Object-Oriented Programming in Java

Lecture 1 - Organization and Introduction

Emily Lucia Antosch

HAW Hamburg

12.08.2025

Contents

1. Organization

1.1 The Goal of This Chapter

- I want to introduce myself to you and discuss the course structure of this module.
- You will get an overview of the prerequisites for this module and will be able to meet them.
- You will know how to reach me.

1.2 About Me

- Emily Lucia Antosch, 25 years old
- Bachelor's degree in Electrical and Information Technology
- Currently working as an application developer at NVL
- Currently doing my Master's degree in Practical Computer Science
- Email: emilylucia.antosch@haw-hamburg.de

Info

I'm rather new with teaching at university, so please be lenient with me!

1.3 Course Schedule

- Lectures are divided into sessions on Tuesdays and Thursdays.
  - At the beginning, there are many sessions designed to prepare you for the lab.
- I would ask you to actively participate in the lectures.
- There will be small questions and tasks that you can answer live and code along with.

! Memorize

If you don’t understand something, please ask immediately!
I’m more than happy to repeat any part of the lecture!

1.4 Lecture Focus

- We want to build upon your prior knowledge.
- By the end of the lecture, you should be able to create simple programs in Java.
- Also, we want to look at mastering object-oriented programming paradigm and be able to highlight the differences to other programming paradigms.
- You can also read about the exact content outside of the lecture in the module handbook.

1.5 Prerequisites

- You will need an installation of the **Java SDK**.
  - For this, I have written a guide that you can find in the Moodle room.
- Additionally, the lecture will use the tool **JetBrains IntelliJ**.
  - This is, in my opinion, a very good and simple IDE for beginners.

2. Introduction

2.1 The Goal of This Chapter

- You will be able to apply your existing knowledge from previous lectures to new content.
- You will understand the fundamental concepts of object-oriented programming and understand the difference from programming in C.
- We will create a simple program in the IntelliJ IDEA development environment and execute it.

2.2 Topic Overview: Fundamentals

The first lectures focus on the following principles:

1. Imperative Concepts
2. Classes and Objects
3. Class Library
4. Inheritance
5. Interfaces

2.3 Topic Overview: Advanced Concepts

From the fundamentals, we then want to derive further concepts:

6. Graphical User Interfaces
7. Exception Handling
8. Input and Output
9. Multithreading (Parallel Computing)

2.4 Objects and Classes

In the real world, things are often determined and described by their properties:

- A car has properties such as
  - a manufacturer
  - a color
  - fuel consumption

> 💡 **Idea**
>
> Using object-oriented programming, we can apply this intuitive approach to programming as well!

2.4 Objects and Classes

? Question

What properties could you use to describe a person, for example? How might this fit into the programming context?

2.4 Objects and Classes

? Question

What properties could you use to describe a person, for example? How might this fit into the programming context?

- For students:
  - Name, address, student ID number
- For programs/websites:
  - Username, password, join date

2.4 Objects and Classes

- To create multiple similar objects from this similar blueprint, a class is created:
  - It contains all the properties we just defined in variables.
  - From it, completely different objects can be created that have these properties filled in differently.

Example

From the **Student** class, for example, the two students **Max** and **Ines** can be created, who both have different names and their own student ID number.

2.4 Objects and Classes

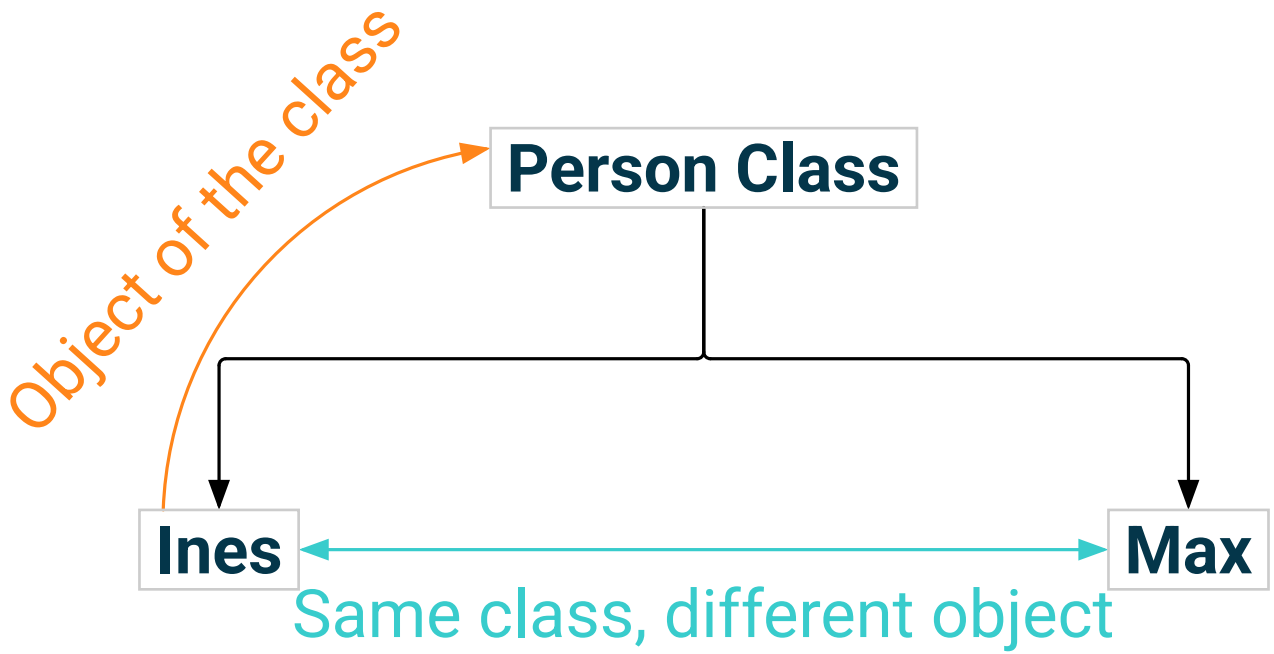


Figure 1: Relationship between classes and objects of that class

2.4 Objects and Classes

- Variables and functions are thus combined into a class.
  - A set of variables is defined.
  - For these variables, functions are introduced that can read and modify them.

! Memorize

- Variables are called **attributes**.
- The values of these variables describe the **state** of the object.
- Functions are called **methods**.

2.4 Objects and Classes

- So-called UML class diagrams can be used to describe classes with their **attributes** and **methods**.

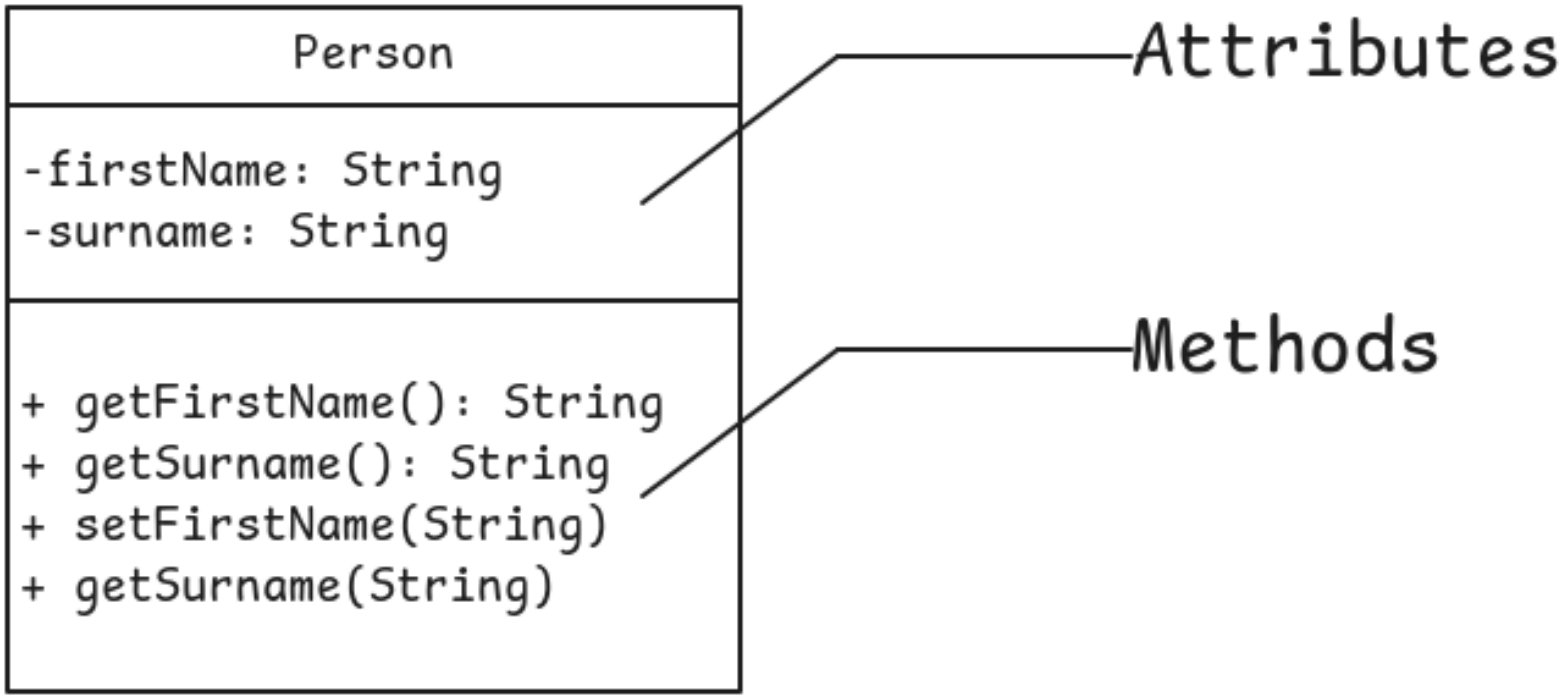


Figure 2: UML Class Diagram

2.5 Data Encapsulation

- Data can be encapsulated using attributes and methods of classes.

2.5 Data Encapsulation

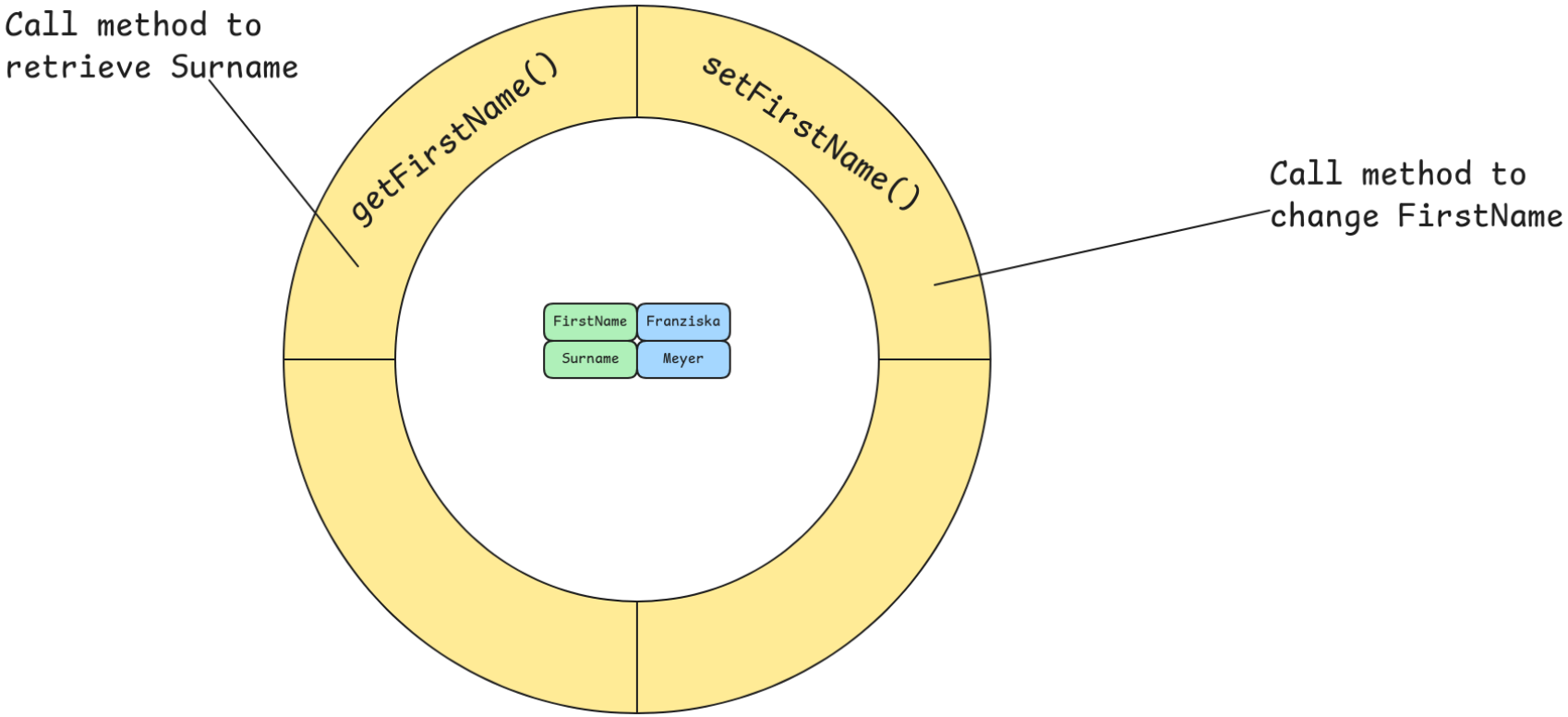


Figure 3: Data encapsulation through classes

2.5 Data Encapsulation

- Data can be encapsulated using attributes and methods of classes.
  - Not all parts of the program can access encapsulated data, which increases security.
  - Additionally, attributes can be protected from erroneous values this way.

2.5 Data Encapsulation

? Question

Where do the differences lie in comparison to the C programming language?

2.5 Data Encapsulation

? Question

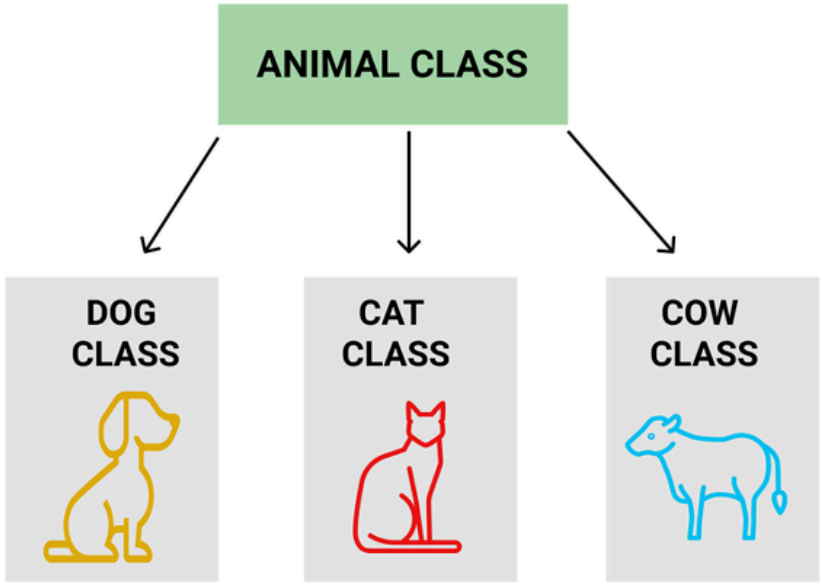
Where do the differences lie in comparison to the C programming language?

- The data structure (i.e., the **struct**) must be made public for access to the elements.
- The data is not protected.
- There is no association between the data and the functions.

2.6 Inheritance

- Through inheritance, new classes can be created from other classes.
  - The methods and attributes of the base class are inherited and extended with additional code.
  - No duplicated code.

2.6 Inheritance



2.7 Composition

- Classes can also be composed of other classes.
- This is called **composition**.
- For example, the **House** class would be composed of **Windows**, **Walls**, and **Doors**.

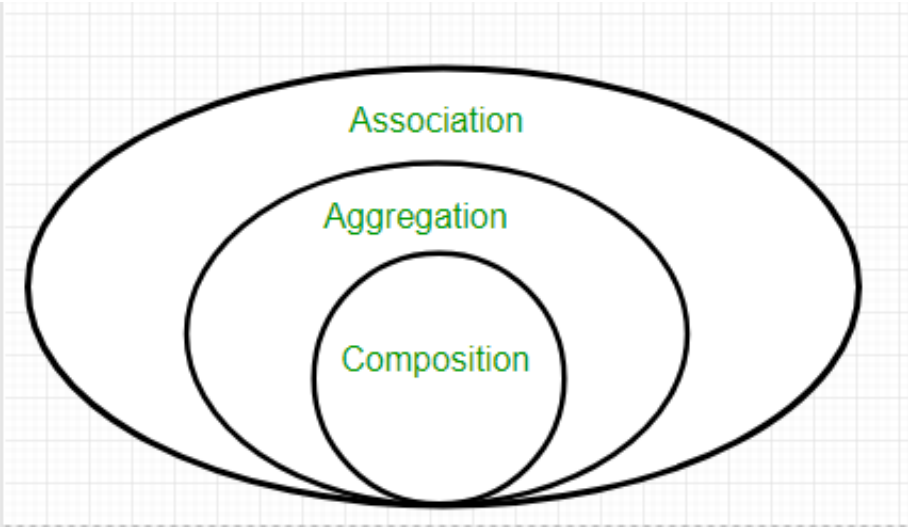


Figure 5: Composition of a given class

2.8 Program Execution in Java

1. At program startup, a special **main** method is executed in the *main object*.
2. In this method, objects are created and the **references** to these objects are stored in variables.
3. These variables can then be used to access the respective object.
4. Objects in the program can then create additional objects and call methods.
5. As soon as the **main** method is finished, the program ends.

2.8 Program Execution in Java

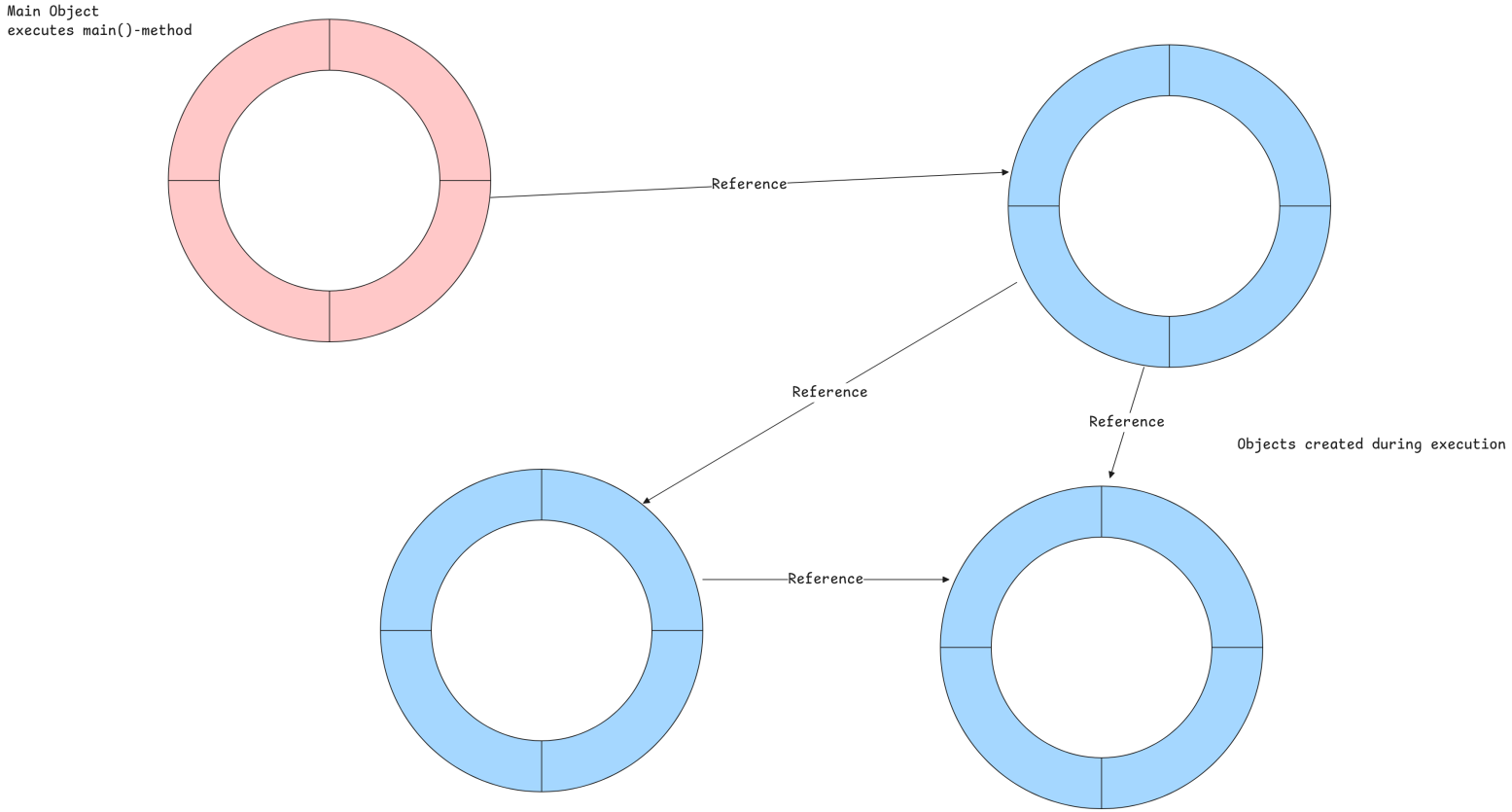


Figure 6: References in the lifetime of a program

3. The Java Programming Language

Task 1

Let's first write a few simple tasks in the programming language C:

- Sum of numbers 1 to n using a **for** loop.
- Maximum of two numbers using an **if** statement.
- Determine the maximum of two numbers using the **getMax()** function.

3.1 Java vs. C

- I have good news: You could have executed this code in Java without any problems!
- The **syntax**, i.e., the keywords and structure of the language, is very close to C and C++!
- Therefore, we want to continue building on your prior knowledge.

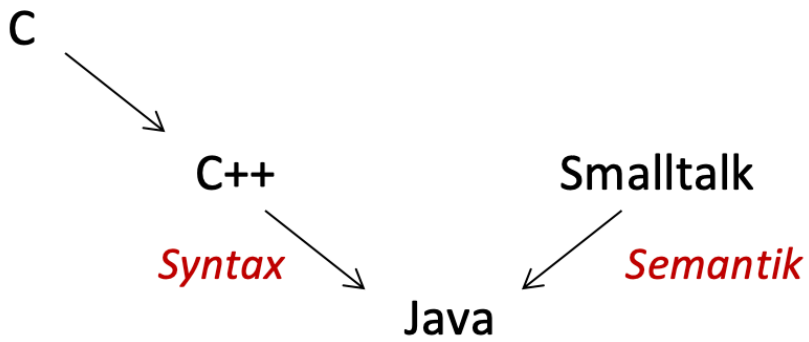


Figure 7: The influences on the Java programming language

3.2 Compilation

1. Development
   - Source code is written on the PC.
   - Compiler compiles source code into **bytecode**.
2. Execution
   - Bytecode is executed on the **JVM** (Java Virtual Machine).
   - Execution does not require recompilation for each target platform.

3.2 Compilation

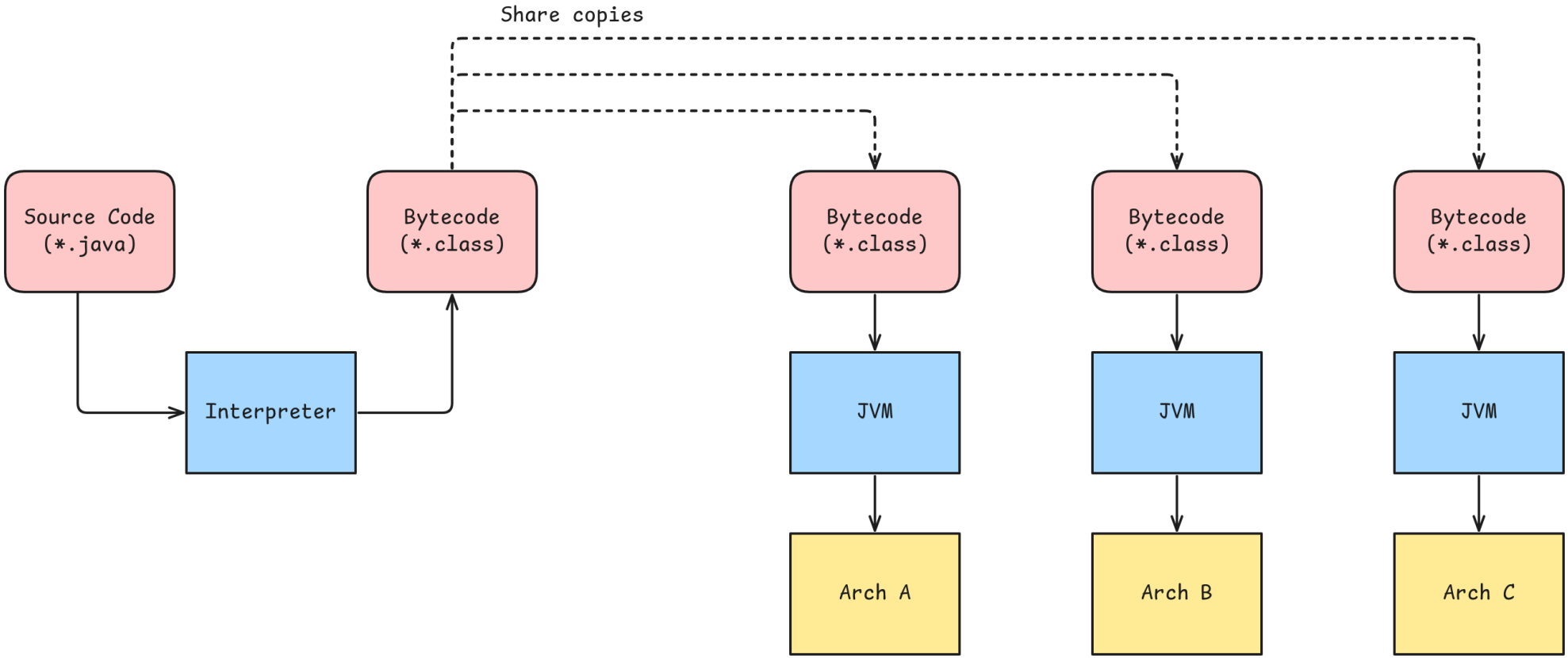


Figure 8: Program execution with the JVM

3.2 Compilation

- Differences in other programming languages that are compiled or interpreted:
  - **Compiled languages** must be recompiled for each target platform.
  - **Interpreted languages** must be interpreted by their own interpreter on the target platform itself.

3.2 Compilation

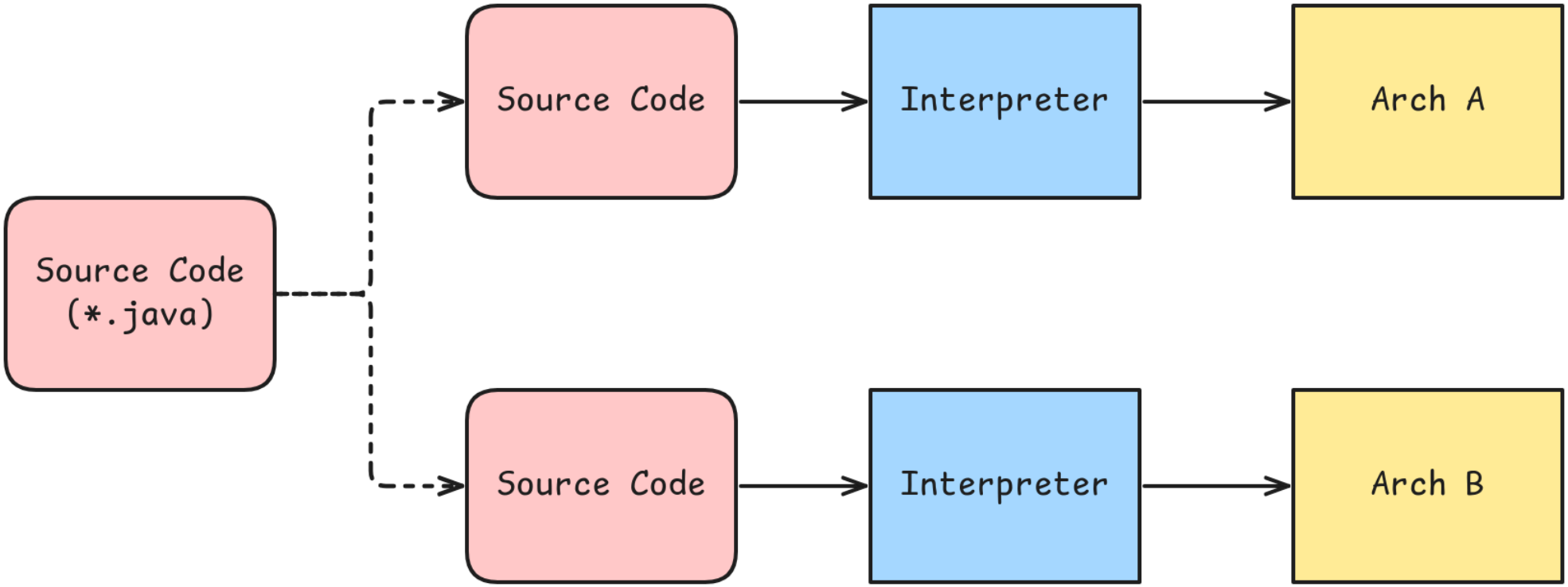


Figure 9: Execution of compiled languages

3.2 Compilation

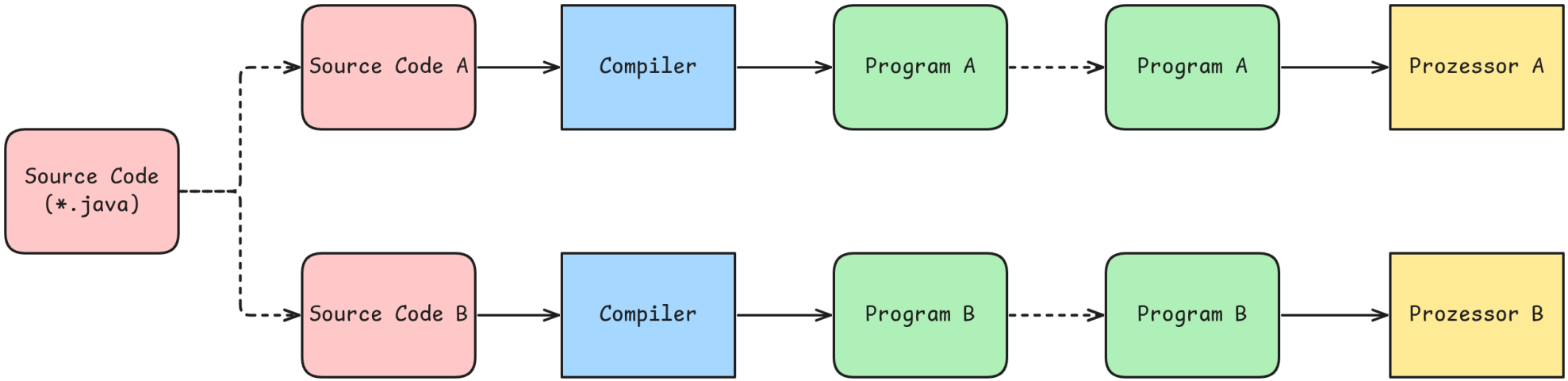


Figure 10: Execution of interpreted languages

3.3 Properties of Java

? Question

If you could develop a new language, what would be important to you? What would you change about C/C++?

3.3 Properties of Java

? Question

If you could develop a new language, what would be important to you? What would you change about C/C++?

- Java
  - Object-oriented language (i.e., classes, objects, and inheritance)
  - Platform-independent (via **JVM**)
  - Strongly typed (fixed types like **int** and **String**)
  - Robust (i.e., Garbage Collector)

? Question

Which is the better programming language: C or Java?

4. The First Program

4.1 IDE

- I would recommend IntelliJ IDEA from JetBrains as an IDE.
  - This tool will also be used in the exam.
  - The IDE also includes the Java JDK, which you need for programming.

- Choose the Community Edition at https://www.jetbrains.com/idea/download/?section=windows, or scan the QR code.

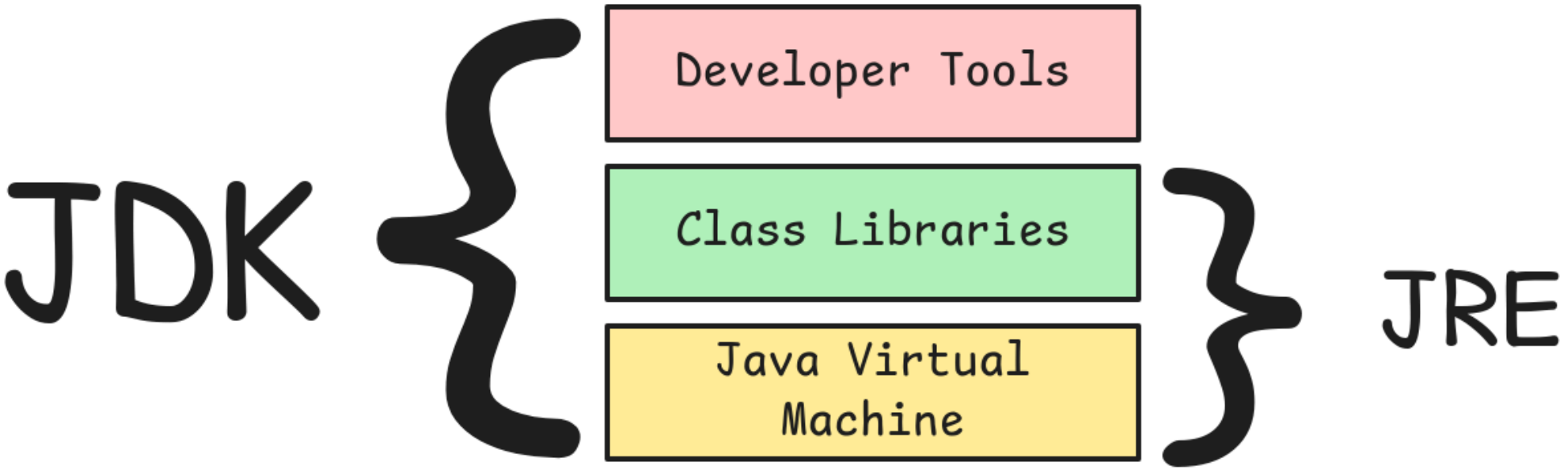


Figure 8: The structure of the Java Toolchain*

*Application Programming Interface

Task 2

- Preparation:
  1. First open a directory where you will store the programming files.
  2. Open IntelliJ IDEA.
- Create project:
  1. Select File > New > Project.
  2. Assign a name and a location.
  3. Choose Java and IntelliJ and the appropriate JDK
  4. Click “Create”

Task 3

- Create package
  1. Right-click on src
  2. Select New > Package
  3. Enter name
- Create class
  1. Right-click on package
  2. Choose New > Java Class

4.2 The First Program

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

Java

Idea

Enter the code into the file you just created. If you're already that far, feel free to code along!

4.3 Applications in Java

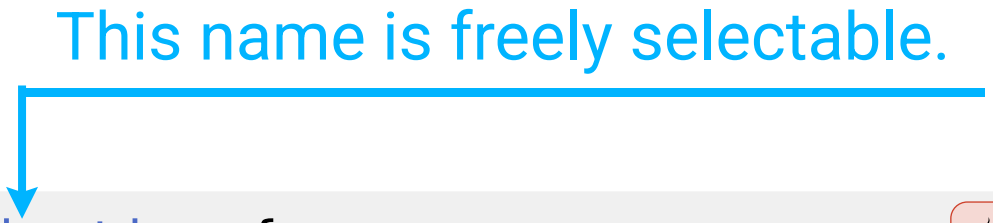
- A Java file can be executed if it has a public (**public**) class:
  `public class MyApplication {...}`
- The class must also have the same name as the file, for example `MyApplication.java`
- The class has the method: `public static void main(String[] args)`

4.3 Applications in Java

Java

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

4.3 Applications in Java



Java

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

4.3 Applications in Java

This name is freely selectable.

```java
1  public class MyApplication {
2    public static void main(String[] args) {
3      System.out.println("Hello World!");
4    }
5  }
```

This method must always be called mai

5. Literature

5.1 Literature Recommendations

- Some books that might help you during the course:
  - D. Abts: Grundkurs JAVA, Springer-Vieweg
  - H.-P. Habelitz: Programmieren lernen mit Java, Rheinwerk Computing

6. License Notice

6.1 Attribution

- This work is shared under the CC BY-NC-SA 4.0 License and the respective Public License
- https://creativecommons.org/licenses/by-nc-sa/4.0/
- This work is based off of the work Prof. Dr. Marc Hensel.
- Some of the images and texts, as well as the layout were changed.
- The base material was supplied in private, therefore the link to the source cannot be shared with the audience.