OOP Coursework

Coursework Structure

Create a project applying Object-Oriented Programming (OOP) principles and compose a formal paper explaining its structure, function, design, and implementation. Scope: cover functional requirements with code, for the report - write MD format file, formatting MD file - Markdown language guidelines.

Steps to complete this coursework:

1. Select a topic (see Select topic)
2. Write code - implement Functional Requirements (see Functional requirements)
3. Write a report paper (look again at Functional requirements)
4. Check if you did everything in the Evaluation system (see Evaluation system)

Select a Topic

Each student selects their topic. Coursework topic examples to pick from or use as example to think of your own:

1. Management Systems:
    - Library
    - Movie theatre
    - Flight ticket reservation
    - Hospital
    - Inventory
    - Hotel
    - School
    - Restaurant
2. File Manager:
    - Perform file operations like insert, delete, search, copy, move, rename, etc.
    - Implement undo and redo functionality to revert or repeat file operations performed by the user.
    - Ensure proper error handling and validation for various scenarios (e.g., invalid file paths, insufficient permissions, etc.).
    - Allow users to navigate through directories and view files in different directories.
3. Games:
    - Battleship game
    - Tic-Tac-Toe
    - Chess
    - Checkers
    - Sudoku Solver
    - Snake
4. App Integration:
    - Develop an application integrating external APIs, such as the one available on RapidAPI (https://rapidapi.com/hub).

5. Finance Tracking:
    - Create a finance tracking application
    - Track expenses
    - Add/remove expenses
    - Print expense history
    - Save expense history to file
    - Set daily/weekly/monthly deposit limits
    - Handle group deposits for multiple users
6. Birthday Reminder:
    - Build a program to remind birthdays
    - Add/remove birthdays
    - Print birthday reminders
    - Save birthdays to file
    - Send notifications on the day of the birthday
    - Support multiple users
7. Dungeons and Dragons (DND) Helper:
    - Develop a digital character sheet creator
    - Create/edit characters with name, class, history
    - Edit statistics (STR, DEX, CON, INT, WIS, CHA)
    - Set up health, inventory, and abilities
8. Smart Calculator:
    - Build a Python program acting as a smart calculator supporting basic operations.
9. Stack Class:
    - Construct a Python class for Last-In, First-Out (LIFO) data structure.
    - Implement methods for pushing, popping, and peeking elements.
10. Sort Class and Strategies:
    - Create a Python Sort class with generic sorting algorithms.
    - Extend with subclasses for BubbleSort, MergeSort, QuickSort, etc.
11. Graph Class:
    - Develop a Python class for graph structures.
    - Create a WeightedGraph class inheriting from Graph, with weights on edges.
12. Converter System:
    - Build a Python system for converting between Roman and Decimal numbers.

Note: topics used in laboratory works cannot be used :)

Topic registration

- Each student should register their chosen topic through Moodle **until April 14th**.
- Each topic after submission must be confirmed with the laboratory works lecturer.
- A maximum of three students may select the same topic for their coursework to ensure a diverse and varied range of topics within the course.

Requirements

Requirement	Explanation
Use Git and upload your work to Github	The program, as well as the Markdown format file, should be uploaded to Github into one repository.
4 OOP pillars, their meaning and usage (in code and overall)	The program should implement all 4 object-oriented programming pillars: ● Polymorphism ● Abstraction ● Inheritance ● Encapsulation Each of them should be described in the report, explaining what it is, how it works, and how it was used in code (tip: use code snippets/screenshots of your program).
Use at least 2 design patterns	Choose 2 of the patterns discussed in lectures/lab. sessions or one of the following (for those who want a challenge - you can select something outside this course): ● Singleton ● Factory Method ● Abstract Factory ● Builder ● Prototype ● Adapter ● Composite ● Decorator Used pattern should fit the program and be explained how it works and why it is most suitable compared to others *E. g.:* ● *“In scenarios where we need different types of x objects with varying behaviours and characteristics, the Factory Pattern provides a more flexible approach.”* ● *“Singleton Pattern could be suitable if we want to ensure that there is only one central point responsible for managing x object creation and behaviour.”*
Reading from file & writing to file	Choose a file type (e.g., TXT, CSV) and implement functions to import and export data: saving results, state, output, or progress of the program. If you want, feel free to use a database, file, API, etc.

Requirement	Explanation
Testing	Core functionality should be covered with unit tests (using `unittest` framework).
Code style	Program must be written in Python. Program code should follow PEP8 style guidelines.
Report structure	Coursework paper (report) should consist of 4 parts: 1. Introduction a. What is your application? b. How to run the program? c. How to use the program? 2. Body/Analysis a. Explain how the program covers (implements) functional requirements 3. Results and Summary a. See "Results" functional requirement b. See "Conclusions" functional requirement c. How it would be possible to extend your application? 4. **Optional**: Resources, references list. Report can be written either in **English or Lithuanian**. Report should be written using best practices of Markdown file style (see Documentation links, good practices, resources)
Introduction	Describe the goal of the coursework. Describe your topic. What is your application? How to run the program? How to use the program?
Body/Analysis	Analysis and explanation of the program's implementation. Describes how the program meets the defined objectives and functional requirements. Use code snippets of your program to cover requirements and explain your implementation.
Results	3-5 sentences (separate bullet points) about the results, which can include challenges faced during the implementation.
Conclusions	Short summary of the key findings and outcomes of the coursework. What has this work achieved? What is the result of your work (program)? What are the future prospects of your program?

Documentation links, good practices, resources

- About GitHub
- Github tutorial
- Git command cheat sheet (1)
- Git command cheat sheet (2)
- Pro Git Book for extra Git knowledge
- PEP8
- Markdown syntax (1)
- Markdown syntax (2)
- Unit test framework
- How to write unit tests (1)
- How to write unit tests (2)
- How to write unit tests (3)
- PEP8 examples
- Clean Code in Python
- SOLID in Python *(able to view only with a LinkedIn account)*
- Design patterns
- Python application layout
- File import in Python
- CSV (Comma Separated Values) format
- PyCharm Code Editor (IDE)

Evaluation System

No.	Deadlines	Requirement	Points
0.	2024-04-14	Select your topic	-
1.	From the start of your project	Use Git and upload your work to GitHub	70% (2.1 p.)
		Code style	
2.	2024-05-05	4 OOP pillars, their meaning, and usage (in code and overall)	
3.		Use at least 2 design patterns	
4.		Reading from file & writing to file	
5.		Testing	
6.	2024-05-12	Report requirements	30% (0.9 p.)
7.		Introduction	
8.		Body/Analysis	
9.		Results	
10.		Conclusions	
		Total point sum:	**3**

- Minimum amount of points that need to be collected is **2 points**.
- Both components (the program and the report) must be fully completed and submitted on GitHub before the deadline. In case of delay in submission, the corresponding part's **grade will be decreased by one-fourth of its total value for each week**.
- Final deadlines for uploading & presenting your work:

Lab. session day of your group	Date
Monday	2024-05-20
Wednesday	2024-05-22
Friday	2024-05-17

- Both components of the coursework (the program and the report) are mandatory and must be completed; omitting either one is not permissible.
- Students are encouraged to regularly share their progress before the deadline for periodic consultation. Failure to do so may lead to unintentionally deviating significantly from the intended scope of the assignment.

- On the day of presenting or defending your coursework, students are required to upload a compressed folder in .ZIP format containing the project exported from GitHub.