PartHouse.h

#pragma once

#include <string>

#include <iostream>

#define interface class

interface PartHouse

{

public:

virtual std::string getName() const = 0;

virtual ~PartHouse() {}

};

class Floor : public PartHouse

{

private:

std::string name;

public:

Floor() :

name("Floor") {}

virtual std::string getName() const

{

return name;

}

};

class Wall : public PartHouse

{

private:

std::string name;

public:

Wall() :

name("Wall") {}

virtual std::string getName() const

{

return name;

}

};

class Roof : public PartHouse

{

private:

std::string name;

public:

Roof() :

name("Root") {}

virtual std::string getName() const

{

return name;

}

};

IHouseBulder.h

#pragma once

#define interface class

interface IHouseBuilder

{

public:

virtual void BuildFloor() = 0;

virtual void BuildWall() = 0;

virtual void BuildRoof() = 0;

virtual ~IHouseBuilder() {};

};

HouseDirector.h

#pragma once

#include "House.h"

#include "PartHouse.h"

#include "IHouseBuilder.h"

class HouseDirector

{

private:

IHouseBuilder\* builder;

public:

HouseDirector() :

builder(NULL){}

void setBuilder(IHouseBuilder\* builder)

{

delete this->builder;

this->builder = builder;

}

~HouseDirector()

{

delete builder;

}

void BuildAll()

{

builder->BuildFloor();

builder->BuildRoof();

builder->BuildWall();

}

};

House.h

#pragma once

#include <iostream>

#include <string>

#include <list>

#include <memory>

#define interface class

#include "PartHouse.h"

class House

{

private:

std::list<PartHouse\*> partsOfHouse;

public:

void AddPartHouse(PartHouse\* partHouse)

{

partsOfHouse.push\_back(partHouse);

}

std::string Result() const

{

std::string str;

for (auto it = partsOfHouse.begin(); it != partsOfHouse.end(); ++it)

{

str += (\*it)->getName() + " ";

}

return str;

}

~House()

{

for (auto it = partsOfHouse.begin(); it != partsOfHouse.end(); ++it)

{

delete \*it;

}

}

};

GreateHouseBuilder.h

#pragma once

#include <iostream>

#include <memory>

#define interface class

#include "House.h"

#include "PartHouse.h"

class GreatHouseBuilder : public IHouseBuilder

{

private:

House\* house;

public:

GreatHouseBuilder()

{

house = new House();

}

void Reset()

{

delete house;

house = new House();

}

virtual void BuildFloor()

{

house->AddPartHouse(new Floor());

}

virtual void BuildWall()

{

house->AddPartHouse(new Wall());

}

virtual void BuildRoof()

{

house->AddPartHouse(new Roof());

}

const auto& GetHouse() const

{

return (\*house);

}

virtual ~GreatHouseBuilder()

{

delete house;

house = NULL;

};

};

15.cpp

#include <iostream>

#include <list>

#include <memory>

#define interface class

#include "HouseDirector.h"

#include "PartHouse.h"

#include "GreateHouseBuilder.h"

int main()

{

HouseDirector hd;

GreatHouseBuilder\* ghb = new GreatHouseBuilder();

hd.setBuilder(ghb);

hd.BuildAll();

std::cout << ghb->GetHouse().Result() << std::endl;

std::cout << ghb->GetHouse().Result() << std::endl;

system("pause");

return 0;

}