

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего  
профессионального образования  
"Донской государственный технический университет"  
ДГТУ

Кафедра "Программное обеспечение вычислительной техники и  
автоматизированных систем"

Статические поля и методы класса. Друзья класса.

Методические указания к лабораторной работе по дисциплине  
"Объектно-ориентированное программирование"

Ростов-на-Дону

20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.

УДК 512.3

Статические поля и методы класса. Друзья класса: методические указания –  
Ростов н/Д: Издательский центр ДГТУ, 20 . – с.

В методической разработке рассматриваются поля и методы класса в целом и дружественные функции, методы, классы. Даны задания по выполнению лабораторной работы. Методические указания предназначены для студентов направления 231000 "Программная инженерия".

© Издательский центр ДГТУ, 20

## I. Статические поля класса в C++.

Если в классе созданы поля, то каждый объект класса будет обладать собственными копиями всех полей, поэтому такие поля называют полями экземпляра класса. Можно объявлять такие поля, которые будут принадлежать классу в целом. Такие поля будут существовать в единственном экземпляре независимо от того, сколько объектов класса создано. Они будут существовать, даже если нет ни одного объекта класса. Чтобы объявить поле класса надо задать при его описании модификатор `static`. Такое поле будет храниться в сегменте данных программы, вместе с глобальными и статическими переменными. Кроме того, C++ требует, чтобы эти поля были явно созданы вне описания класса. По умолчанию статические поля инициализируются нулевыми значениями.

Например,

```
class A {
public:
    int a1;          /* описание поля экземпляра класса */
    static int a2;    /* описание поля класса в целом */
    static int a3;    /* описание поля класса в целом */
};

/* объявление статического поля, инициализация нулем по умолчанию */
int A::a2;

int A::a3 = 100; /* объявление статического поля, явная инициализация */

int main() {
    /* статическое поле существует, даже когда нет ни одного объекта класса */
    A::a2 = -1;
    A x; /* объект класса */
}
```

```

        x.a2 = -10; /* обращение к статическому полю через объект */
        return 0;
    }

```

## II. Статические методы класса в C++.

Методы, объявленные в классе, могут быть вызваны только для какого-нибудь объекта этого класса ("привязаны" к объекту), поэтому их называют методами экземпляра класса. Адрес объекта, для которого вызван метод, передается последнему через скрытый параметр и доступен в теле метода как указатель с именем `this`. Если же при объявлении метода задан модификатор `static`, то такой метод не будет привязан к объекту класса, а станет методом класса в целом. Так как при вызове статического метода может не задаваться объект, у него не может быть скрытого аргумента – адреса объекта и, поэтому, в теле такого метода не определено имя `this`. С другой стороны, статический метод можно вызывать даже тогда, когда нет ни одного объекта класса. Например,

```

class A {
    int a1;
    static int a2;
public:
    A(int x) { a1 = x; }
    static void stFun() {
        a1 = 1; /* Ошибка. Нет this, поэтому нет доступа к нестатическим полям */
        a2 = 1; /* Есть доступ */
    }
};

int A::a2;

int main() {

```

```

A::stFun(); /* можно вызвать, хотя нет ни одного объекта A */
A x(10);
x.stFun(); /* так тоже можно */
return 0;
}

```

### III. Статические поля и методы в Java.

Статические поля и методы в Java имеют тот же смысл, что и в C++. Но в Java статические поля не только описываются, но и определяются внутри описания класса. Например,

```

public class A {
    public static int a = 20;
}

```

Кроме того, для инициализации статических полей в Java можно использовать статические блоки инициализации. Блоки инициализации – это блоки кода, которые могут содержать также локальные описания. Например,

```

public class A { public static int[] a;
    static { /* статический блок инициализации */
        a = new int[20];
        for(int i=0; i<20; ++i) a[i] = i + 1;
    }
}

```

Блоков инициализации может быть несколько. Компилятор объединяет их в один блок в том порядке, в котором они заданы в описании класса. Этот блок выполняется при загрузке класса.

В Java нет внешних функций, поэтому точкой входа в программу должен быть статический метод (main), чтобы его можно было вызвать тогда, когда еще нет ни одного объекта.

```
public class A {  
    public static void main(String[] args) {  
        System.out.println("Hello!");  
    }  
}
```

#### IV. Дружественные функции и дружественные классы.

Если функция объявлена в классе дружественной, то она получает такие же привилегии что и методы самого класса, т.е. получает доступ не только к открытой, но и к защищенной и закрытой частям класса. Чтобы воспользоваться этими привилегиями функция должна каким-то образом получить объект класса: в качестве аргумента или создавая такой объект во время своей работы, т.к. скрытого параметра this у нее нет. Например,

```
class A {  
    friend void fun(A x); // объявление внешней функции другом класса  
    int a; // закрытое поле класса      public:  
    A(int x) { a = x; }  
};  
void fun(A x) { // реализация функции  
    cout<<"a="<<x.a // доступ к закрытому полю  
    <<endl;  
}
```

Дружественной функцией может быть объявлен метод другого класса, и даже другой класс в целом. В последнем случае все методы класса становятся дружественными функциями. Например,

```
class A; // предварительное описание класса
```

```

class B {
public:
    void f(A& x);          // доступ к закрытому полю A
};
class C {};
class A {
    friend void B::f(A&); // объявление дружественного метода
    friend class C; // объявление дружественного класса
    int a;
public:
    A(int x) { a = x; }
};
void B::f(A&) { cout<<x.a<<endl; }

int main() {
    A aObj(5);
    B bObj;
    bObj.f( aObj );
    return 0;
}

```

#### IV. Задания.

1. Создайте класс, который содержит счетчик созданных объектов. Напишите программу-тест, в которой проверяется, сколько объектов класса создано при

входе в функцию main, после статического создания массива объектов, после динамического создания объекта, после удаления динамического объекта.

2. Создайте класс с закрытыми конструкторами и деструктором. Реализуйте **методы** для создания и уничтожения объектов класса. Напишите программу-тест.

3. Создайте класс с закрытыми конструкторами и деструктором. Реализуйте **внешние функции** для создания и уничтожения объектов класса. Напишите программу-тест.

4. Паттерн проектирования «Одиночка» - Singleton. Создайте класс, у которого может быть только один объект (экземпляр). Реализуйте две версии: 1) объект существует с того момента, когда программа начинает выполняться, его нельзя уничтожить или заменить другим объектом, т.е. всегда существует один и только один объект класса; 2) изначально нет ни одного объекта класса, при первом запросе на получение объекта он создается, последующие запросы получают этот созданный объект, объект можно уничтожить и создать (в общем случае уже другой) заново, т.е. в любой момент времени выполнения программы существует не более одного объекта класса.

## V. Контрольные вопросы.

1. В чем отличие статических и нестатических полей?
2. Где размещаются статические поля класса? Нестатические поля?
3. Как можно проинициализировать статическое поле в C++? 4. Как можно проинициализировать статическое поле в Java?
4. Когда выполняются статические блоки инициализации?
5. Для решения каких задач можно использовать статические поля?
6. Чем статический метод отличается от нестатического?



7. Можно ли в теле статического метода получить доступ к нестатическим полям класса?
8. Можно ли в теле статического метода вызвать нестатический метод класса?
9. Может ли точка входа в Java-приложение быть нестатическим методом? Почему?
10. Что такое дружественная функция?
11. Можно ли объявить некоторую функцию дружественной вне описания класса? Почему?
12. Что такое дружественный класс?

#### *Литература*

1. Brian W. Kernighan, Dennis M. Ritchie. "The C programming language". Second edition – Prentice Hall. – 1988, 272 p.
2. Стэнли Б. Липпман. "Язык программирования C++. Вводный курс". – М.: Вильямс. – 2007, 896 с.
3. Г. Шилдт. "Самоучитель C++". – СПб.: БХВ-Петербург. – 2001, 688 с.
4. К.Арнольд, Дж.Гослинг, Д.Холмс. "Язык программирования Java". 3-е изд. – М.: Вильямс. – 2001, 624 с.