

Vocational English II
(Mesleki Yabancı Dil II)
Week 4



Engineering Faculty
Computer Engineering

Prepared by: Dr Ercan Ezin

INTRODUCTION

THIS WEEK WE WILL WORK ON

Software Engineering Principles

BLOG POST

TITLE: Top 10 Software Engineering Principles

<https://fullscale.io/blog/software-engineering-principles/>

INTRODUCTION TO SOFTWARE ENGINEERING PRINCIPLES



Software engineering is complex and multifaceted.



Principles help navigate challenges and ensure project success.



Key benefits: **quality assurance, efficiency, collaboration, maintainability, risk mitigation.**



Applying principles ensures long-term software success.

**WHY SOFTWARE
ENGINEERING
PRINCIPLES MATTER**

Quality Assurance: Reduces defects, improves reliability.

Efficiency & Productivity: Streamlines development, reduces waste.

Collaboration: Ensures clear guidelines for teamwork.

Maintainability & Scalability: Enables long-term modifications.

Risk Mitigation: Identifies and resolves issues early.

**PRINCIPLE #1 –
KISS (KEEP IT
SIMPLE, STUPID)**



Simplicity is key to maintainability and readability.



Avoid unnecessary complexity.



Clean, concise, and readable code improves efficiency.

PRINCIPLE #2 &
#3 – DRY & YAGNI

2- DRY (Don't Repeat Yourself): Avoid redundancy, promote modular design.

- Code reuse enhances efficiency and reduces errors.

3-YAGNI (You Aren't Gonna Need It): Only implement required features.

- Prevent over-engineering and unnecessary functionality.

PRINCIPLES #4 & #5 – SEPARATION OF CONCERNS & MODULARITY

4-Separation of Concerns: Break software into **independent modules.**

- Each module should have a **clear responsibility.**

5-Modularity: Software should be a **collection of reusable, self-contained modules.**

- Enables **easy modification, testing, and scalability.**

PRINCIPLES #6 & #7 – SRP & OCP

6-Single Responsibility Principle (SRP): Each module, class, or function should have **only one responsibility**.

- Prevents mixing multiple concerns in a single unit.

7-Open-Closed Principle (OCP): Software should be **open for extension but closed for modification**.

- Encourages using **abstractions and interfaces** for flexibility.

PRINCIPLES #8, #9 & #10 – LSP, ISP & DIP



8-Liskov Substitution Principle (LSP):
Subtypes must be substitutable for base types
without affecting functionality.



**9-Interface Segregation
Principle (ISP):** Clients
should not depend on
unused interfaces.

Use **smaller, more
specific interfaces**.



**10-Dependency Inversion
Principle (DIP):** High-level
modules should depend on
abstractions, not concrete
implementations.

Encourages
**dependency
injection** to
improve flexibility.

BENEFITS OF APPLYING THESE PRINCIPLES



Higher Software Quality: Fewer defects, better performance.



Increased Productivity: Faster development cycles.



Better Collaboration: Shared understanding improves teamwork.



Reduced Technical Debt: Easier to maintain and scale.



Greater Agility: Adaptability to changing requirements.

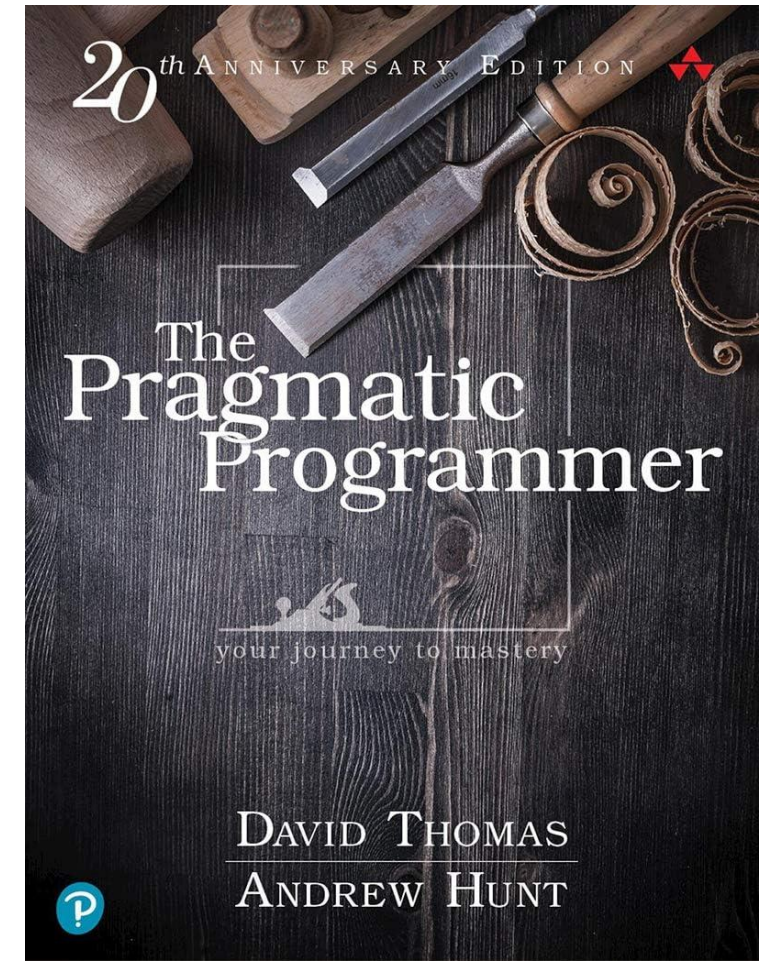
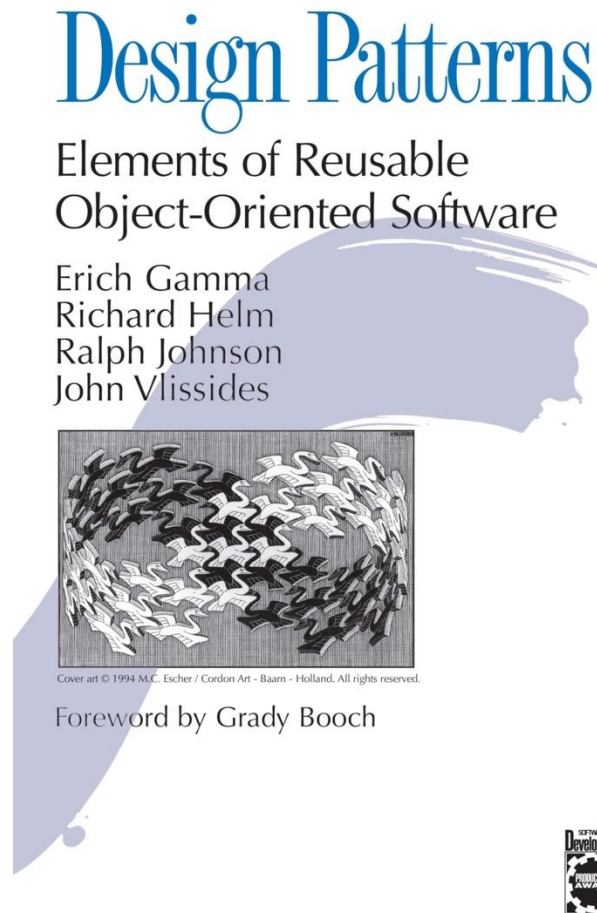
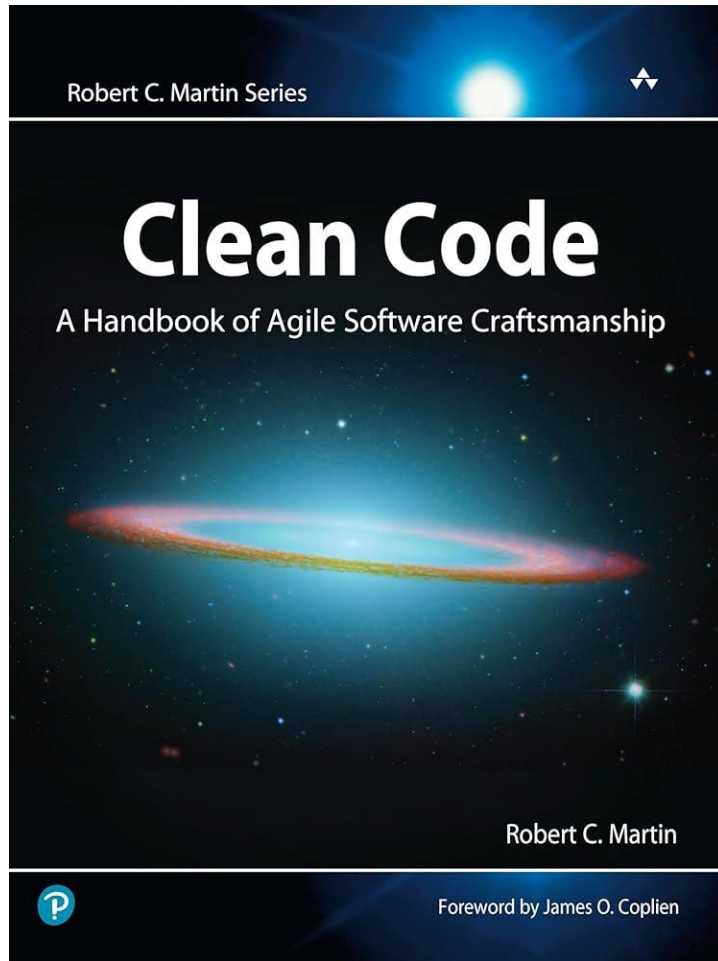


Cost Savings: Minimized rework and optimized resources.

SOLID

LISTENING ACTIVITY

<https://www.youtube.com/watch?v=V3TUEeB0kVW0>



BOOK RECOMMENDATIONS

WORDS OF THE WEEK

1. **Abstraction** – Hides implementation details.
2. **Encapsulation** – Bundles data and methods.
3. **Cohesion** – Degree of module focus.
4. **Coupling** – Dependency between modules.
5. **Scalability** – Handles growth efficiently.
6. **Maintainability** – Easy to modify software.
7. **Reusability** – Use components multiple times.
8. **Modularity** – Divide system into modules.
9. **Robustness** – Handles errors gracefully.
10. **Extensibility** – Allows feature expansion.
11. **DRY (Don't Repeat Yourself)** – Eliminates redundancy.
12. **KISS (Keep It Simple, Stupid)** – Avoids unnecessary complexity.
13. **YAGNI (You Aren't Gonna Need It)** – Prevents over-engineering.
14. **Single Responsibility Principle (SRP)** – One job per module.
15. **Open-Closed Principle (OCP)** – Extend without modifying.
16. **Liskov Substitution Principle (LSP)** – Maintain type compatibility.
17. **Interface Segregation Principle (ISP)** – Small, specific interfaces.
18. **Dependency Inversion Principle (DIP)** – Depend on abstractions.
19. **Agile Development** – Iterative, flexible development.
20. **Technical Debt** – Future code maintenance burden.

PS: Keep a journal where you note these words with their meanings and usages in a sentence.



EOF*

*End of Fun/File