

**Homework:**

Implement **ONE** of the following design patterns by the following specifications (don’t forget to read the requirements for all tasks in the end):

1. **Factory method**.

Implement an application that:

- Gets a full folder path as its input value

- Collects a list of all files in this folder (you may implement it as recursive or not, it is up to you), their properties (like size/creation time/modification time/access type/archive/read-only etc.). Select at least 3 properties.

- Prints a “report” about the folder content in the following formats (select at least 2): txt, csv, xml, json.

Use factory method.

1. **Observer.**

Implement a console application that:

- Prints “hello world” (or any other text) in the console.

- watches the mouse moving:

When the cursor moves to the console window area, the text color in the console changes and the current time is written into the log file with the tag “Enter”.

When the cursor leaves the console window area, the text color changes to the original and the current time is written into the log file with the tag “Leave”.

You may hardcode the path to the log file or put it in the same folder as exe file.

Use observer.

1. **Adapter**
2. Implement a class that provides search on the file system. It should get start path and search string as input data, then it performs recursive file traversing inside this path. It should return only files that contain search string in the name. It should also return file creation time, size and the name + full file path as well.
3. Now don’t change this class. We consider it as legacy code, that we cannot change or recompile.
4. Conditions are changed: the input string has another encoding (let it be UTF-8 if it was UTF-16 before and vice versa), this is also required for file name and path in output. And the creation time should be UTC (if you had local time) or local time + time zone if you had UTC before. Size should be also in Kb instead of bytes for example.
5. Implement new conditions using the Adapter to the old code.
6. Implement some client code that shows usage of original search and then of new search. You can take a screenshot with the results or copy them to file from console. It’s not required to write code that saves results to file.

Requirements for all tasks:

You may use the standard library or/and boost

Each task should meet the following requirements:

1. Use smart pointers for memory management. Use self-written guards or stl/boost solutions for other resources.
2. Use stl/boost containers: vector/map/list, etc. if you need containers.
3. Provide exceptions error handling with strong safety guarantee and don’t forget about the best practices.
4. Think about the application architecture: about interfaces and class responsibilities (don’t forget the single-responsibility principle). As we discuss OOP and design patterns, the architecture is an important part of the homework. You should show that you understand how the pattern works.
5. Organize the classes by files. This is also an important part of decomposition. You may put several classes in the same file if this meets the application logic, but don’t put everything in main.cpp.

**Evaluation:**

Everyone has max points (100) at the beginning. And each mistake costs points.

**What is a mistake**:

- you don’t understand clearly how the pattern works.

- Incorrect error handling or no error handling. Throw exception where your logic has an exceptional situation (in most cases std::runtime\_error is enough). Don’t forget to catch all thrown exceptions and show the user the appropriate error. Remember that stl functions also can throw exceptions. Also, take into account that some functions can return a status instead of throwing an exception. Read the documentation.

- Resource management problems: leaks, incorrect smart pointers usage. Read the documentation about non-memory resources and how to free them.

- Improper decomposition, mixed responsibilities, not enough encapsulation, redundant fields or methods.

- Not implemented requirements (form the “minimal list”)

- Unclear naming like ex1 instead of expectedResult.

- Maybe something else which I cannot imagine.