UNIVERSITATEA TEHNICĂ
A MOLDOVEI

# OOP

*Laboratory work: #2*

Inheritance & Polymorphism

2024

HOMEWORK N°

# 2

## Assignment

**Task:**

**Create an object-oriented program that monitors and detects changes in documents within a designated folder.**

In this laboratory, the task is to develop a simplified version of git functionality that detects changes within a specific folder. Here's an example of an expected output of the program:

```
Created Snapshot at: 2023-10-11T09:01:28.806114900
test.txt - No Change
image.png - Changed
python_script.py - Changed
>
```

Figure 1.1: Program output example

As one can notice each file has a different extension **.txt**, **.png** and **.py**. The user is informed of the files changed status since the last snapshot time. From this output one can gather:

***test.txt has experienced no change since the snapshot time of 2023-10-11, 09:01:28, while image.png and python_script.py did change since the snapshot.***

**Task:** VERY IMPORTANT!

1. **No third party libraries are allowed, you can use only the ones available in your programming language.**
2. **You are free to create as many classes as you need to achieve a well structured working system.**
3. **Limitation of concepts: You are required to structure you're program using the concepts of Inheritance and Polymorphism (Runtime and Compile time Polymorphism is a must)**
4. **USING GIT: You have to commit each significant change in your program. Failing doing so results in a 2 point deduction, plus writing a 2 page report on "Git usefulness". If there's no report provided, laboratory isn't accepted.**

# Base Laboratory (8 Grade)

**Task:**

Build a program loop, an interactive command line for the user to monitor changes in a folder. The folder location is hardcoded, meaning it's constant, and up to the implementer to specify it's location. The 3 mandatory actions a user can perform are:

**Actions:**

1. **commit** - Simply update the snapshot time (can be a variable) to the current time. It's functionality is to emulate change detection since the previous snapshot. When commit is invoked, the snapshot is updated and the state is again clean (no changes).
2. **info <filename>** - prints general information about the file. *filename* can be treated as an unique identifier, since even the OS doesn't allow same filename + extension in the same folder.
   - **all files** - file name, pretty print of the file extension, created and updated date and time.
   - **image files (png, jpg)** - image size (ex. 1024x860).
   - **text files (txt)** - line count, word count and character count.
   - **program files (.py, .java)** - line count, class count, method count (files from previous laboratory can be used).
3. **status** - shows the output presented in 1.1. When calling status an iteration occurs through all the files stored in program memory and prints if they were changed during snapshot time and current time.

(Action naming isn't mandatory to follow. The choice is yours.)

# Improved Laboratory (9 Grade)

```
test.txt - No Change
test.txt - Changed
image.png - New File
python_script.py - Deleted
> commit
```

Figure 1.2: File add/delete detection output example

**Task:**

For an improved laboratory, the detection should occur also when files are added or deleted in the specified folder. Follow the example 1.2, image.png is a new file, while python_script.py was deleted since the last snapshot.

# Working System (10 Grade)

The file change detection system should happen in almost real-time. As soon as a change happens, the console should print the detection message, and what exactly has changed, without the user to run an action.

**Task:**

1. **Schedule** - every 5 seconds, the program should run it's detection flow, and print each change to the console. If no change has occured - then print nothing.
2. **Console availability** - the user should still be able to call actions from the console, while the scheduled detection program runs. The scheduler should not impede the program's availability in any way. (Missprints due to multiple writes (console + scheduler) are **totally okay**, and also other minor artifacts that may appear due to this fact).

# Earning Extra Point(s)/Penalties:

*You may earn an extra point or lose if you don't:

- Keep a clean project structure +/-1 point;
- Not following language coding convention -1 point ;

# Resources:

Java Inheritance: W3 - Java Inheritance.

Java Inheritance: GFG - Java Inheritance.

Java Polymorphism: W3 - Java Polymorphism.

Java Polymorphism: GFG - Java Polymorphism.

Runtime and Compile time Polymorphism - Types of Polymorphism in Java.

Program vs Process vs Thread - important (especially for mark 10).

Java Threads: How to start/implement - important (especially for mark 10).

Java Directories Library - important for the laboratory as a whole.

Git Flow (1) - get more comfortable with the concept of Git and working with branches.

Git Flow (2) - get more comfortable with the concept of Git and working with branches.

Git exercises - perfect your Git experience using these exercises. Achieve git fluency by googling new encountered commands and practice using them withing this laboratory also.

Google - use your search engine to get to know more about your programming language, language features and libraries such as **File** class in Java.