МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Череповецкий государственный университет»

Институт информационных технологий

Кафедра: Математического и программного

Обеспечения ЭВМ

Дисциплина: Основы программной инженерии

**Лабораторная работа № 1**

**«КОНСТРУИРОВАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ ПРОГРАММ»**

**Выполнил:**

**студент гр.** 1ИВТпб-01-31оп

Климов А.Г.  
**Проверил: преподаватель**

Ершов Е.В.  
**Отметка о зачете:**

Череповец

2017 год

**Задание:** Для основных понятий предметной области “Осветительные приборы” необходимо:

1) Разработать модель реализации;

2) Выполнить конструирование и реализацию ассоциируемых классов;

3) Выполнить моделирование и реализацию взаимодействия объектов классов.

**1. Разработка модели реализации**

Разработанная модель реализации, для предметной области “Осветительные приборы” состоит из трёх классов:

* user (пользователь);
* light (осветительный прибор);
* lightbulb (лампочка).

Пользователь “user” имеет имя, а также два статуса, которые меняются в зависимости от произведённых действий пользователем над осветительным прибором и лампочкой.

Осветительный прибор “light” имеет имя, а также два статуса – самого осветительного прибора и лампочки, которые меняются в зависимости от произведённых действий над ними.

Лампочка “lightbulb” имеет имя, а также имеет два статуса, которые показывают изменения, совершённые пользователем “user” и осветительным прибором “light” над ней.

Зависимости классов:

* пользователь “user” может включать либо выключать осветительный прибор, а также вкручивать или выкручивать лампочку;
* осветительный прибор “light” включает либо выключает осветительный прибор, а также освещает помещение;
* лампочка “lightbulb” оповещает пользователя, с помощью индикатора, вкручена или выкручена, а также оповещает осветительный прибор о своей работе.

Значения статусов классов по умолчанию:

* user (пользователь) – пользователь не вкрутил лампочку и не включил осветительный прибор;
* light (осветительный прибор) – прибор выключен и лампочка выключена;
* lightbulb (лампочка) – лампочка выключена и выкручена.

UML диаграмма, для разработанной модели изображена на рис. 1.

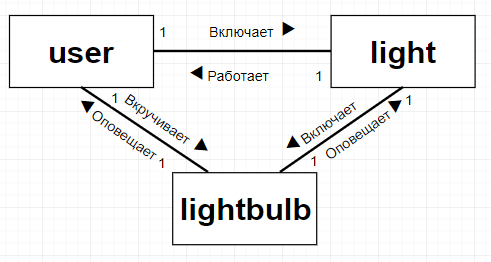


Рис. 1. UML диаграмма “Осветительные приборы”

**2. Конструирование и реализация ассоциированных классов**

В табл. 1, табл. 2 и табл. 3. произведено конструирование, а так же произведена реализация ассоциированных классов.

Условные обозначения для табл. 1, табл. 2 и табл. 3:

* public “+”;
* private “-”;
* protected “#”.

Таблица 1. Класс “user”

|  |
| --- |
| **user** |
| +user(); //конструктор по умолчанию  +user(char \*i, int j, int k); // конструктор с параметрами  +void destr\_user(); // деструктор  +void print\_user(); // вывод пользователя  +void change\_user\_status\_1(int n); // изменить 1-ый статус пользователя  +void change\_user\_status\_2(int n); // изменить 2-ой статус пользователя  +void light\_turn\_on(light &m1); // включить осветительный прибор  +void light\_turn\_off(light &m1); // выключить осветительный прибор  +void lightbulb\_screw\_in(lightbulb &n1); // вкрутить лампочку  +void lightbulb\_unscrew(lightbulb &n1); // выкрутить лампочку |
| -char \*username; // имя пользователя  -int user\_status\_1; // 0 – пользователь не вкрутил лампочку / 1 – пользователь вкрутил лампочку  -int user\_status\_2; // 0 – пользователь не включил осветительный прибор / 1 – пользователь включил осветительный прибор |

Таблица 2. Класс “light”

|  |
| --- |
| **light** |
| +light(); //конструктор по умолчанию  +light(char \*i1, int j1, int k); // конструктор с параметрами  +void destr\_light(); // деструктор  +void print\_light(); // вывод осветительного прибора  +void change\_light\_status(int n); // изменить статус прибора  +void turn\_on(); // включить осветительный прибор  +void turn\_off(); // выключить осветительный прибор  +void lightbulb\_turn\_on(lightbulb &m2); // включить лампочку  +void lightbulb\_turn\_off(lightbulb &m2); // выключить лампочку  + void light\_status\_show\_to\_user(user &n3); // пользователь видит включен прибор или выключен |
| -char \*lightname; // название осветительного прибора  -int light\_status; 0 – прибор выключен / 1 – прибор включен  -int light\_status0; // 0 – лампочка выключена / 1 – лампочка включена |

Таблица 3. Класс “lightbulb”

|  |
| --- |
| **lightbulb** |
| +lightbulb();//конструктор по умолчанию  +lightbulb(char \*i2, int j2, int k2); // конструктор с параметрами  +void destr\_light(); // деструктор  +void print\_lightbulb(); // вывод лампочки  +void turn\_on\_lb(); // включить лампочку  +void turn\_off\_lb(); // выключить лампочку  +void screw\_in();// вкрутить лампочку  +void unscrew(); // выкрутить лампочку  +void lightbulb\_status\_show\_to\_light(light &n3); // прибор видит включена лампочка или выключена  +void lightbulb\_status\_show\_to\_user(user &n3); // пользователь видит вкручена лампочка или выкручена |
| -char \*lightbulbname; // название лампочки  -int lightbulb\_status\_1; // 0 – лампочка не горит / 1 – лампочка горит  -int lightbulb\_status\_2; // 0 – лампочка выкручена / 1 – лампочка вкручена |

**3. Моделирование и реализация взаимодействия объектов классов**

**Описание взаимодействия объектов классов:**

1. Пользователь вкручивает лампочку и у неё меняется статус:

u1.lightbulb\_screw\_in(lb1);

2. Пользователь видит, что лампочка вкручена:

lb1.lightbulb\_status\_show\_to\_user(u1);

3. Пользователь включает прибор и у него меняется статус:

u1.light\_turn\_on(l1);

4. Пользователь видит, что осветительный прибор работает:

l1.light\_status\_show\_to\_user(u1);

5. Прибор включает лампочку и у неё меняется статус:

l1.lightbulb\_turn\_on(lb1);

6. Прибор видит, что лампочка включена:

lb1.lightbulb\_status\_show\_to\_light(l1);

**Текст программы:**

**main.cpp**

#include "stdafx.h"

#include <iostream>

#include "user.h"

#include "light.h"

#include "lightbulb.h"

int main()

{

lightbulb lb1;

light l1;

user u1;

lb1.print\_lightbulb();

l1.print\_light();

u1.print\_user();

std::cout << std::endl;

u1.lightbulb\_screw\_in(lb1); // пользователь вкручивает лампочку и у неё меняется статус

lb1.lightbulb\_status\_show\_to\_user(u1); // пользователь видит, что лампочка вкручена

lb1.print\_lightbulb();

u1.print\_user();

std::cout << std::endl;

u1.light\_turn\_on(l1); // пользователь включает прибор и у него меняется статус

l1.light\_status\_show\_to\_user(u1); // пользователь видит, что осветительный прибор работает

l1.print\_light();

u1.print\_user();

std::cout << std::endl;

l1.lightbulb\_turn\_on(lb1); // прибор включает лампочку и у неё меняется статус

lb1.lightbulb\_status\_show\_to\_light(l1); // прибор видит, что лампочка включена

l1.print\_light();

lb1.print\_lightbulb();

system("pause");

return 0;

}

**user.h**

#ifndef USER\_H\_INCLUDED

#define USER\_H\_INCLUDED

#include <iostream>

#include "light.h"

#include "lightbulb.h"

class user

{

private:

char \*username;

int user\_status\_1; // пользователь не вкрутил лампочку или вкрутил (0/1)

int user\_status\_2; // пользователь не включил осветительный прибор или включил (0/1)

public:

user();

user(char \*i, int j, int k);

void destr\_user() { delete[]username; }

void print\_user();

void change\_user\_status\_1(int n);

void change\_user\_status\_2(int n);

void light\_turn\_on(light &m1);

void light\_turn\_off(light &m1);

void lightbulb\_screw\_in(lightbulb &n1);

void lightbulb\_unscrew(lightbulb &n1);

};

#endif //MY\_SYMBOL\_H

**light.h**

#ifndef LIGHT\_H\_INCLUDED

#define LIGHT\_H\_INCLUDED

#include <iostream>

#include "lightbulb.h"

#include "user.h"

class light

{

private:

char \*lightname;

int light\_status; // прибор выключен или включен (0/1)

int light\_status0; // лампочка выключена или включена (0/1)

public:

light();

light(char \*i1, int j1, int k1);

void destr\_light() { delete[]lightname; }

void print\_light();

void change\_light\_status(int n);

void turn\_on();

void turn\_off();

void lightbulb\_turn\_on(lightbulb &n2);

void lightbulb\_turn\_off(lightbulb &n2);

void light\_status\_show\_to\_user(user &n3);

};

#endif //MY\_SYMBOL\_H

**lightbulb.h**

#ifndef LIGHTBULB\_H\_INCLUDED

#define LIGHTBULB\_H\_INCLUDED

#include <iostream>

#include "user.h"

#include "light.h"

class user;

class light;

class lightbulb;

class lightbulb

{

private:

int lightbulb\_status\_2; // выкручена лампочка или вкручена (0/1)

int lightbulb\_status\_1; // не горит лампочка или горит (0/1)

char \*lightbulbname;

public:

lightbulb();

lightbulb(char \*i2, int j2, int k2);

void destr\_light() { delete[]lightbulbname; }

void print\_lightbulb();

void turn\_on\_lb();

void turn\_off\_lb();

void screw\_in();

void unscrew();

void lightbulb\_status\_show\_to\_light(light &n3);

void lightbulb\_status\_show\_to\_user(user &n3);

};

#endif //MY\_SYMBOL\_H

**user.cpp**

#include "user.h"

#include "light.h"

#include "lightbulb.h"

user::user()

{

username = "user1";

user\_status\_1 = 0;

user\_status\_2 = 0;

}

user::user(char \*i, int j, int k)

{

username = new char[strlen(i) + 1];

strcpy(username, i);

user\_status\_1 = j;

user\_status\_2 = k;

}

void user::print\_user()

{

std::cout << username << " " << user\_status\_1 << " " << user\_status\_2 << std::endl;

}

void user::change\_user\_status\_1(int n)

{

user\_status\_1 = n;

}

void user::change\_user\_status\_2(int n)

{

user\_status\_2 = n;

}

void user::light\_turn\_on(light &m1)

{

m1.turn\_on();

}

void user::light\_turn\_off(light &m1)

{

m1.turn\_off();

}

void user::lightbulb\_screw\_in(lightbulb &n1)

{

n1.screw\_in();

}

void user::lightbulb\_unscrew(lightbulb &n1)

{

n1.unscrew();

}

**light.cpp**

#include "user.h"

#include "light.h"

#include "lightbulb.h"

light::light()

{

lightname = "light1";

light\_status = 0;

light\_status0 = 0;

}

light::light(char \*i1, int j1, int k1)

{

lightname = new char[strlen(i1) + 1];

strcpy(lightname, i1);

light\_status = j1;

light\_status0 = k1;

}

void light::print\_light()

{

std::cout << lightname << " " << light\_status <<" "<<light\_status0<< std::endl;

};

void light::turn\_on()

{

light\_status = 1;

}

void light::turn\_off()

{

light\_status = 0;

}

void light::lightbulb\_turn\_on(lightbulb &n2)

{

n2.turn\_on\_lb();

}

void light::lightbulb\_turn\_off(lightbulb &n2)

{

n2.turn\_off\_lb();

}

void light::light\_status\_show\_to\_user(user &n3)

{

n3.change\_user\_status\_2(light\_status);

}

void light::change\_light\_status(int n)

{

light\_status0 = n;

}

**lightbulb.cpp**

#include "user.h"

#include "light.h"

#include "lightbulb.h"

lightbulb::lightbulb()

{

lightbulbname = "lightbulb1";

lightbulb\_status\_1 = 0;

lightbulb\_status\_2 = 0;

}

lightbulb::lightbulb(char \*i2, int j2, int k2)

{

lightbulbname = new char[strlen(i2) + 1];

strcpy(lightbulbname, i2);

lightbulb\_status\_1 = j2;

lightbulb\_status\_2 = k2;

}

void lightbulb::print\_lightbulb()

{

std::cout << lightbulbname << " " << lightbulb\_status\_2 << " " << lightbulb\_status\_1 << std::endl;

};

void lightbulb::turn\_on\_lb()

{

lightbulb\_status\_1 = 1;

}

void lightbulb::turn\_off\_lb()

{

lightbulb\_status\_1 = 0;

}

void lightbulb::screw\_in()

{

lightbulb\_status\_2 = 1;

}

void lightbulb::unscrew()

{

lightbulb\_status\_2 = 0;

}

void lightbulb::lightbulb\_status\_show\_to\_light(light &n3)

{

n3.change\_light\_status(lightbulb\_status\_1);

}

void lightbulb::lightbulb\_status\_show\_to\_user(user &n3)

{

n3.change\_user\_status\_1(lightbulb\_status\_2);

}

**Тестирование**

1. Пользователь вкручивает лампочку и видит, что лампочка вкручена. Результат на рис. 2.

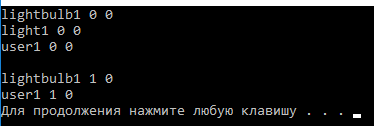


Рис. 2. Результат тестирования 1.

2. Пользователь включает осветительный прибор и видит, что прибор включен. Результат на рис. 3.

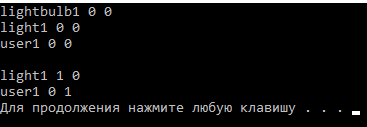


Рис. 3. Результат тестирования 2.

3. Осветительный прибор включает лампочку и показывает, что лампочка включена. Результат на рис. 4.

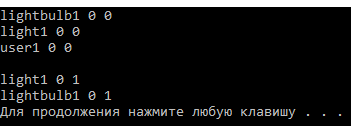


Рис. 4. Результат тестирования 3.

4. Результат последовательного выполнения команд, описанных ранее, изображен на рис. 5.

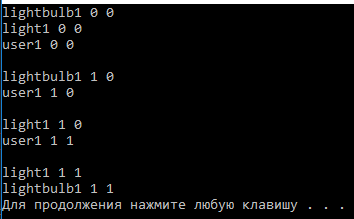


Рис. 5. Результат тестирования 4.