/\* GCharacter.h

\* Created on Dec 26, 2015

\* Author: Hongmin

\*/

#ifndef GCHARACTER\_H

#define GCHARACTER\_H

#include"Header.h"

class GCharacter{

//Overload the insertion operator to output a GCharacter object

friend std::ostream& operator<<(std::ostream& os, const GCharacter& gc);

public:

static const int DEFAULT\_CAPACITY = 5;

//Constructor

GCharacter(string = "Grayer", int = DEFAULT\_CAPACITY);

//Copy Constructor

GCharacter(const GCharacter&);

//Overloaded Assignment Operator(returning a GCharacter&, because we could chain like gc1 = gc2 = gc3)

GCharacter& operator=(const GCharacter&);

//Destructor

~GCharacter();

//Insert a new tool into the toolHolder

void insert(const string&);

private:

//data members

string name;

int capacity;

int used;

string \*toolHolder;

};

#endif /\* GCHARACTER\_H \*/

/\* GCharacter.cpp

\* Created on Dec 26, 2015

\* Author: Hongmin

\*/

#include "GCharacter.h"

//Overload the insertion operator to output a GCharacter object

std::ostream& operator<<(std::ostream& os, const GCharacter& gc){

os << "Game Character: " << gc.name <<

"\nhas the following tools: " << endl << "| ";

//output all the items in toolHolder

for (int i = 0; i < gc.used; i++){

os << gc.toolHolder[i] + " | ";

}

return os << endl;

}

//Constructor

GCharacter::GCharacter(string n, int cpt){

this->name = n;

this->capacity = cpt;

this->used = 0;

this->toolHolder = new string[cpt];

}

//Copy Constructor

GCharacter::GCharacter(const GCharacter& source){

cout << "Copy Constructor called." << endl;

this->name = source.name;

this->capacity = source.capacity;

this->used = source.used;

this->toolHolder = new string[source.capacity];

//copy(source.toolHolder, source.toolHolder + used, toolHolder);

//the same functionality with the above "copy"

for (int i = 0; i < source.used; i++){

this->toolHolder[i] = source.toolHolder[i];

}

}

//Overloaded Assignment Operator(returning a GCharacter&, because we could chain like gc1 = gc2 = gc3)

GCharacter& GCharacter::operator=(const GCharacter& source){

cout << "Overloaded Assignment called." << endl;

//check for self assignment

//gc1 = gc1

if (this == &source){

return \*this;

}

this->name = source.name;

this->capacity = source.capacity;

this->used = source.used;

//this->toolHolder = new string[source.capacity];

//copy(source.toolHolder, source.toolHolder + used, this->toolHolder);

//the same functionality with the above "copy"

for (int i = 0; i < source.used; i++){

this->toolHolder[i] = source.toolHolder[i];

}

return \*this;

}

//Destructor

GCharacter::~GCharacter(){

cout << "Destructor called for " << this->name

<< " @ this memory location " << this << endl;

delete[]this->toolHolder;

}

//Insert a new tool into the toolHolder

void GCharacter::insert(const string& toolName){

if (used == capacity){

cout << "Tool Holder is full, cannot add more tools\n" << endl;

}

else{

this->toolHolder[used] = toolName;

this->used++;

}

}

/\*GCApp.cpp

\*

\* Created on Dec 26, 2015

\* Author: Hongmin

\*/

#include"GCharacter.h"

//example for copy constructor be called

GCharacter exmpl(GCharacter TempGC){

cout << "//Copy Constructor be called twice." << endl;

cout << "//Once for the formal parameter being passed by value" << endl;

cout << "//Once for the return value" << endl;

return TempGC;

}

int main(){

GCharacter gc1;

gc1.insert("potion"); gc1.insert("crossbow"); gc1.insert("candle");

gc1.insert("cloak"); gc1.insert("sword"); gc1.insert("book of spell");

cout << gc1 << endl;

//cout << endl;

GCharacter gc2("Bob", 5);

gc2.insert("ax");

cout << endl;

cout << "An example of function to invoke copy constructor" << endl;

exmpl(gc2); // example for copy constructor being called.

cout << endl;

//copy constructor

GCharacter gc3(gc1); //copy constructor

GCharacter gc4 = gc2; //copy constructor

cout << endl;

//overloaded assignment operator

gc4 = gc1;

cout << endl;

return 0;

}

