Session 1: Hello World! (Session Notes)

In which we learn:

- that software development is all about communication;
- that every programming is instigated by a problem;
- that development should start not with writing code, but with formalizing a language we can use to build code and reason about it;
- that delivering good software requires a common language, which is a complex and difficult task in itself;

- Software development is a vast field of human and technological endeavor filled with complexity.
- Every programming job **starts with a problem**. From the very moment a client comes to us and asks for a software solution, they present a problem that we as a developers have to solve.
- In order to solve a problem with the means of computers, we have to analyze it. In order to analyze it, we have to understand it. Understanding, again, requires language. It follows from this that all problem solving by means of software development **requires language**.
- Language is communication, and communication is language. Effective problem solving and effective software development thus hinges on effective communication and a **proper, common language**.
- One of the first steps of developing software is **requirements analysis**: What does the client need? What is the problem the client presents? Do we understand the problem properly enough to reason about it?
- Thus, we are dealing with the **problem domain** of software:
    - It comprises all **terms and concepts** needed to describe everything **the client** knows about **their problem space** — some explicitly, much only implicitly (unaware knowledge).

    - Such terms and concepts relate to the trade the client is involved in (e.g. the restaurant domain if the client is a pizza chef)
    - It is best described and reasoned about in its own language, the **problem domain language.**
    - It is the language we developers use to communicate with the client.
- When coming up with a solution, we need to implement the problem using models a computer can understand (programming languages and technologies).
- Thus, we are dealing with the **solution domain** of software:
    - It comprises all **terms and concepts** needed to describe everything **we developers** know about **our solution space** — most explicitly, but some only implicitly.
    - Such terms and concepts relate to technologies and techniques of software development (e.g. programming languages, design patterns, software stacks).
    - It is best described and reasoned about in its own language, the **solution domain language.**
    - It is the language we developers use to communicate among each other (but not to with client!).
- Both problem and solution domains are best documented in terms of **domain dictionaries** that contain all concepts contained therein and their definitions.

Example of a Problem and Solution Domain

- Andrea, a pizza chef from NY is coming to you because he wants to deliver pizza online.
- You come up with the following domain languages:

Problem Domain	Solution Domain
pizza, order, order item, toppings, customer, prices, menu, payment, …	**<Order> (class), <Customer> (class), UI, module, CSS, SQL, logging, …**

- Problem and solution domain are inherently complex in themselves.
- While the problem domain **permeates** through the boundary and into the solution domain (see e.g. **<Pizza> class, <Customer> class**), the reverse is not true. The solution domain (implementation details) is better kept separate from the problem domain.

Task: Create a domain language (dictionary) for the Problem Domain of our Wordle Clone.

- To do so, you should head over to the original game site (https://www.nytimes.com/games/wordle/index.html) and see how the game is structured:
    - What parts is the game composed of?
    - What operations are executed?
    - Think in terms of game concepts, not in terms of technological implementation!

Resources and Further Reading

- “Domain Driven Design” on Wikipedia (https://www.wikiwand.com/en/Domain-driven_design)
- Evans, Eric: Domain-Driven Design. Tackling complexity in the heart of software. Pearson Education. (Short review from Martin Fowler: https://martinfowler.com/bliki/DomainDrivenDesign.html)
- A summary of the book is freely available as a PDF/EPUB/MOBI download: https://www.infoq.com/minibooks/domain-driven-design-quickly