UML

Helped.h

#ifndef \_\_HELPED\_H\_\_

#define \_\_HELPED\_H\_\_

#include <iostream>

class Helped

{

private:

int \_helpReceived = 0;

public:

int getHelp() { return \_helpReceived; }

virtual int getHelpReceived() = 0;

virtual void setHelpReceived(int help) { \_helpReceived += help; }

virtual void resetHelpReceived() { \_helpReceived = 0; }

virtual void receiveHelp(int help) = 0;

virtual void printHelp() { std::cout << "Help: " << \_helpReceived << std::endl; }

};

#endif

Region.h

#ifndef \_\_REGION\_H\_\_

#define \_\_REGION\_H\_\_

#include "Helped.h"

#include "Person.h"

#include "Village.h"

#include "RegionObject.h"

#include <vector>

class Region: public Helped

{

std::vector<RegionObject\*> \_vector;

public:

void add(RegionObject\* element);

void receiveHelp(int help);

int getHelpReceived();

virtual void printHelp();

};

#endif

Region.cpp

#import "Region.h"

void Region::add(RegionObject\* element) { \_vector.push\_back(element); }

void Region::receiveHelp(int help)

{

int partOfvalue; // dividing the help

setHelpReceived(help);

if (\_vector.size()) // if it has elements

{

int partOfValue = help/\_vector.size();

int i;

for(i = 0; i<\_vector.size(); i++)

\_vector.at(i)->receiveHelp(partOfValue);

}

}

int Region::getHelpReceived()

{

if (\_vector.size()) {

resetHelpReceived();

int i;

for(i = 0; i<\_vector.size(); i++)

setHelpReceived(\_vector.at(i)->getHelpReceived());

}

return getHelp();

}

void Region::printHelp() { std::cout << "Help: " << getHelpReceived() << std::endl; }

Village.h

#ifndef \_\_VILLAGE\_H\_\_

#define \_\_VILLAGE\_H\_\_

#include "Helped.h"

#include "Person.h"

#include "RegionObject.h"

#include <vector>

class Village: public RegionObject

{

std::vector<Person\*> \_vector;

public:

void add(Person\* person);

void receiveHelp(int help);

int getHelpReceived();

virtual void printHelp();

};

#endif

Village.cpp

#import "Village.h"

void Village::add(Person\* person) { \_vector.push\_back(person); }

void Village::receiveHelp(int help)

{

int partOfvalue; // dividing the help

setHelpReceived(help);

if (\_vector.size()) // if it has elements

{

int partOfValue = help/\_vector.size();

int i;

Person\* person;

for(i = 0, person = \_vector.at(0); i<\_vector.size(); i++){

person = \_vector.at (i);

person->receiveHelp(partOfValue);

}

}

}

int Village::getHelpReceived()

{

if (\_vector.size()) {

resetHelpReceived();

int i;

Person\* person;

for(i = 0, person = \_vector.at(0); i<\_vector.size(); i++){

person = \_vector.at (i);

setHelpReceived(person->getHelpReceived());

}

}

return getHelp();

}

void Village::printHelp() { std::cout << "Help: " << getHelpReceived() << std::endl; }

Person.h

#ifndef \_\_PERSON\_H\_\_

#define \_\_PERSON\_H\_\_

#include "Helped.h"

#include "RegionObject.h"

class Person: public RegionObject

{

public:

void receiveHelp(int value);

int getHelpReceived();

};

#endif

Person.cpp

#include "Person.h"

void Person::receiveHelp(int value) { setHelpReceived(value); }

int Person::getHelpReceived() { return getHelp(); }

RegionObject.h

#ifndef \_\_REGIONOBJ\_H\_\_

#define \_\_REGIONOBJ\_H\_\_

#include "Helped.h"

class RegionObject: public Helped{

};

#endif

Main.cpp

#include "Person.h"

#include "Region.h"

#include "Village.h"

#include <iostream>

int main(){

Person p1;

Person p2;

Person p3;

Person p4;

Person p5;

Person p6;

Village v1;

Village v2;

Region r1;

Region r2;

// test persons receiving help

p1.receiveHelp(6);

p1.printHelp();

p2.receiveHelp(10);

p2.printHelp();

// add p1 and p2 to village v1

v1.add(&p1);

v1.add(&p2);

// add people to village v2

v2.add(&p3);

p4.receiveHelp(2);

v2.add(&p4);

v2.printHelp();

// test help division

v2.receiveHelp(10);

p3.printHelp();

p4.printHelp();

v1.printHelp();

v2.printHelp();

// add villages to region

r1.add(&v1);

r1.add(&v2);

r1.printHelp();

// test money division in region

v1.printHelp();

v2.printHelp();

p1.printHelp();

p2.printHelp();

p3.printHelp();

p4.printHelp();

r1.receiveHelp(40);

v1.printHelp();

v2.printHelp();

p1.printHelp();

p2.printHelp();

p3.printHelp();

p4.printHelp();

// create region with people and test division

r2.add(&p5);

r2.add(&p6);

r2.receiveHelp(100);

p5.printHelp();

p6.printHelp();

}