1. Purpose

The purpose of the QONTAINER project is the development in C ++ / Qt of a manager of an object container of a type hierarchy using a graphical user interface (GUI).

1. Template Container<T>

Define a Container <T> class template whose objects represent a container of T-type objects. The Container <T> template must provide at least minimum features of:

(1) insertion (2) removal (3) search. There is a single constraint to be respected in the development of the Container <T> class template: the use of STL / Qt containers (or other libraries) is not allowed. Choose the implementation of Container <T> considered most appropriate to use the container as a data structure used by the GUI of the QONTAINER manager.

1. Type Hierarchy

Define a G hierarchy of at least three instantiable types and having a common basic supertype B (abstract or instantiable). For example, the type abstract LibraryItem can be a supertype of Book, Magazine, Daily, in a hierarchy that models objects to be cataloged

in a library. The types of the G hierarchy must be equipped with appropriate public interfaces. Use your imagination and be inspired by your own interests in choosing the reality to be modeled using this type hierarchy. Of course, even more complex hierarchies are allowed which contain more than three types.

Consider the opportunity for design purposes to define a DeepPtr <T> class template of polymorphic "smart " pointers to type T. Then DeepPtr <T> should implement an automatic management of "deep" and polymorphic memory of T pointers requiring that T be

a type that supports cloning and polymorphic destroys it.

1. QT GUI

Develop a GUI in Qt that allows the user of QONTAINER a simple management of a container of polymorphic objects of the hierarchy G. For example, the management of the catalog of items available in a library. It is therefore a matter of developing a GUI for the management of a container C of polymorphic (possibly smart) pointers to the supertype A of the hierarchy G, which as mandatory requirements allows:

(1) Insertion of G objects in C. Example: insertion of a new book in the library catalog.

(2) Removal of objects from C that satisfy certain characteristics. Example: removing all the newspapers of a certain year from the catalog of the library.

(3) Searches for G in C objects. Example: search for all the books by the author "Bjarne Stroustrup". In particular, the searches could be based on some public characteristics of the types of G, for example, searches for: year of publication of any

LibraryItem, author of a book, publisher of a magazine, date of a newspaper.

(4) Modifications of G objects in C. Example: modification of the cost of the loan for a particular book.

(5) Load and save of container C.

(5.1) (LOAD) Loading from file of the objects that populate the container C. Example: at the start of the GUI the loading is automatically from a default file, or there is also the possibility of adding new objects from the file to the C.

(5.2) (SAVE) When the gui exits the C container is automatically saved in a default file, or there’s also the possibility to save at any time C (or even some of its parts in a new file.

(5.3) Formats such as XML, JSON or other exchange formats can be considered for the format of the upload / save file of data (AXON, YAML, etc.)

(6) The GUI must display the different types of G objects appropriately. Example: the display of a book may show the image of its cover while the display of a newspaper could only show textual content.

The GUI can freely draw inspiration from content aggregators available on the web, both desktop applications and applications for mobile systems. You can join the `Model-View-Controller or Model-View design pattern for architectural design of the GUI, where the Model will include the logical model of QONTAINER. Qt includes a set of "view" classes that use an architecture "Model / view" to manage the relationship between the logical data of the GUI and the way in which they are presented to the user of the GUI (see http: //qt-project.org/doc/qt-5/model-view-programming.html). As is known, the Qt library has documentation complete and precise that will be the main reference guide in the development of the GUI, as well as offering the QtCreator IDE and the QtDesigner tool.

(MAYBE NOT USE QT DESIGNER IT MAKE A LOT OF USELESS CODE)

1. OTHER MANDATORY REQUIREMENTS

The QONTAINER project must meet the following mandatory requirements:

1. Separation between logical model and GUI: the logical model of the G hierarchy has no relation with the GUI code. There

G hierarchy should also be used by a GUI written in a framework other than Qt.

2. Polymorphous code must be used that exploits the hierarchical organization of the types of G

1. IMPORTANT RULES

A good project will have to be developed following the fundamental principles of object-oriented programming, even as far as'

concerns the development of the graphic interface. The evaluation of the project will take into consideration the following criteria: `

1. Correctness: the project must:

(a) compile and execute correctly in the machine ssh.studenti.math.unipd.it, of which an image has been provided by use as a virtual machine in VirtualBox.

NB: this is a necessary condition for the evaluation of the project.

The compilation should be possible by invoking the sequence of commands:

qmake -project ⇒ qmake ⇒ make.

If the compiling the project requires a project file (.pro) for qmake different from the one obtainable through the invocation of qmake -project then it must also be delivered a progetto.pro file that allows automatic generation via qmake of the Makefile, and this should be explicitly documented in the report.

(b) fully and correctly satisfy all mandatory requirements

2. Object orientation: the project must include polymorphic calls. They will be subject to evaluation the following qualities of the product code:

(a) encapsulation

(b) modularity (in particular, maximum separation between the logical and graphic part (GUI) of the code)

(c) extensibility and evolution, in particular by polymorphism

(d) efficiency and robustness 3.

Functionality: how many and which functionalities the project makes available, and their quality.

GUI: correct use of the Qt library; quality, usability and robustness of the GUI.

Relationship: clarity and quality of the relationship, which will concern the following aspects: ` (a) mandatory description of all hierarchies of types used

(b) mandatory description of the use of polymorphic calls (c)

mandatory description of the container upload / save file format

(d) if the application requires it, the GUI user manual (e) if necessary, indication of any compilation and execution instructions

The project must compile and execute correctly on the Linux machine ssh.studenti.math.unipd.it (which was given a ` image to be used as a virtual machine for VirtualBox) with the GNU compiler g ++ 5.x (currently 5.4.0), the Qt library in version 5.x (currently 5.5.1) and with Java compiler and virtual machine (currently 1.8.0). It is of course possible to develop ` project on other operating systems like MacOS / Windows. In this case, before submitting the project, remember to carry out a test compilation, execution and operation on the Linux machine.