МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е. О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

«Прямое и обратное проектирование»

ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ»

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4831 |  |  |  | К.А. Корнющенков |
| к |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

1. Цель работы:

Целью данной работы является получение навыков генерации кода для UML-модели и анализ программного обеспечения через обратное проектирование.

1. Задание на лабораторную работу:

По разработанной в ЛР No2 диаграмме классов осуществить генерацию кода в любой объектно-ориентированный язык и выполнить обратное проектирование полученного кода. Сравнить исходную и полученную модель.

Тема: агентство недвижимости.

1. Листинги кодов, полученных при прямом проектировании:

Admin.cpp

#include "Admin.h"

// Constructors/Destructors

//

Admin::Admin()

{

initAttributes();

}

Admin::~Admin()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Admin::initAttributes()

{

}

Admin.h

#ifndef ADMIN\_H

#define ADMIN\_H

#include "User.h"

#include <string>

/\*\*

\* class Admin

\*

\*/

class Admin : public User

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Admin();

/\*\*

\* Empty Destructor

\*/

virtual ~Admin();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void createItem()

{

}

/\*\*

\*/

void DeleteItem()

{

}

/\*\*

\*/

void EditItem()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

std::string name;

std::string phoneNumber;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of name

\* @param new\_var the new value of name

\*/

void setName(std::string new\_var)

{

name = new\_var;

}

/\*\*

\* Get the value of name

\* @return the value of name

\*/

std::string getName()

{

return name;

}

/\*\*

\* Set the value of phoneNumber

\* @param new\_var the new value of phoneNumber

\*/

void setPhoneNumber(std::string new\_var)

{

phoneNumber = new\_var;

}

/\*\*

\* Get the value of phoneNumber

\* @return the value of phoneNumber

\*/

std::string getPhoneNumber()

{

return phoneNumber;

}

void initAttributes();

};

#endif // ADMIN\_H

Buyer.cpp

#include "Buyer.h"

// Constructors/Destructors

//

Buyer::Buyer()

{

initAttributes();

}

Buyer::~Buyer()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Buyer::initAttributes()

{

}

Buyer.h

#ifndef BUYER\_H

#define BUYER\_H

#include "User.h"

#include <string>

#include <vector>

/\*\*

\* class Buyer

\*

\*/

class Buyer : public User

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Buyer();

/\*\*

\* Empty Destructor

\*/

virtual ~Buyer();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void regist()

{

}

/\*\*

\*/

void updateProfile()

{

}

/\*\*

\*/

void lookPurchaseList()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

std::string purchaseList;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of purchaseList

\* @param new\_var the new value of purchaseList

\*/

void setPurchaseList(std::string new\_var)

{

purchaseList = new\_var;

}

/\*\*

\* Get the value of purchaseList

\* @return the value of purchaseList

\*/

std::string getPurchaseList()

{

return purchaseList;

}

void initAttributes();

};

#endif // BUYER\_H

Catalog.cpp

#include "Catalog.h"

// Constructors/Destructors

//

Catalog::Catalog()

{

initAttributes();

}

Catalog::~Catalog()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Catalog::initAttributes()

{

}

Catalog.h

#ifndef CATALOG\_H

#define CATALOG\_H

#include "<<interface>>\_Search.h"

#include <string>

#include <vector>

/\*\*

\* class Catalog

\*

\*/

class Catalog : public \_interface\_Search

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Catalog();

/\*\*

\* Empty Destructor

\*/

virtual ~Catalog();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void viewData()

{

}

/\*\*

\*/

void search\_()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

std::string filterSearch;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of filterSearch

\* @param new\_var the new value of filterSearch

\*/

void setFilterSearch(std::string new\_var)

{

filterSearch = new\_var;

}

/\*\*

\* Get the value of filterSearch

\* @return the value of filterSearch

\*/

std::string getFilterSearch()

{

return filterSearch;

}

void initAttributes();

};

#endif // CATALOG\_H

Category.cpp

#include "Category.h"

// Constructors/Destructors

//

Category::Category()

{

initAttributes();

}

Category::~Category()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Category::initAttributes()

{

}

Category.h

#ifndef CATEGORY\_H

#define CATEGORY\_H

#include <string>

#include <vector>

/\*\*

\* class Category

\*

\*/

class Category

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Category();

/\*\*

\* Empty Destructor

\*/

virtual ~Category();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void returnAllProperty()

{

}

/\*\*

\*/

void addItem()

{

}

/\*\*

\*/

void removeItem()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

int idCategory;

std::string name;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of idCategory

\* @param new\_var the new value of idCategory

\*/

void setIdCategory(int new\_var)

{

idCategory = new\_var;

}

/\*\*

\* Get the value of idCategory

\* @return the value of idCategory

\*/

int getIdCategory()

{

return idCategory;

}

/\*\*

\* Set the value of name

\* @param new\_var the new value of name

\*/

void setName(std::string new\_var)

{

name = new\_var;

}

/\*\*

\* Get the value of name

\* @return the value of name

\*/

std::string getName()

{

return name;

}

void initAttributes();

};

#endif // CATEGORY\_H

Department.h

#ifndef DEPARTMENT\_H

#define DEPARTMENT\_H

#include <string>

#include <vector>

/\*\*

\* class Department

\*

\*/

class Department

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Department();

/\*\*

\* Empty Destructor

\*/

virtual ~Department();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void viewPeopleList()

{

}

/\*\*

\*/

void addTask()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

std::string worker;

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

/\*\*

\* Set the value of worker

\* @param new\_var the new value of worker

\*/

void setWorker(std::string new\_var)

{

worker = new\_var;

}

/\*\*

\* Get the value of worker

\* @return the value of worker

\*/

std::string getWorker()

{

return worker;

}

private:

// Static Private attributes

//

// Private attributes

//

std::string name;

std::string task;

int peopleCount;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of name

\* @param new\_var the new value of name

\*/

void setName(std::string new\_var)

{

name = new\_var;

}

/\*\*

\* Get the value of name

\* @return the value of name

\*/

std::string getName()

{

return name;

}

/\*\*

\* Set the value of task

\* @param new\_var the new value of task

\*/

void setTask(std::string new\_var)

{

task = new\_var;

}

/\*\*

\* Get the value of task

\* @return the value of task

\*/

std::string getTask()

{

return task;

}

/\*\*

\* Set the value of peopleCount

\* @param new\_var the new value of peopleCount

\*/

void setPeopleCount(int new\_var)

{

peopleCount = new\_var;

}

/\*\*

\* Get the value of peopleCount

\* @return the value of peopleCount

\*/

int getPeopleCount()

{

return peopleCount;

}

void initAttributes();

};

#endif // DEPARTMENT\_H

Empluyee.h

#ifndef EMPLUYEE\_H

#define EMPLUYEE\_H

#include "User.h"

#include <string>

#include <vector>

/\*\*

\* class Empluyee

\*

\*/

class Empluyee : public User

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Empluyee();

/\*\*

\* Empty Destructor

\*/

virtual ~Empluyee();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void lookSalesList()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

std::string salesList;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of salesList

\* @param new\_var the new value of salesList

\*/

void setSalesList(std::string new\_var)

{

salesList = new\_var;

}

/\*\*

\* Get the value of salesList

\* @return the value of salesList

\*/

std::string getSalesList()

{

return salesList;

}

void initAttributes();

};

#endif // EMPLUYEE\_H

Firma.cpp

#include "Firma.h"

// Constructors/Destructors

//

Firma::Firma()

{

initAttributes();

}

Firma::~Firma()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Firma::initAttributes()

{

}

Firma.h

#ifndef FIRMA\_H

#define FIRMA\_H

#include <string>

#include <vector>

/\*\*

\* class Firma

\*

\*/

class Firma

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Firma();

/\*\*

\* Empty Destructor

\*/

virtual ~Firma();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void info()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

std::string name;

int yearOfFounder;

int INN;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of name

\* @param new\_var the new value of name

\*/

void setName(std::string new\_var)

{

name = new\_var;

}

/\*\*

\* Get the value of name

\* @return the value of name

\*/

std::string getName()

{

return name;

}

/\*\*

\* Set the value of yearOfFounder

\* @param new\_var the new value of yearOfFounder

\*/

void setYearOfFounder(int new\_var)

{

yearOfFounder = new\_var;

}

/\*\*

\* Get the value of yearOfFounder

\* @return the value of yearOfFounder

\*/

int getYearOfFounder()

{

return yearOfFounder;

}

/\*\*

\* Set the value of INN

\* @param new\_var the new value of INN

\*/

void setINN(int new\_var)

{

INN = new\_var;

}

/\*\*

\* Get the value of INN

\* @return the value of INN

\*/

int getINN()

{

return INN;

}

void initAttributes();

};

#endif // FIRMA\_H

Payment.cpp

#include "Payment.h"

// Constructors/Destructors

//

Payment::Payment()

{

initAttributes();

}

Payment::~Payment()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Payment::initAttributes()

{

}

Payment.h

#ifndef PAYMENT\_H

#define PAYMENT\_H

#include <string>

#include <vector>

/\*\*

\* class Payment

\*

\*/

class Payment

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Payment();

/\*\*

\* Empty Destructor

\*/

virtual ~Payment();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void pay()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

std::string type;

bool checkPay;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of type

\* @param new\_var the new value of type

\*/

void setType(std::string new\_var)

{

type = new\_var;

}

/\*\*

\* Get the value of type

\* @return the value of type

\*/

std::string getType()

{

return type;

}

/\*\*

\* Set the value of checkPay

\* @param new\_var the new value of checkPay

\*/

void setCheckPay(bool new\_var)

{

checkPay = new\_var;

}

/\*\*

\* Get the value of checkPay

\* @return the value of checkPay

\*/

bool getCheckPay()

{

return checkPay;

}

void initAttributes();

};

#endif // PAYMENT\_H

Property.cpp

#include "Property.h"

// Constructors/Destructors

//

Property::Property()

{

initAttributes();

}

Property::~Property()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void Property::initAttributes()

{

}

Property.h

#ifndef PROPERTY\_H

#define PROPERTY\_H

#include <string>

#include <vector>

/\*\*

\* class Property

\*

\*/

class Property

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

Property();

/\*\*

\* Empty Destructor

\*/

virtual ~Property();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void info()

{

}

/\*\*

\*/

void newProperty()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

int idProperty;

std::string name;

int cost;

int size;

std::string adress;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of idProperty

\* @param new\_var the new value of idProperty

\*/

void setIdProperty(int new\_var)

{

idProperty = new\_var;

}

/\*\*

\* Get the value of idProperty

\* @return the value of idProperty

\*/

int getIdProperty()

{

return idProperty;

}

/\*\*

\* Set the value of name

\* @param new\_var the new value of name

\*/

void setName(std::string new\_var)

{

name = new\_var;

}

/\*\*

\* Get the value of name

\* @return the value of name

\*/

std::string getName()

{

return name;

}

/\*\*

\* Set the value of cost

\* @param new\_var the new value of cost

\*/

void setCost(int new\_var)

{

cost = new\_var;

}

/\*\*

\* Get the value of cost

\* @return the value of cost

\*/

int getCost()

{

return cost;

}

/\*\*

\* Set the value of size

\* @param new\_var the new value of size

\*/

void setSize(int new\_var)

{

size = new\_var;

}

/\*\*

\* Get the value of size

\* @return the value of size

\*/

int getSize()

{

return size;

}

/\*\*

\* Set the value of adress

\* @param new\_var the new value of adress

\*/

void setAdress(std::string new\_var)

{

adress = new\_var;

}

/\*\*

\* Get the value of adress

\* @return the value of adress

\*/

std::string getAdress()

{

return adress;

}

void initAttributes();

};

#endif // PROPERTY\_H

User.cpp

#include "User.h"

// Constructors/Destructors

//

User::User()

{

initAttributes();

}

User::~User()

{

}

//

// Methods

//

// Accessor methods

//

// Other methods

//

void User::initAttributes()

{

}

User.h

#ifndef USER\_H

#define USER\_H

#include <string>

/\*\*

\* class User

\*

\*/

class User

{

public:

// Constructors/Destructors

//

/\*\*

\* Empty Constructor

\*/

User();

/\*\*

\* Empty Destructor

\*/

virtual ~User();

// Static Public attributes

//

// Public attributes

//

// Public attribute accessor methods

//

// Public attribute accessor methods

//

/\*\*

\*/

void authorization()

{

}

protected:

// Static Protected attributes

//

// Protected attributes

//

// Protected attribute accessor methods

//

// Protected attribute accessor methods

//

private:

// Static Private attributes

//

// Private attributes

//

int userId;

std::string password;

std::string loign;

std::string email;

// Private attribute accessor methods

//

// Private attribute accessor methods

//

/\*\*

\* Set the value of userId

\* @param new\_var the new value of userId

\*/

void setUserId(int new\_var)

{

userId = new\_var;

}

/\*\*

\* Get the value of userId

\* @return the value of userId

\*/

int getUserId()

{

return userId;

}

/\*\*

\* Set the value of password

\* @param new\_var the new value of password

\*/

void setPassword(std::string new\_var)

{

password = new\_var;

}

/\*\*

\* Get the value of password

\* @return the value of password

\*/

std::string getPassword()

{

return password;

}

/\*\*

\* Set the value of loign

\* @param new\_var the new value of loign

\*/

void setLoign(std::string new\_var)

{

loign = new\_var;

}

/\*\*

\* Get the value of loign

\* @return the value of loign

\*/

std::string getLoign()

{

return loign;

}

/\*\*

\* Set the value of email

\* @param new\_var the new value of email

\*/

void setEmail(std::string new\_var)

{

email = new\_var;

}

/\*\*

\* Get the value of email

\* @return the value of email

\*/

std::string getEmail()

{

return email;

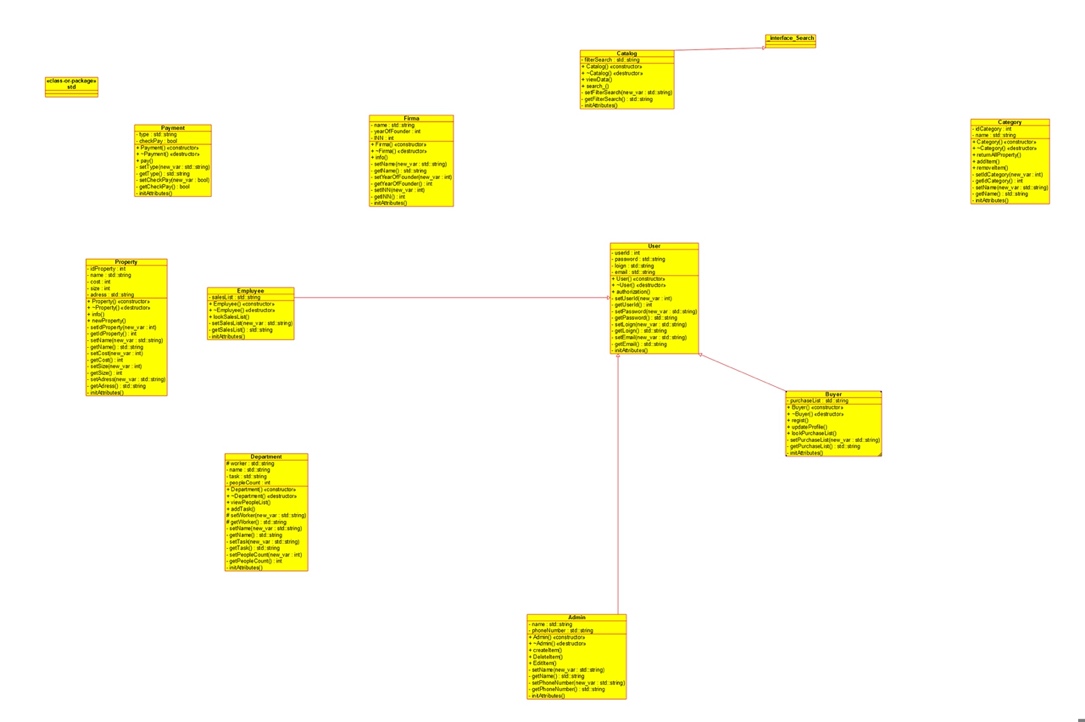
}

void initAttributes();

};

#endif // USER\_H

1. Диаграмма классов при обратном проектировании:



1. Выводы по работе:

В ходе выполнения работы были получены навыки генерации кода для UML-модели и анализ программного обеспечения через обратное проектирование.

1. Использованные источники:

Язык UML. Руководство пользователя — Гради Буч, Джеймс Рамбо, Ивар Якобсон

# UML. Основы — Мартин Фаулер