		using mapKeyValues = std::map <std::string, std::map<std::string,="" std::string="">>;</std::string,>
UI		INIEditor
+ UI(INIEditor *);	~	1+ INIEditor()
- editor: unique_ptr <inieditor></inieditor>		+ ~INIEditor(): Has custom implementation for unexpected exits
 - void clearCin() const; - bool confirmInputPrompt() const; - string inputName() const; - unsigned int inputNum() const; 		+ vector <string> getLines() const; + vector<string> getNumberOfLines() const; + void listSections() const; + string getCurrentFilename() const; + pair<string, string=""> getKeyValuePair(unsigned int) const;</string,></string></string>
- vector <string> detectINIs() const;</string>		+ const shared_ptr <mapkeyvalues> getPtrToKeyValuesMap() const;</mapkeyvalues>
 - void mainMenu() const; - void createNew() const; - void loadFile() const; - void workWithFile() const; - void lineMenu(unsigned int) const; - void deleteLine(unsigned int) const; - void editLine(unsigned int) const; - void insertLine(unsigned int) const; 		+ void insertLine(unsigned int, string); + void clearAll(); + void deleteLine(unsigned int); + void parseWorkingFile(); + void replaceEntireLine(int, string); + void replaceLineByKeyValuePair(int, pair <string, string="">); + void setExpectedExit(); + void setWorkingFile(string); + void writeLinesToFile(); - lines: vector<string></string></string,>
		- sectionsKeyValues: shared_ptr <map<string, map<string,="" string="">>> - workingFile: string - isUnexectedExit: bool</map<string,>
		 void parseMapFromLines(); void replaceLine(int, string&); void saveCurrentLines(string); vector<string> parseFileLinesToStringVector(string) const;</string> pair<string, string=""> getKeyValuePair(string) const;</string,> shared_ptr<mapkeyvalues> getSectionsKeyValuePairsMap(vector<string>) const;</string></mapkeyvalues>

public methods without s/e

public methods with s/e

> private attributes

private helper methods