MINISTRY OF EDUCATION
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

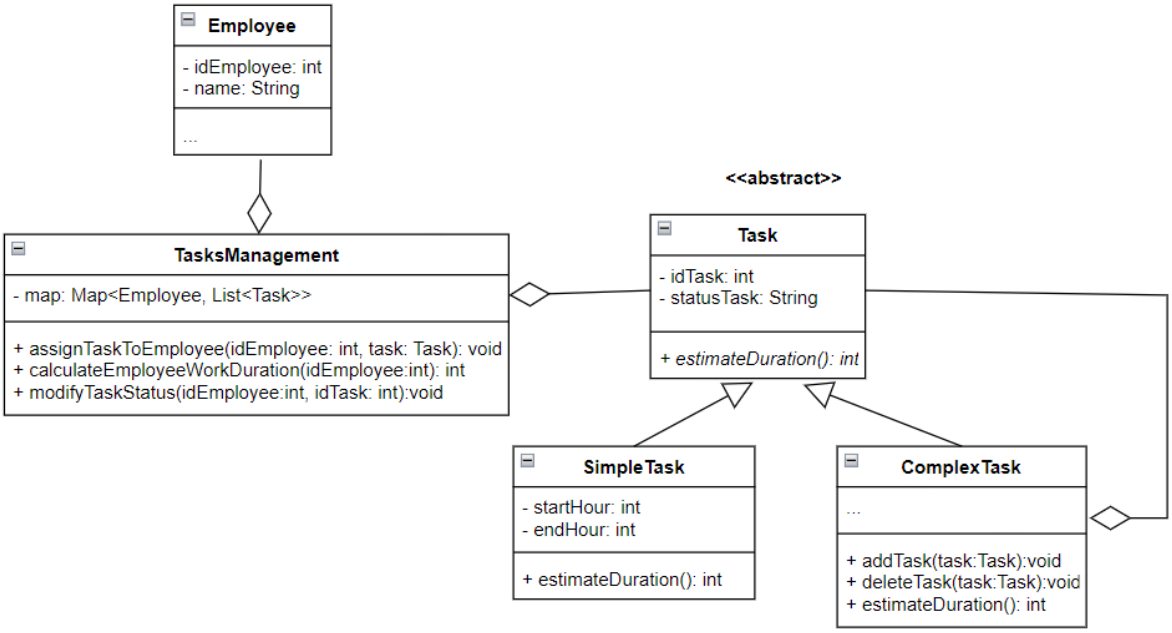
FUNDAMENTAL PROGRAMMING TECHNIQUES

ASSIGNMENT 1

TASK MANAGEMENT

1. Requirements

Design and implement an application for managing the tasks assigned to the employees of a software company, starting from the class diagram below.



Implement the following classes:

- *Employee*
- *Task* - a sealed abstract class with an abstract method, *estimateDuration*()
- *SimpleTask* –overrides the *estimateDuration*() method to return the estimated duration of a task based on the start and end hours
- ComplexTask - can be composed of simple and/or complex tasks
- *TasksManagement* with the methods: (i) *assignTaskToEmployee*, which assigns a task to an employee; (ii) *calculateEmployeeWorkDuration*, which estimates the work duration of an employee based on the completed tasks (i.e., *statusTask* = “Completed”) assigned to them; (iii) *modifyTaskStatus*, which modifies the status of a task assigned to an employee to “Completed” or “Uncompleted” depending on the case.
- *Utility* with the following methods: (i) a method that filters all employees who have a work duration greater than 40 hours, sorts them in ascending order according to the work duration, and displays their names; (ii) a method that calculates, for each employee, the number of completed and uncompleted tasks and returns a map where the key is the employee's name and the value is a Map<String, Integer>, where the key represents the possible status of the tasks (i.e., "Completed" or “Uncompleted”), and the value is the number of tasks in that category.

A project manager of the software company can interact with the application through a graphical user interface and can perform the following operations:

- Add employees, add task (simple, or complex), assign task to employee
- View employees and their tasks (including the estimation of duration)

- Modify the status of a task assigned to an employee
- View the statistics provided by the methods of the Utility class

All data will be persisted using the serialization technique.

NOTE: Implement constructors, get/set methods, and add new classes, attributes and methods in addition to those specified in the diagram, such that the functionalities specified in the requirements can be implemented.

2. Deliverables

- **Source code files** – will be uploaded on the personal **Gitlab** account created according to the instructions in the **Laboratory Resources** document, and following the steps:
    - Create a private repository on ***Gitlab*** named according to the following template:
      PT2025_Group _LastName_FirstName_Assignment_1
    - Push the source code and the documentation (**!!!not an archive with the code!!!**).
    - Share the repository with the user **utcn_dsrl**
- The **draw.io file** with the UML use case diagram, package diagram and class diagram – will be uploaded in the repository with the source code files.

3. Evaluation

The assignment will be graded as follows:

Requirement	Grading
• Use an object-oriented programming design. • Use the layered architectural pattern. • Implement the classes: *Employee*, *Task*, *SimpleTask*, *ComplexTask* • Implement a graphical user interface using Java Swing, such that the project manager can: add and view employees and tasks (simple, or complex) • Implement serialization for data persistence • Use *foreach* instead of *for(int i=0…)*. • Implement classes with maximum 300 lines (except the UI classes) and methods with maximum 30 lines. • Use the Java naming conventions (see link). • Correct UML diagrams.	5 p
Implement the method *assignTaskToEmployee* and the corresponding functionality in the GUI	1
Implement the method *calculateEmployeeWorkDuration* and the corresponding functionality in the GUI	1
Implement the method *modifyTaskStatus* and the corresponding functionality in the GUI	1
Implement the filtering method from the *Utility* class and the corresponding functionality in the GUI	1
Implement the method that calculates, for each employee, the number of completed and uncompleted tasks and the corresponding functionality in the GUI	1

4. Bibliography

- **Swing:** https://docs.oracle.com/javase/tutorial/uiswing/index.html
- **Java serialization**: https://www.baeldung.com/java-serialization
- **Composite design pattern**: https://www.baeldung.com/java-composite-pattern
- **Java sealed classes:** https://www.baeldung.com/java-sealed-classes-interfaces