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

#ifndef COMBATLOGLINE_H
#define COMBATLOGLINE_H

class CombatLogLine
{
public:
	CombatLogLine(string _theLineOfRawText);
	//void ParseTheLine();
	void splitString(const string &inputString, char delim, vector<string> &elements);
	vector<string> theParsedLine;
	unsigned int sizeOfLine;
	string getTokenNumber(unsigned int x);
	string determineTypeOfEvent();
	string typeOfEvent;

	bool virtual areFieldsOk();
	void virtual parseLine();

protected:
	string theLineOfRawText;
};

class CombatLogVersionLine : public CombatLogLine
{
public:

//http://stackoverflow.com/questions/8403896/c-error-no-appropriate-default-constructor-availabl
e
	//need to call CombatLogLine's constructor in the initializer of the constructor for the
inheriting class
	CombatLogVersionLine(string _theLineOfRawText) :
CombatLogLine(_theLineOfRawText)
	{
		typeOfEvent = "COMBAT_LOG_VERSION";
	}
	virtual bool areFieldsOk();
	void parseLine();
private:
};
```

```
class UnitDiedLine : public CombatLogLine
{
public:
	UnitDiedLine(string _theLineOfRawText) : CombatLogLine(_theLineOfRawText)
	{
		typeOfEvent = "UNIT_DIED";
	}
};

#endif




#include "stdafx.h"
#include "CombatLogLine.h"

CombatLogLine::CombatLogLine(string _theLineOfRawText)
{
	sizeOfLine = 0;
	splitString(_theLineOfRawText,',',theParsedLine);
	sizeOfLine = theParsedLine.size();
	typeOfEvent = "TBA";
}

bool CombatLogLine::areFieldsOk()
{
	cout << "This is the base classes areFieldsOk" << endl;
	return false;
}

void CombatLogLine::parseLine()
{

}


bool CombatLogVersionLine::areFieldsOk()
{
	if(typeOfEvent == "COMBAT_LOG_VERSION")
	{
		if(sizeOfLine == 2)
		{
```

```
                        cout << "Hallelujah!" << endl;
                        return true;
                }
        }
        return false;
}

void CombatLogVersionLine::parseLine()
{

}



void CombatLogLine::splitString(const string &inputString, char delim, vector<string>
&elements)
{
        std::stringstream ss;
   ss.str(inputString);
   string item;
   while (std::getline(ss, item, delim))
        {
      elements.push_back(item);
   }
}

string CombatLogLine::getTokenNumber(unsigned int x)
{
        if(x < sizeOfLine)
                return theParsedLine[x];
        else
        {
                cout << x << " was the element number requested but it didn't exist." << endl;
                system("PAUSE");
                return "NULL";
        }
}
```

```
// Reflection.cpp : Defines the entry point for the console application.
//

//project properties -> general -> linker -> enable incremental linking -> No
//http://www.cplusplus.com/doc/tutorial/pointers/

#include "stdafx.h"
#include "Parser.h"

//http://forums.askmrrobot.com/index.php?topic=9627.0
//https://forums.wowace.com/showthread.php?t=10661  combat log events
//http://forums.worldoflogs.com/viewtopic.php?p=15240#p15240  combat log events 2

//https://bytes.com/topic/c/answers/805719-safely-reading-large-files
//http://insanecoding.blogspot.com/2011/11/how-to-read-in-file-in-c.html
//http://stackoverflow.com/questions/3613065/when-to-pass-by-reference-and-when-to-pass-by-pointer-in-c

//http://stackoverflow.com/questions/4049580/what-is-the-difference-between-typecasting-and-typeconversion-in-c-or-java


std::vector<std::string> split(const std::string &s, char delim);
void splitString(const string &s, char delim, vector<string> &elems);
void splitAndRemoveLastChar(const std::string &s, char delim, std::vector<std::string> &elems);

CombatLogLine determineTypeOfEvent(string _fieldZero);

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned int startTime = static_cast<unsigned int>(time(0));
	unsigned int endTime = static_cast<unsigned int>(time(0));
{
	_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
	srand(startTime);
	string newLine = "\n";
	const char* newLineChar = "\n";

	//char* logFilePathChar("C:\\supershorttext.txt");
```

```
	//char* logFilePathChar("C:\\shorttext.txt");
	char* logFilePathChar("C:\\25MegLog.txt");
//https://www.warcraftlogs.com/reports/fJXMFPAhBzxNj1nH/
//http://www.worldoflogs.com/reports/raxixg9gswmmmjjk/
	//char* logFilePathChar("C:\\50MegLog.txt");
	//char* logFilePathChar("C:\\100MegLog.txt");
	//char* logFilePathChar("C:\\180MegLog.txt");
	//char* logFilePathChar("C:\\400MegLog.txt");

	vector<string> test1;
	test1.push_back("Hello");


	string theWholeCombatLog;
	unsigned long megsInFile;
	//check file size
	ifstream myTestForFileSizeInputFileStream(logFilePathChar, ifstream::ate |
ifstream::binary);
	if(myTestForFileSizeInputFileStream)
	{
		ifstream::pos_type myPosition = myTestForFileSizeInputFileStream.tellg();
		unsigned long myPositionLong = static_cast<unsigned long> (myPosition);  //this
might not work if myPosition is greater than long
		megsInFile = (myPositionLong / (1000000) + 1);
		cout << "Megs in file:" << megsInFile << endl;
		if(megsInFile > 400)
		{
			cout << "File larger than 400 MB, exiting." << endl;
			system("PAUSE");
			return 1;
		}
	}
	else
	{
		cout << "File did not exist." << endl;
		system("PAUSE");
		return 2;
	}
	//load file into theWholeCombatLog
	ifstream myInputFileStream(logFilePathChar, std::ios::in | std::ios::binary);
	if (myInputFileStream)
	{
		myInputFileStream.seekg(0, std::ios::end);
```

```
		//http://stackoverflow.com/questions/15172315/c-resolving-istreamtellg-warning
		unsigned int temporarySize = static_cast<unsigned int>
(myInputFileStream.tellg());  //this might not work if the file is very large
		theWholeCombatLog.resize(temporarySize);
		myInputFileStream.seekg(0, std::ios::beg);
		myInputFileStream.read(&theWholeCombatLog[0], theWholeCombatLog.size());
		myInputFileStream.close();
	}
	else
	{
		cout << "Problem reading the file" << endl;
		system("PAUSE");
		return 3;
	}

//http://stackoverflow.com/questions/10108985/is-it-more-efficient-to-set-the-size-of-a-vector-up-f
ront
	vector<string> combatLogSplitIntoLines;
	combatLogSplitIntoLines.reserve(4200 * megsInFile);//roughly 4200 lines per meg
	splitAndRemoveLastChar(theWholeCombatLog, '\n', combatLogSplitIntoLines);
	//cout << "Size of combatLog vector is " << combatLogSplitIntoLines.size() << endl;

	std::string::size_type sz;   // alias of size_t  //dunno why that's needed


	//for(int i = 0; i < combatLogSplitIntoLines.size(); i++)
	unsigned int healingDoneByYserasGift = 0;
	unsigned int healingDoneByYserasGiftCounter = 0;
	unsigned int overHealingDoneByYserasGift = 0;
	unsigned int healingDoneByYserasGift145110 = 0;
	unsigned int healingDoneByYserasGift145110Counter = 0;
	unsigned int overHealingDoneByYserasGift145110 = 0;

	unsigned int sizeOfCurrentLine = 0;
	//for(unsigned int i = 0; i < 100000 && i < combatLogSplitIntoLines.size(); i++)
	for(unsigned int i = 0; i < combatLogSplitIntoLines.size(); i++)
	{
		CombatLogLine currentLine =
determineTypeOfEvent(combatLogSplitIntoLines[i]);//determine type of event also creates an
appropriate CombatLogLine
		//vector<string> oneLine = vector<string>();
		//oneLine.reserve(50);
		//splitString(combatLogSplitIntoLines[i],',',oneLine);
```

```
		//sizeOfCurrentLine = oneLine.size();

		if(currentLine.typeOfEvent == "COMBAT_LOG_VERSION")
		{
			cout << "COMBAT_LOG_VERSION" << endl;
			//CombatLogVersionLine temp =
static_cast<CombatLogVersionLine>(currentLine);
			//CombatLogLine* temp = &currentLine;
			//CombatLogVersionLine* temp2 =
dynamic_cast<CombatLogVersionLine*>(temp);
			//temp2->areFieldsOk();
			currentLine.areFieldsOk();
			//currentLine.areFieldsOk();
			cout << "type id " << typeid(currentLine).name() << endl;

			//CombatLogVersionLine temp3 = (CombatLogVersionLine) currentLine;


			//CombatLogVersionLine * temp4 = (CombatLogVersionLine*)
&currentLine;
			CombatLogVersionLine * temp5 = dynamic_cast<CombatLogVersionLine
*>(&currentLine);
			if(temp5 == 0)
			{
				cout << "The dynamic cast failed." << endl;
				system("PAUSE");
				return 0;
			}
			else
			{
				temp5->areFieldsOk();
				continue;
			}
		}
		if(currentLine.sizeOfLine < 26)
		{
			//cout << "continue activated " << i << endl;
			continue;
		}
		//cout << i << " " << sizeOfCurrentLine << endl;
		//if(sizeOfCurrentLine < 25)
		//	continue;
		if((currentLine.getTokenNumber(2)).find("Tantreyetal") != string::npos)
```

```
		{
			//cout << oneLine[2].find("Tantreyetal") << " was oneLine[2].find(tant)" ;
			//cout << "I finally actually parsed something!" << endl;
			if(currentLine.getTokenNumber(0).find("SPELL_HEAL") != string::npos)
			{
				if(currentLine.getTokenNumber(9).find("145109") != string::npos)
				{
					unsigned int temp =
std::stoi(currentLine.getTokenNumber(26), &sz);
					overHealingDoneByYserasGift += temp;
					healingDoneByYserasGift = ((healingDoneByYserasGift +
std::stoi(currentLine.getTokenNumber(25),&sz)) - temp);
					healingDoneByYserasGiftCounter++;
					cout << "i " << i << " Ysera 145109: " <<
healingDoneByYserasGift << "  " << healingDoneByYserasGiftCounter << " " << "overHealing: "
<< overHealingDoneByYserasGift << endl;
				}
				if(currentLine.getTokenNumber(9).find("145110") != string::npos)
				{
					unsigned int temp =
std::stoi(currentLine.getTokenNumber(26), &sz);
					overHealingDoneByYserasGift145110 += temp;
					healingDoneByYserasGift145110 =
((healingDoneByYserasGift145110 + std::stoi(currentLine.getTokenNumber(25),&sz)) - temp);
					healingDoneByYserasGift145110Counter++;
					cout << "i " << i << " Ysera 145110: " <<
healingDoneByYserasGift145110 << "  " << healingDoneByYserasGift145110Counter << " " <<
"overHealing: " << overHealingDoneByYserasGift145110 << endl;
				}

			}
		}
	}
	cout << "Healing done by Ysera's Gift 145109:  " << healingDoneByYserasGift << endl;
	cout << "Healing done by Ysera's Gift 145110:  " << healingDoneByYserasGift145110 <<
endl;
//  11/18 10:55:35.592
SPELL_HEAL,Player-1165-013EFAFE,"Tantreyetal-Arathor",0x511,0x0,Player-90-04EF8353,"H
ealthenkill-Terenas",0x514,0x0,145110,"Ysera's
Gift",0x8,0000000000000000,0000000000000000,0,0,0,0,-1,0,0,0,0.00,0.00,0,68566,0,0,nil
	endTime = static_cast<unsigned int>(time(0));
	cout << "Time before close quotes: " << endTime-startTime << endl;
```

```
}  //interesting, deallocating the memory in debug mode takes forever but is instant in release
mode or with ctrl f5
endTime = static_cast<unsigned int>(time(0));
cout << "Final Time: " << endTime- startTime << endl;

system("PAUSE");
return 0;
}

//vector<string> split(const string &s, char delim)
//{
//    vector<string> elems;
//    split2(s, delim, elems);
//    return elems;
//}
CombatLogLine determineTypeOfEvent(string _aLine)
{
        if(_aLine == "")
        {
                cout << "Something went wrong asdghaer" << endl;
                system("PAUSE");
        }
        if(_aLine.find("COMBAT_LOG_VERSION") != string::npos)
        {
                CombatLogVersionLine returnValue(_aLine);
                cout << "A CombatLogVersionLine was created. " <<
typeid(CombatLogVersionLine).name() << endl;
                return returnValue;
        }
        return CombatLogLine(_aLine);
        //vector<string> fail2(0);
//http://stackoverflow.com/questions/2376989/why-dont-stdvectors-elements-need-a-default-con
structor
        return (CombatLogLine("FAIL"));
}


void splitString(const string &inputString, char delim, vector<string> &elements)
{
        std::stringstream ss;
   ss.str(inputString);
   string item;
   while (std::getline(ss, item, delim))
```

```
        {
    elements.push_back(item);
  }
}


void splitAndRemoveLastChar(const string &inputString, char delim, vector<string> &elements)
{
        //removes the carriage return at the end of the line
        //unsigned int counter = 0;
  std::stringstream ss;
  ss.str(inputString);
  string item;
        string truncatedItem;
  while (std::getline(ss, item, delim))
        {
                truncatedItem = item.substr(0, item.length() - 1);
    elements.push_back(truncatedItem);
                //counter++;
  }
        //cout << "Number of lines from split and remove last char: " <<  counter << endl;
}
```

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#include "targetver.h"

#include "targetver.h"
#include <stdio.h>
#include <tchar.h>
#include <time.h>
#include <random>
```

```
#include <string>

#include <boost/scoped_ptr.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/ptr_container/ptr_vector.hpp>
#include <boost/multi_array.hpp>
#include <boost/lambda/lambda.hpp>
#include <boost/tokenizer.hpp>


#include <cassert>
#include <crtdbg.h>
#include <cctype>

#include <fstream>
#include <ostream>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <ios>

#include <typeinfo>

#include <iterator>
#include <algorithm>
#include <forward_list>
#include <limits.h>

using std::string;
using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::fstream;
using std::ifstream;

#include <cerrno>


// TODO: reference additional headers your program requires here
```