UDACITY

‹ Return to "C++" in the classroom

DISCUSS ON STUDENT HUB

Memory Management Chatbot

REVIEW

CODE REVIEW 7

HISTORY

Meets Specifications

You have nailed it.Your application looks fine.
This is a great start! Keep it up and get ready for much more awesome knowledge to come in the next projects!
Good luck! Happy learning👍
Cheers

Quality of Code

The code compiles and runs with `cmake` and `make`.

✅ The code compiles and runs with cmake.

Task 1: Exclusive Ownership 1

In file `chatgui.h` / `chatgui.cpp`, `_chatLogic` is made an exclusive resource to class `ChatbotPanelDialog` using an appropriate smart pointer. Where required, changes are made to the code such that data structures and function parameters reflect the new structure.

✅ Used an appropriate smart pointer.

Task 2: The Rule of Five

In file `chatbot.h` / `chatbot.cpp`, changes are made to the class `ChatBot` such that it complies with the Rule of Five. Memory resources are properly allocated / deallocated on the heap and member data is copied where it makes sense. In each of the methods (e.g. the copy constructor), a string of the type "ChatBot Copy Constructor" is printed to the console so that it is possible to see which method is called in later examples.

✅ Great work on the Rule of Five.

Task 3: Exclusive Ownership 2

In file `chatlogic.h` / `chatlogic.cpp`, the vector `_nodes` are adapted in a way that the instances of `GraphNodes` to which the vector elements refer are exclusively owned by the class `ChatLogic`. An appropriate type of smart pointer is used to achieve this.

✅ An appropriate type of smart pointer is used to achieve this.

When passing the `GraphNode` instances to functions, ownership is not transferred.

✅ Great work on the GraphNode.

Task 4: Moving Smart Pointers

In files `chatlogic.h` / `chatlogic.cpp` and `graphnodes.h` / `graphnodes.cpp` all instances of `GraphEdge` are changed in a way such that each instance of `GraphNode` exclusively owns the outgoing `GraphEdges` and holds non-owning references to incoming `GraphEdges`. Appropriate smart pointers are used to do this. Where required, changes are made to the code such that data structures and function parameters reflect the changes.

✅ Great work here!
Changes are made to the code such that data structures and function parameters reflect the changes.

In files `chatlogic.h` / `chatlogic.cpp` and `graphnodes.h` / `graphnodes.cpp`, move semantics are used when transferring ownership from class `ChatLogic`, where all instances of `GraphEdge` are created, into instances of `GraphNode`.

✅ Great work here!

Task 5: Moving the ChatBot

In file `chatlogic.cpp`, a local `ChatBot` instance is created on the stack at the bottom of function `LoadAnswerGraphFromFile` and move semantics are used to pass the `ChatBot` instance into the root node.

✅ Great work here!
Moved semantics are used to pass the ChatBot instance into the root node.

`ChatLogic` has no ownership relation to the `ChatBot` instance and thus is no longer responsible for memory allocation and deallocation.

✅ Great work here!

When the program is executed, messages are printed to the console indicating which Rule of Five component of `ChatBot` is being called.

✅ Great work here!

DOWNLOAD PROJECT

7 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review