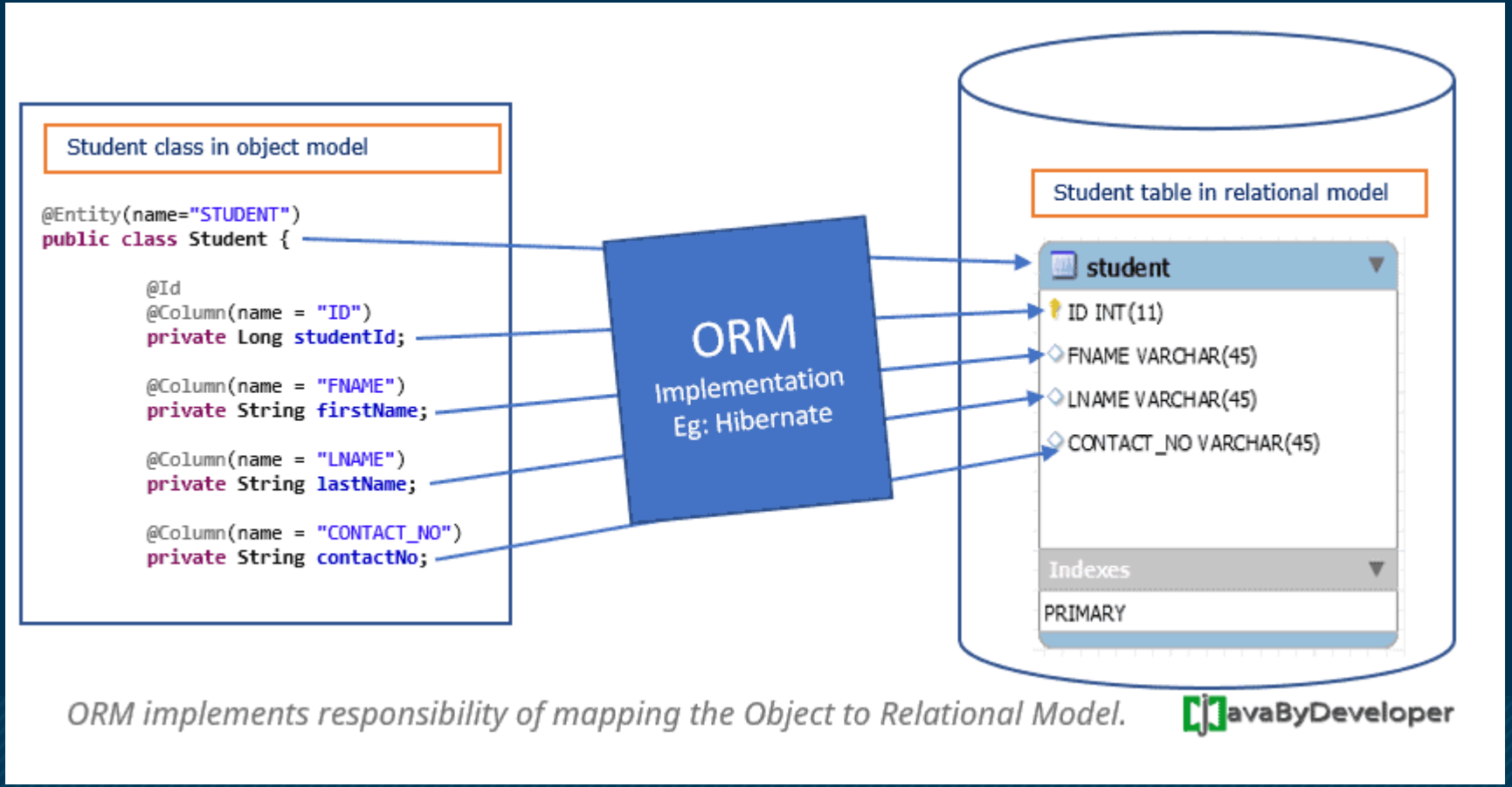
    (Discuss the impact of the field TotalAmount in Orders)

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: put(key, object)
    Java Class instance add: putString (key, stringObject)
    putInteger(key, integerObject) ….

  - Retrieve data from MemKV:

    Generic Object retrieval: get(key);
    Java Class instance retrieval: getString (key)
    getDouble(key) …….

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:	size()
Is DB empty?	isEmpty()
Remove all data from DB	clear()
Delete an entry	delete(key) or remove(key)
Does key exist?	containsKey(key)
Get copy of DB	getCopyOfDB()
List of all keys	getListOfKeys()
Add data from other MemKV DB	addOtheDb(db)
Loop through data	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 persistet 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.