OBJECT-ORIENTED PROGRAMMING IN C++

Feza BUZLUCA

Istanbul Technical University Computer Engineering Department https://akademi.itu.edu.tr/en/buzluca

https://www.buzluca.info

tional License. (CC BY-NC-ND 4.0)



(a) (1999 - 2025 Feza BUZLUCA

INTRODUCTION

Main Objectives of the Course

- To introduce Object-Oriented Programming and Generic Programming
- To show how to use these programming schemes with the C++ programming language to build "good" (high-quality) programs

Need for high-quality design and good programming metho

- · Today, almost every electronic device includes a computer system controlled by software.
- Software plays a vital role in our daily lives.

- Software project costs (especially maintenance costs) are high.
- Maintenance: Changes (requirement changes or bug fixes) and extensions must be made to the software system after it has been delivered to the customer.
- Software errors may cause loss of lives and financial losses.



1999 - 2025 Feza BUZLUCA

Examples of software failures:

- In 2018, a software bug was discovered in the UK's NHS (National Health Service) system that put over 10,000 patients at risk of getting the wrong medication.
- In 2024, a faulty security update from CrowdStrike, a cloud security platform, caused widespread system crashes (blue screens of death) on Windows devices globally. This affected millions of machines across various sectors, including banking, healthcare, and transportation, leading to an estimated \$3 billion in financial losses
- Two Boeing 737 Max jets crashed, one in Indonesia in 2018 and another in Ethiopia in 2019, resulting in a total of 346 fatalities.
- The crashes were caused by flaws in software design and not by the pilots or the airline's performance. Tesla recalled 12,000 cars in 2021 after finding a glitch in its Full-Self Driving beta

A software bug caused vehicles to falsely detect forward collisions, triggering the automatic emergency braking (AEB) system and bringing them to a sudden stop.



(a) ⊕ (b) 1999 - 2025 Feza BUZLUCA

Many grounding 73 after 2nd fatal crass

151

The goal of a software development project:

- To deliver a softw re system that
 - 1. meets the quality needs of different stakeholders (user, developer, customer ...)

 - Functionality
 Performance (speed, accuracy, etc.)
 - Efficiency (processor, memory, network, etc.)
 Reliability (error-free)

 - Security (access control, data protection)
 Maintainability (modify, extend, reuse)
 - 2. is on time
- 3. is within budget. Once the system is operational, the challenges of being on time, on budget, and having the expected quality do not disappear.
- The system must be maintained and developed to meet changing needs and environments

Just writing a code that runs somehow is not sufficient!

· One must consider the quality needs of the system's stakeholders

Some of the

software quality

(c) ⊕ (3) 1999 - 2025 Feza BUZLUCA

Expectations of different stakeholders (Quality needs):

Price

Source: D. Falessi, 6. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design," ACM Computing Surveys, vol. 43, pp. 1-28, Oct. 2011.

Quality characteristics of a software system

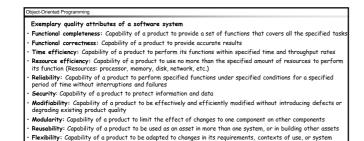
- ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) prepared standards for quality models.

 For definitions of the quality attributes of a software system, refer to the following standards:
- - o ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) Product
 - This standard defines quality characteristics related to the software development team
 - o ISO/IEC 25019: Systems and software Quality Requirements and Evaluation (SQuaRE) Quality-in-

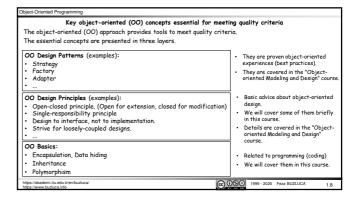
This is the external quality of the system and the impact on stakeholders (customers, direct and indirect users, etc.) in specific contexts of use.

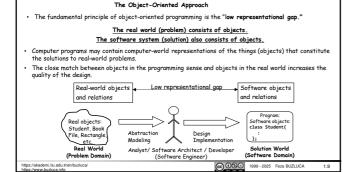
- Details of the quality models are outside the scope of this course
- They will be covered in BLG 468E Object-Oriented Modeling and Design (8th semester, undergrad) and BLG 625 Software Design Quality (graduate).
- This course (OOP) will provide a brief insight into a software system's quality attributes (see 1.7) that must always be considered during software development.

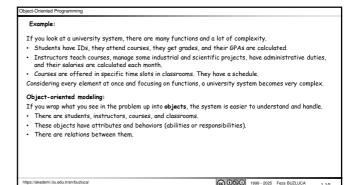
(c) (1999 - 2025 Feza BUZLUCA

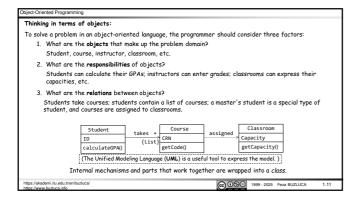


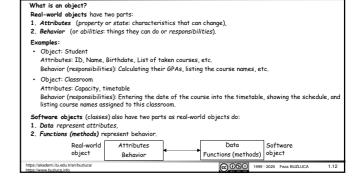
While designing and coding a program (and learning a programming language), these quality attributes must always be considered.

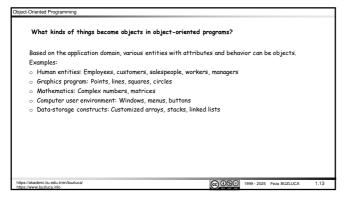


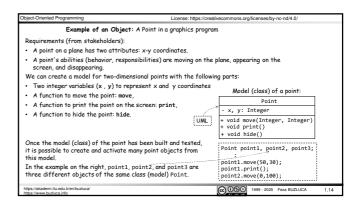


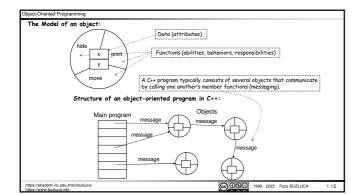


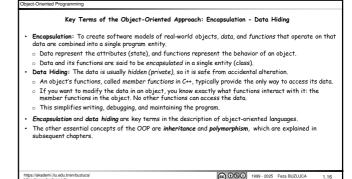












- The object-oriented approach provides tools for the programmer to represent elements in the problem space ($Low\ representational\ gap$).
- We refer to the elements in the problem space ($real\ world$) and their representations in the solution space (program) as "objects."
- OOP allows you to describe the problem in terms of the problem rather than in terms of the computer where the solution will run.
- So, when you read the code describing the solution, you also read words expressing the problem
- Some benefits of the OOP if the techniques are applied properly:
- o Understandability: It is easy to understand a good program. Consequently, it is easy to analyze, modify,
- and improve the program.
- o Reliability: Reducing the possibility of errors
- Extensibility: Reducing the cost of adding new features to the program
- Modifiability: Reducing the effort to adapt an existing system (quicker reaction to changes in the business environment, requirements)
- Reusability: Existing modules can be used in new projects
- Teamwork: Modules can be written by different team members and integrated easily

(C) (D) 1999 - 2025 Feza BUZLUCA

\cdot Programming is fun, but it is related (only) to the implementation phase of software development. Development of quality software is a bigger job, and besides programming skills, other capabilities are also necessary. $\dot{\,}$ This course will cover OO basics: Encapsulation, data hiding, inheritance, and polymorphism Although OO basics are important building blocks, a software architect must also be aware of **design principles** and **software design patterns**, which help in developing high-quality software. See the chess vs. software analogy in the following slides. · Design principles and patterns are covered in another course: Object-Oriented Modeling and Design (8th semester). http://www.ninova.itu.edu.tr/tr/dersler/bilgisavar-bilisim-fakultesi/2097/blg-468e/ (⊕ (⊕ () 1999 - 2025 Feza BUZLUCA

Analogy: Learning to play chess - Learning to design software

Rules and physical requirements of the game, the names of all the pieces, and the way that pieces move

2. Learning principles:

The value of protecting the pieces, the relative value of those pieces, and the strategic value of the

At this point, people can play chess, although they will probably not be outstanding players

At this point, people can become good players.

3. Studying the games of other masters (Patterns):

Buried in those games are **patterns** that must be understood, memorized, and repeatedly applied until they become second nature.

At this point, people can become chess masters

(a) (1999 - 2025 Feza BUZLUCA

Learning to play chess - Learning to design software (cont'd)

1. Learning basics:

The rules of the languages, data structures, algorithms, and OOP basics. At this point, one can write programs, albeit not always very "good" ones.

2. Learning principles:

Object-oriented modeling and design.

Importance of abstraction, information hiding, cohesion, dependency (coupling) management, etc.

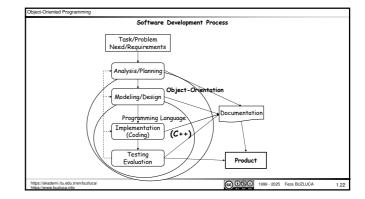
3. Studying the designs of other masters (Patterns):

Deep within those designs are patterns that can be used in other designs.

Those patterns must be understood, learned, and repeatedly applied until they become second nature.

This chess analogy has been borrowed from Douglas ${\it C.}$ Schmidt He states that it is courtesy of Robert Martin.

Imperative/Procedural Programming Technique In an imperative/procedural programming language (technique), each statement (command) in the program tells the computer to do something. The emphasis is on doing things (functions/procedures). A program is divided into **functions** (procedures) and ideally- each function has a clearly-defined purpose and a clearly-defined interface to the other functions in the program. Imperative programming also has some advantages, and it is also possible to write good programs using procedural programming (e.g., C programs). However, object-oriented programming offers programmers many advantages enabling them to write high-quality programs. SHARED (GLOBAL) DATA 1999 - 2025 Feza BUZLUCA



Basic steps of the software development process

Analysis: Gaining a clear understanding of the problem. (Role: Analyst) Understanding requirements. Understanding what the user wants. Requirements may change during (or after) the development of the system!

This about <u>understanding</u> the system (the problem). What should the system do?

Design: Identifying the concepts (entities) and their relations involved in a solution. (Role: Software architect, designer)

Here, our design style is object-oriented. So, entities are objects (classes).

This stage has a strong effect on the quality of the software.

- Implementation (Coding): The solution (model) is expressed in a program. (Role: Developer)
- Coding relates to the programming language. In this course, we will use C++.
- Documentation: Each phase of a software project must be clearly explained. Evaluation: Testing, measurement, performance analysis, quality assessment

The behavior of each object and the whole program for possible cases must be examined. (Role: Quality

Details of the software development process are covered in the "Software Engineering" course.

(C) (D) 1999 - 2025 Feza BUZLUCA

The Unified (Software Development) Process - UP

The Unified Process is a popular iterative software development process for building object-oriented systems. It promotes several best practices.

· Iterative:

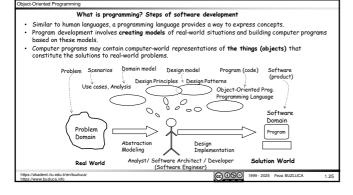
- Development is organized into a series of short, fixed-length (for example, three-week) mini-projects called iterations.
- The outcome of each iteration is a tested, integrated, and executable partial system.
- Each iteration includes requirements analysis, design, implementation, and testing activities
- An iteration step of four weeks, for example.

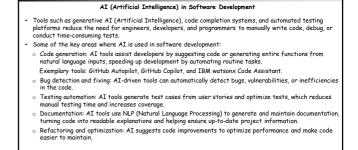
Incremental, evolutionary: The system grows incrementally

· Risk-driven:

o Risky parts first

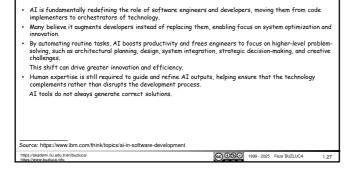
Analysis Analysis grows incrementally Design Implement Design Implement. Test Test Tes: Iterations are fixed in length. For example, 4 w Produgi Product



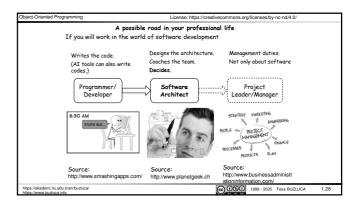


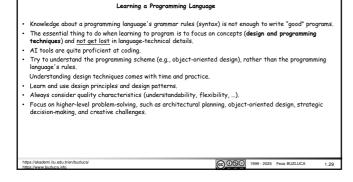
(a) 1999 - 2025 Feza BUZLUCA

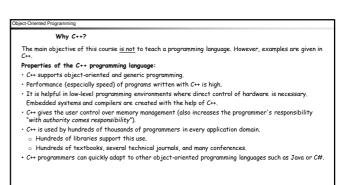
Source: https://www.ibm.com/think/topics/ai-in-software-development



AI in Software Development (cont'd)







(C) 1999 - 2025 Feza BUZLUCA

The applications domain of C++:

- Game (engine) development: Speed and control over hardware are crucial.

 Examples: Fortnite and Unreal Engine
- Graphics and user interface programs
- Systems programming: Operating systems, device drivers. Here, direct manipulation of hardware under real-time constraints is essential.
- High-performance applications: Scientific computing and financial modeling.
- Embedded systems: For example, systems for cars and medical devices.
 It is possible to implement relatively small and efficient programs that can run on limited hardware resources.

- hardware resources.

 Examples of applications written in C++:
 Apple's Mac OS X,
 Adobe Systems,
 Backend services of Facebook,
 Google's Chrome browser,
 Microsoft Windows operating systems, MS Office, Visual Studio
 Mozilla Firefox, Thunderbird,
 MySQL
 are written in part or in their entirety with C++.

@ 0 0 1999 - 2025 Feza BUZLUCA 1.31



C++ Standards
C++ is standardized by the working group WG 21 of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

 $Official: ISO/IEC\ JTC1\ (Joint\ Technical\ Committee\ 1)\ /\ SC22\ (Subcommittee\ 22)\ /\ W621\ (Working\ Group\ 21):\ JTC1/SC22/W621$

- The current C++ standard is C++23, i.e., ISO/IEC 14882:2024.

 It was finalized in 2023 and published in 2024.

 The next planned standard is C++26.

- You can get the standard in ÎTÜ campus from the website of the British Standards Online: http://bsol.bsigroup.com/
- Information about C++ standards: https://isocpp.org/std/the-standard
- Be aware of programming standards and use compilers that support the current one. https://en.cppreference.com/w/cpp/compiler_support
 For example, you can use MS Visual Studio, Clang, or GCC.

@ 0 ⊕ © 1999 - 2025 Feza BUZLUCA