#include <iostream>

#include <string>

#include <fstream>

#include <stdexcept>

#include "TreasureMap.h"

#include "FileReadException.h"

#include "Location.h"

using namespace std;

//overload operator << of Location class, no changes needed

ostream& operator<<(ostream& os, Location& pt) {

return os << pt.getName();

}

//overload operator >> of Location class, no changes needed

istream& operator >> (istream& os, Location& pt) {

string name;

float x,y;

os >> name >> x >> y;

pt.setName(name);

pt.setX(x);

pt.setY(y);

return os;

}

//FIX ME: Implement readmap function that reads from a file and return a proper TreasureMap object

template<typename T>

TreasureMap<T> readMap(string file\_name, int max\_steps) {

ifstream input;

input.open(file\_name);

if (input.is\_open()) {

TreasureMap<T> Tmap = new TreasureMap<T>(max\_steps);

T a;

string line;

int count = 0;

while (input >> a)

{

if (count == max\_steps) {

FileReadException err = new FileReadException("Error reading map : " );

throw err;

}

Tmap.addStep(a);

count++;

}

return Tmap;

}

else {

FileReadException err = new FileReadException("Error reading map: ");

throw err;

}

}

int main()

{

string file\_name, file\_type;

int max\_steps;

cin >> file\_name >> file\_type >> max\_steps;

try {

if (file\_type == "string") {

TreasureMap<string> Tmap = readMap<string>(file\_name, max\_steps);

for (int i = 0; i < Tmap.getTotalSteps() - 1; i++) {

Tmap.nextStep();

}

}

else if (file\_type == "int") {

//FIX ME: Create a TreasureMap of type int, call readMap to read the file\_name and file\_type

//Then use a for loop to call the map.nextStep()

TreasureMap<int> Tmap = readMap<int>(file\_name, max\_steps);

for (int i = 0; i < Tmap.getTotalSteps() - 1; i++) {

Tmap.nextStep();

}

}

else if (file\_type == "char") {

//FIX ME: Create a TreasureMap of type char, call readMap to read the file\_name and file\_type

//Then use a for loop to call the map.nextStep()

TreasureMap<char> Tmap = readMap<char>(file\_name, max\_steps);

for (int i = 0; i < Tmap.getTotalSteps() - 1; i++) {

Tmap.nextStep();

}

}

else if (file\_type == "location") {

//FIX ME: Create a TreasureMap of type Location, call readMap to read the file\_name and file\_type

//Then use a for loop to call the map.nextStep()

TreasureMap<Location> Tmap = readMap<Location>(file\_name, max\_steps);

for (int i = 0; i < Tmap.getTotalSteps() - 1; i++) {

Tmap.nextStep();

}

}

}

catch (FileReadException& err) {

cout << err.getMessage();

//FIX ME: prints out the exception message using getMessage() here

}

return 0;

}

#ifndef FILEREADEXCEPTION\_H

#define FILEREADEXCEPTION\_H

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

# pragma once

class FileReadException {

private:

string err\_message;

public:

FileReadException(string str);

string getMessage();

};

#endif

#include <string>

#include <iostream>

#include <fstream>

#include "FileReadException.h"

using namespace std;

FileReadException::FileReadException(string str) {

err\_message = str;

}

string FileReadException::getMessage() {

return err\_message;

}