```
#pragma once
#include "CTextLineObject.h"


class CTextEditorObject :
    public CAppObject
{
public:
    // Public contructors
    CTextEditorObject(CRect bounds, CString ID, BOOL lineNums, int maxLines = 0, int defBoxHeight = 0, BOOL active = FALSE, std::vector<CString> text = {L""}, std::vector<int> line = {1}, int lineOffset = 0);
    ~CTextEditorObject();

    // Public implementations
    int draw(CDC* pDc, CSize textExtent, int xScrollPosition, int returnNewLines, BOOL printing, CRect printAreaLength = CRect());

    // Public getters, setters and checkers
    int getCaretPos();
    CPoint getCaretPoint(CSize caretSize);
    CSize getTextExtent();
    void setTextExtent(CSize size);

    int getRecentPos();

    BOOL pointHighlighted(CPoint point);
    BOOL hasHighlight();
    void hlghtingOff();
    CRect getHighlightClippingRect();
    CRgn* getHighlightExactRgn(int x_offset, int y_offset);
    int getStartLine();
    BOOL isHlghtMultiline();
    int lineHighlight(int line);
    // RETURNS  0 for none
    //          1 for partially highlighted
    //          2 for line highlighted

    std::vector <int> iGetLineNum(int a = 0);
    CString sGetLineNum(int a = 0);

    void incrementSublines(int subline, int val);
    int getActiveLine();
    int getBlockLine();
    int getPrintLine();
    CString getLineText(int line = 0);
    CRect getLineBounds(int line = 0);
    CString getHighlightedText();
    int getLineTextWidth();
    int getNumLines();
    int getBoxHeight(BOOL default = FALSE);

    virtual void setActive(BOOL active);
    virtual void move(int x, int y);
```

```
52      virtual void setBounds(CRect bounds);
53      std::tuple<CRect, int> getPrintBounds(int returnNewLines, int
          printAreaLength);
54
55      int getCursorArrow();
56
57      void initialise();
58
59      // Public message handlers
60      virtual void OnSize(UINT nType, int x, int y);
61
62      virtual BOOL OnLButtonUp(UINT nFlags, CPoint point);
63      virtual BOOL OnLButtonDown(UINT nFlags, CPoint point);
64      virtual void OnLButtonDblClk(UINT nFlags, CPoint point);
65      virtual void OnRButtonUp(UINT nFlags, CPoint point);
66      virtual void OnRButtonDown(UINT nFlags, CPoint point);
67      virtual BOOL OnMouseMove(UINT nFlags, CPoint point);
68
69      virtual int OnRecieveText(CString text, BOOL open = FALSE);
70      virtual void OnRecieveBackspace();
71      virtual void OnRecieveTab();
72      virtual void OnRecieveReturn(BOOL open = FALSE);
73
74      virtual void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
75      virtual BOOL OnContextMenu(CWnd* pWnd, CPoint point);
76
77  private:
78
79      // Private Implementations
80      void setSidebar();
81      void setHighlighter();
82      void setBracketsNull();
83
84      // Private resources
85
86      // Positioning
87      int activeLine;
88      int mouseLine;
89      int blockLine;
90      int printLine;
91      int caretPos;
92
93      // Sizes
94      int defBoxHeight;
95      CSize textExtent;
96
97      // Highlights
98      BOOL hlght;             // Used to tell the editor that there is
          a highlight
99      BOOL hlghting;          // Used to tell that the user is
          currently highlighting
100     CPoint hlghtStartP;     // Stores the start of the highlight
101     CPoint hlghtEndP;       // Stores the end/current mouse pos for
          the highlighting
102     int startPos;           // Stores the original starting pos in
          terms of character spaces for the highlight
```

```
103     int startLine;          // Stores the original starting line for the highlight
104
105     // Recents
106     BOOL recentHlght;
107     int recentPos;
108     int recentLine;
109
110     // Line stuff
111     std::vector<CTextLineObject*> lines;
112     int maxLines;
113     int lineOffset;
114     BOOL lineNums;
115     BOOL hoverSubLine;
116     BOOL clickSubLine;
117     CRect sidebar;
118
119     // Editing
120     std::vector<std::vector<int>> brackets;
121
122     /*
123
124     Brackets contain the positions and types of all the auto completed brackets currently in the editor.
125     Auto-brackets must always be on the same line and it will only be auto for as long the user is editing that line.
126
127     Each record in brackets contain three integers.
128
129     [x][0] contains the position of the first bracket.
130     [x][1] contains the position of the second bracket.
131     [x][2] contains the type of bracket that has been saved.
132
133     if [x][2] == 0 : type = (
134                  1 : type = {
135                  2 : type = [
136                  3 : type = "
137                  4 : type = '
138
139     */
140
141
142     BOOL bracketContains(int value, int& itPos, int type, int side = 1);
143     void moveBrackets(int val, int index);
144
145     // Stuff for view
146     int cursor_arrow;
147     BOOL lMouseDown;
148 };
149
150
```