COMP26020 Part 1– Assignment 2: Library Management Software in C++

The goal of this assignment is to implement a set of C++ classes that aim to be used in a library management software. The library holds documents of different types (novels, comics and magazines) with various attributes (title, author, etc.) and users can borrow and return documents.

- `library.h` has the different classes definitions including inheritance relationships as well as the prototypes for all classes' methods.
- `library.cpp` contains the implementations of the classes.
- `test-suite.cpp`, `catch.h`, and `catch.cpp` implement a test suite that checks the correctness of the library code.

We have already implemented everything! The code is 100% functional. Check your gitlab repo or the Blackboard entry for this assignment.

Wait. What's the point of this assignment then?

Well, the code was written by someone who only watched some of videos for the first week of C++ and then gave up. The code is more C than C++, it does not use the standard library, and it does not follow any of the C++ design approaches. This is not good C++ code.

Your **actual goal** is to rewrite **`library.h`** and **`library.cpp`** applying the lessons learnt in the lectures, including:

- RAII
- new/delete considered harmful
- C-arrays, C-strings considered redundant
- C++ standard library containers, utilities, and algorithms
- Improved type safety
- Using linters, static analyzers, and the C++ Core Guidelines to uncover errors and bad coding patterns

You are free to a) add member functions, b) change the implementation of existing member functions, or c) change private declarations for variables or functions in `library.cpp` and `library.h`.

You should not change any other file. Also, you should **not change the interface of any public member functions** of any of the classes. The only allowed interface change is changing the type of a function argument or the return value to an equivalent one. E.g., an int return value that is meant to be used as a true/false value can be replaced by a bool. A C-string (char *) argument or return value can almost always be changed into an std::string.

Test Suite

To test your implementation you are given a basic test suite in the form of a C++ file, `test-suite.cpp`. It's a C++ program (i.e. it contains a main function) that includes `library.h`. It instantiates/manipulates objects from all the library classes and performs sanity checks. This program uses the *Catch*[1] test framework and requires two additional source files to be compiled: `catch.cpp` and `catch.h`. You can compile the program manually:

```
$ g++ test-suite.cpp library.cpp catch.cpp -o test-suite
```

and execute the test suite with:

```
$ ./test-suite
```

1 https://github.com/catchorg/Catch2

You are not supposed to change the test suite and you don't need to understand the code fully. But if you get failed tests, it might be useful to check the code that is associated with the failed test in order to understand what went wrong. Overall, the suite is divided into tests cases enclosed into `TEST_CASE() { ... }` statements. A test case will fail when one of the `REQUIRE( <condition> )` or `CHECK( <condition> )` statements it contains fails, i.e. when `condition` evaluates to false.

To complete the assignment you do not need to understand the content of `catch.h` and `catch.cpp`.

The test suite is not comprehensive. Passing all the tests means that a good chunk of the code behaves correctly, but it does not mean everything is perfect. When marking, an extended test suite will be used that might uncover problems with the code. So, **make sure you do not change the public behaviour of the library code, not even in minor ways!**

Make

For your convenience, we provide a Makefile with a few convenient targets.

To build the whole test suite, while checking the library code for warnings:

```
$ make
```

To run the gcc static analyzer (if you have a g++ >= 10.0)

```
$ make analyze
```

To run the C++ core guidelines checker and parts of the clang static analyzer:

```
$ make guidelines
```

You can change the three variables at the top of the Makefile to match your setup. CXX should point to your C++ compiler. CXX_STANDARD is the C++ standard version used for compiling the code. Default is 17, change it to 14, if you are using older compilers. BUILTIN_TIDY chooses which clang-tidy to use: 1 means use the pre-packaged one shipped with the code which works on the lab workstations, 0 means use the default system clang-tidy (if it exists)

Deliverables, Submission & Deadline

There are two deliverables: the completed `library.cpp` and `library.h` files. The submission is made through the CS Department's Gitlab. You should have a fork of the repository named "*26020-lab2-S-CPlusPlus_<your username>*". **Make sure you push to that precise repository and not another one**, failure to do so may result in the loss of some/all points. Submit your deliverables by pushing the corresponding files on the master branch and creating a tag named **`lab2-submission`** to indicate that the submission is ready to be marked.

The deadline for this assignment is 02/12/2022 6pm UTC.

Marking Scheme

The exercise will be marked out of 10, using the following marking scheme:

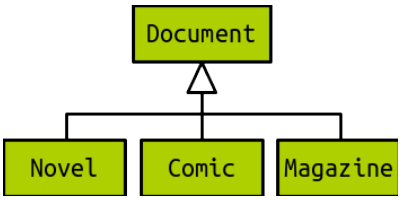
- The program is functional, and passes the basic test suite /1
- The program passes all extended tests /1
- The code takes advantage of C++ capabilities to make the code clearer and more concise /1
- The library code follows the RAII principle /1
- The library code uses Standard Library containers and data types whenever appropriate /2
- The library code uses Standard library algorithms whenever appropriate /1
- The library code does not produce significant warnings when compiled with `-Wall, -Wextra`, or the static analyzer and it does not suffer from memory errors /1
- The library code follows the C++ Core Guidelines (at least the ones that we discussed in the lectures) /1
- The code is clear, well commented, and follows good practices /1

Appendix

Class Hierarchy

The `library.h` header defines 5 classes which are briefly presented below. Note that more detailed information about the methods is present in the header's source code in the form of comments.

- `Document` is an abstract class defining attributes and methods common to all the documents that can be held in the library. Attributes include the document's *title*, *year of release*, and *quantity* held in the library. It defines various methods for printing the document's information on the standard output, getting its concrete type (novel/comic/magazine), and various getters/setters including methods for borrowing/returning the document from/to the library.
- `Novel`, `Comic` and `Magazine` represent the concrete types of documents. Each inherites from `Document` as depicted on the figure on the right. They differ slightly in attributes: a novel and a comic have an *author*, comics and magazines have an *issue number*, and a magazines do not have an author. Each class also defines the relevant getters/setters.



Document class hierarchy

- The last class, `Library`, represents the library i.e. a collection of documents. The documents are held in an array of Document pointers. The library class defines various methods for actions such as adding, removing, searching, borrowing, returning documents, printing the library content on the standard output or dumping it in a CSV file.

Documents/Library Printing and CSV Output Formats

The `print()` method, when called on a novel, prints on the standard output the novel's attributed in this format:

```
Novel, title: Monstrous Regiment, author: Terry Pratchett, year: 2003, quantity: 1
```

For a comic:

```
Comic, title: Watchmen, author: Alan Moore, issue: 1, year: 1986, quantity: 10
```

And for a magazine:

```
Magazine, title: The New Yorker, issue: 1, year: 1925, quantity: 20
```

The `print()` method called on a library containing these 3 documents produces:

```
Novel, title: Monstrous Regiment, author: Terry Pratchett, year: 2003, quantity: 1
Comic, title: Watchmen, author: Alan Moore, issue: 1, year: 1986, quantity: 10
Magazine, title: The New Yorker, issue: 1, year: 1925, quantity: 20
```

The `dumpCSV()` method called on the same library use low level file I/O functions (`open`, etc.) to produce a file with the following format:

```
novel,Monstrous Regiment,Terry Pratchett,,2003,1
comic,Watchmen,Alan Moore,1,1986,10
magazine,The New Yorker,,1,1925,20
```