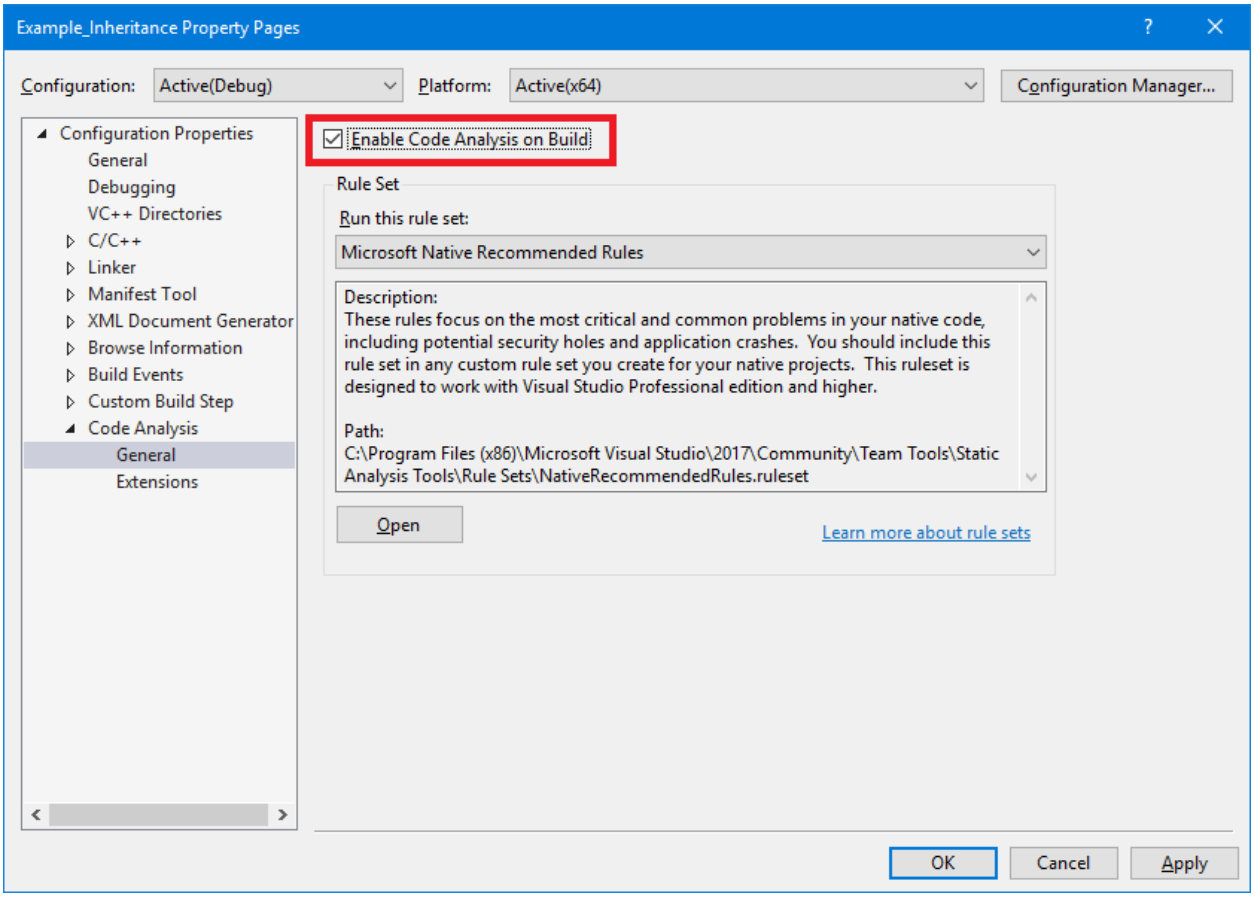
LABORATORY 8-9

This is an assignment for 2 weeks. For week 8, you must implement at least requirements 1, 2 and 3. For week 9, all requirements must be implemented.

To avoid possible defects in your C++ programs, use a static analysis tool for your source code. In Visual Studio, to enable the Code Analysis tool, go to Project *Properties -> Code Analysis -> General* and enable code analysis on build. Then, after a build, make sure you have no code analysis warnings.



If you are using a different IDE, please choose a code analysis tool and use it. A list of such tools for C/C++ can be found at: https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis#C,_C++.

For your assignment from Lab5-6, add the following features:

1. Replace your DynamicVector template with the STL vector. Use STL algorithms wherever possible in your application (e.g. in your filter function you could use **copy_if**, **count_if**). Replace all your for loops either with STL algorithms, or with the ranged-based for loop (C++11).

2. Store your data in a text file. When the program starts, the entities in the database (file) will be read. The modifications made during the execution of the application should be stored in the file. For this feature, **use the iostream library**. Create insertion and extraction operators for your entities and use these when reading/writing to files or console.

3. Use exceptions to signal errors:
   - from the repository;
   - validation errors – validate your entities using Validator classes;
   - create your own exception classes.

   Validate your input data.

Observation: Points 4 and 5 should be solved using inheritance and polymorphism.

4. Depending on your assignment, store your:
   - adoption list
   - movie watch list
   - shopping basket
   - tutorial watch list

   in a file. When the application starts, the user should choose the type of file (CSV or HTML). Depending on this type, the application should save the list in the correct format.

   Indications
   - the CSV file will contain each entity on one line and the attributes will be separated by comma (,)

     E.g. (for Song entities):

     Ed Sheeran,I see fire,4:54, https://www.youtube.com/watch?v=2fngvQS_PmQ
     The Proclaimers,I would walk 500 miles,3:37,https://www.youtube.com/watch?v=otXGqU4LBEI

   - in the HTML file you will have a table, in which each row holds the data of one entity. The columns of the table will contain the names of the data attributes.

     E.g. (for Song entities) – the text written in magenta offers comments (these do NOT belong to the html file).

```
<!DOCTYPE html>                          - write this exactly as it is here
<html>                                   - write this exactly as it is here
    <head>                               - write this exactly as it is here
        <title>Playlist</title>          - replace “Playlist” with the title for your HTML
    </head>                              - write this exactly as it is here (close head tag)
    <body>                               - write this exactly as it is here
        <table border="1">               - write this exactly as it is here
```

```
<tr>                                    - tr = table row; 1 row for each entity
    <td>Artist</td>                     - td = table data; 1 td for each attribute of the entity
    <td>Title</td>
    <td>Duration</td>
    <td>Youtube link</td>
</tr>                                   - write this exactly as it is here (close tr tag)
<tr>
    <td>Ed Sheeran</td>
    <td>I see fire</td>
    <td>4:54</td>
    <td><a href =
"https://www.youtube.com/watch?v=2fngvQS_PmQ">Link</a></td>
    - a href makes the text “Link” clickable and directs us towards the link written after “=”
</tr>
<tr>
    <td>The Proclaimers</td>
    <td>I would walk 500 miles</td>
    <td>3:37</td>
    <td><a href =
"https://www.youtube.com/watch?v=otXGqU4LBEI">Link</a></td>
</tr>                                   - write this exactly as it is here (close tr tag)
</table>                                - write this exactly as it is here (close table tag)
</body>                                 - write this exactly as it is here (close body tag)
</html>                                 - write this exactly as it is here (close html tag)
```

The file above, if opened with a browser, looks like this:

Artist	Title	Duration	Youtube link
Ed Sheeran	I see fire	4:54	Link
The Proclaimers	I would walk 500 miles	3:37	Link

5. Add a new command, allowing the user to see the:
   - adoption list
   - movie watch list
   - shopping basket
   - tutorial watch list

   Displaying the list means opening the saved file (csv or html) with the correct application:
   - CSV file – with Notepad, Notepad++, Microsoft Excel or OpenOffice Calc
   - HTML file – with a browser

6. Create a UML diagram for your entire application. For this, you can use any tool that you like (StarUML is an example of open source software for UML). Do **not** draw the diagram by hand.

7. After delivering this entire assignment (both iterations), you must send an **.zip** archive containing your source code files (without any .obj or executable files) to the email address: commit@scs.ubbcluj.ro.

ADDITIONAL REQUIREMENT - BONUS POSSIBILITY (0.2 P)

In addition to the file-based implementation for the repository, implement an SQL-backed repository. For this, use *inheritance and polymorphism*. Choose any type of database management system (e.g. MySQL, SQLite, PostgreSQL). To receive the bonus, the requirements must be implemented correctly, **by week 9** and the application must function properly.