UDACITY

< Return to Classroom

Memory Management Chatbot

REVIEW

CODE REVIEW 7

HISTORY

▸ src/chatlogic.cpp 5

▸ src/graphnode.h 1

▾ src/chatbot.h 1

```
#ifndef CHATBOT_H_
#define CHATBOT_H_

#include <wx/bitmap.h>
#include <string>

class GraphNode; // forward declaration
class ChatLogic; // forward declaration

class ChatBot
{
private:
    // data handles (owned)
    wxBitmap *_image; // avatar image

    // data handles (not owned)
    GraphNode *_currentNode;
    GraphNode *_rootNode;
    ChatLogic *_chatLogic;

    // proprietary functions
    int ComputeLevenshteinDistance(std::string s1, std::string s2);

public:
    // constructors
    ChatBot();                          // constructor WITHOUT memory allocation
    ChatBot(std::string filename); // constructor WITH memory allocation

    // Rule of Five : 1 (destructor)
    ~ChatBot();

    //// STUDENT CODE : Task 2
```

```
33      ////
34
35      // Rule of Five : 2 (assignment operator)
36      // The default assignment operation performs a shallow copy.
37      // If a deep copy is needed, it has be implemented by the programmer.
38      ChatBot &operator=(const ChatBot &source);
39
40      // Rule of Five : 3 (copy contructor)
41      // The default copy constructor performs a shallow copy.
42      // If something else is needed, the programmer has to implement it.
43      ChatBot(const ChatBot &source);
44
45      // Rule of Five : 4 (move contructor)
46      // Copying objects can be an expensive operation.
47      // The move constructor transfers the ownership of a resource from a rvalue object to a lvalue obje
48      ChatBot(ChatBot &&source);
49
50      // Rule of Five : 5 (move assignment operator)
51      // With this operator, ownership of a resource can be transferred from one object to another.
52      // The internal behavior is very similar to the move constructor.
53      ChatBot &operator=(ChatBot && source);
54
```

AWESOME

All the declarations for the Rule of Five look good.

```
55      ////
56      //// EOF STUDENT CODE
57
58      // getters / setters
59      void SetCurrentNode(GraphNode *node);
60      void SetRootNode(GraphNode *rootNode) { _rootNode = rootNode; }
61      void SetChatLogicHandle(ChatLogic *chatLogic) { _chatLogic = chatLogic; }
62      wxBitmap *GetImageHandle() { return _image; }
63
64      // communication
65      void ReceiveMessageFromUser(std::string message);
66  };
67
68  #endif /* CHATBOT_H_ */
```

▸ **src/graphnode.cpp**

▸ **src/graphedge.h**

▸ **src/graphedge.cpp**

▸ **src/chatlogic.h**

▸ **src/chatgui.h**

▸ **src/chatgui.cpp**

▸ **src/chatbot.cpp**

▸ **src/answergraph.txt**

▸ **CMakeLists.txt**

RETURN TO PATH

Rate this review