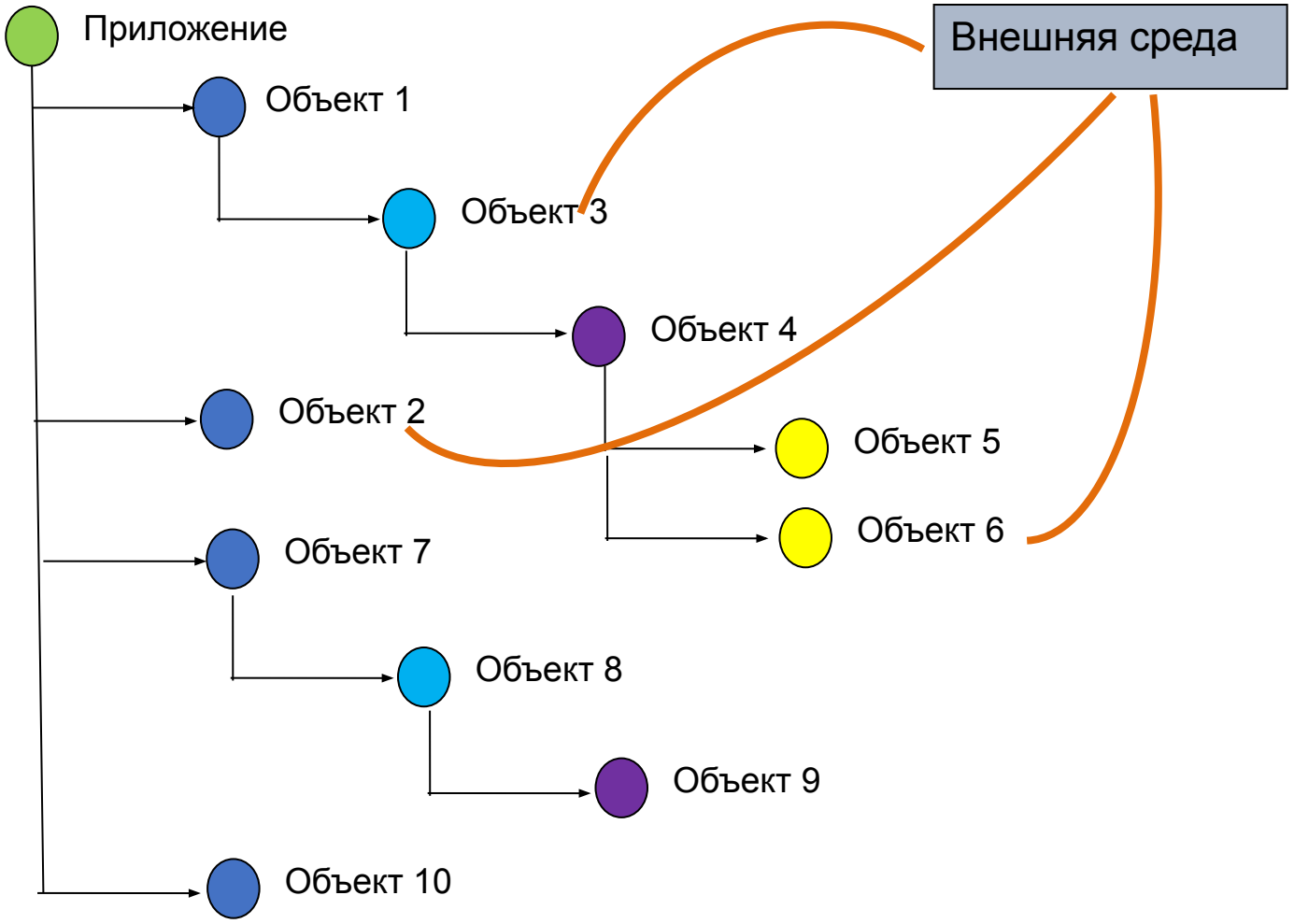
Объектно-ориентированное программирование на алгоритмическом языке C++

МИРЭА, Институт Информационных технологий, кафедра Вычислительной техники

Схема архитектуры программы
Дерево объектов



Отчет по лабораторной работе

Отчет состоит из:

1. Титульного листа лабораторной работы.
2. Содержания.

Содержание

1. Постановка задачи.
2. Методы и объекты.
3. Архитектура программы-системы.
   - 3.1. Иерархия объектов.
   - 3.2. Взаимодействие объектов.
   - 3.3. Алгоритм функционирования системы, решение задачи.

Отчет по лабораторной работе

4. Схемы.
    - 4.1. Схема иерархии наследования классов.
    - 4.2. Схема архитектуры программы.
    - 4.3. Схема взаимодействия объектов.
    - 4.4. Схема алгоритма решения задачи
5. Код программы.
    - 5.1. Код описания классов .
    - 5.2. Код конструирования системы.
    - 5.3. Код взаимодействия объектов.
    - 5.4. Код алгоритма решения задачи.
6. Тестирование.

Пояснения к разделу «Методы и объекты»

№	Объект	Пояснения
1	Основной объект (приложение)	Объект является корневым в дереве иерархии объектов. Содержит 4 объекта второго уровня, 2 объекта третьего уровня и один объект простого третьего уровня Организует построение иерархии объектов. Выполняет проверку готовности объектов к работе. Класс: cl_application
2	Объект второго уровня	Определяет готовность к работе объекта. Класс: cl_2
3	Простой объект третьего уровня	Определяет готовность к работе объекта. Класс: cl_p3
4	Объект третьего уровня	Содержит объект четвертого уровня. Определяет готовность к работе объекта. Класс: cl_3
5	Объект четвертого	Определяет готовность к работе объекта.

Пояснения к пункту «Иерархия объектов»

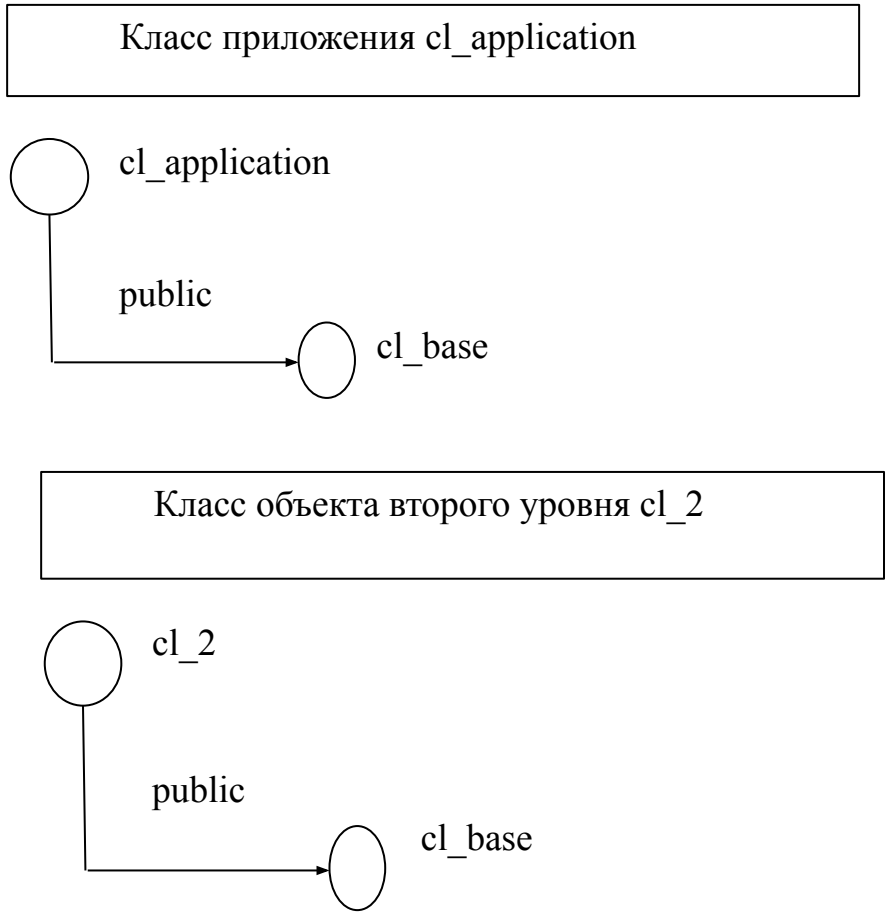
Объект	Объекты в составе	Пояснения	Номер
1 Имя: ob_application		Объект приложение. В иерархии объектов является корневым.	
	Второго уровня		2
	Второго уровня		3
	Второго уровня		4
	Второго уровня		5
	Ввода	Создается компилятором. Связан с другими объектами программы интерфейсом «cin»	
	Вывода	Создается компилятором. Связан с другими объектами программы интерфейсом «cout»	
2 Имя: ob_1	Третьего уровня		6
6 Имя: ob_2	Четвертого уровня		7
7 Имя: ob_4			
3 Имя: ob_3	Простой третьего уровня		8
8 Имя: ob_5			

Пояснения к пункту «Алгоритм решение задачи»

Алгоритм реализован в методе show_state_next (ob_base * ob_parent)

Имя объекта или пункт алгоритма	Предикат	Процедура	Номер перехода
1 **ob_application**	Проверка готовности объекта ob_parent. Объект готов к работе	Вывод сообщения на консоль: Объект «наименование объекта» готов к работе	2
	Объект не готов к работе	Вывод сообщения на консоль: Объект «наименование объекта» не готов к работе	2
2	Если у объекта ob_parent нет подчиненных объектов		∅
		Начало цикла по подчиненным объектам объекта ob_parent	3
3	Цикл не завершен	Вызов метода show_state_next (ссылка на очередной подчиненный объект)	3
			∅

Схема иерархии наследования классов



Иерархия классов

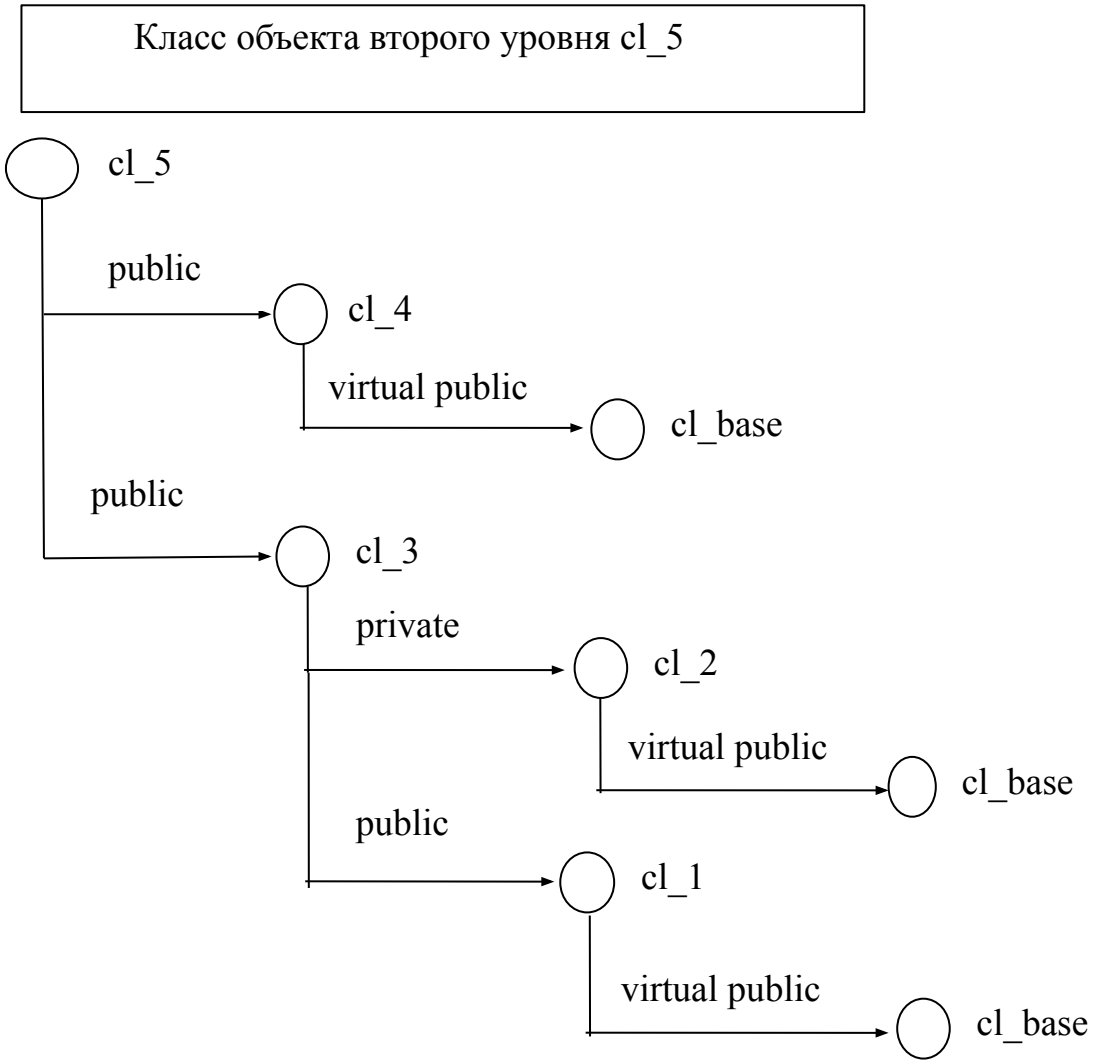
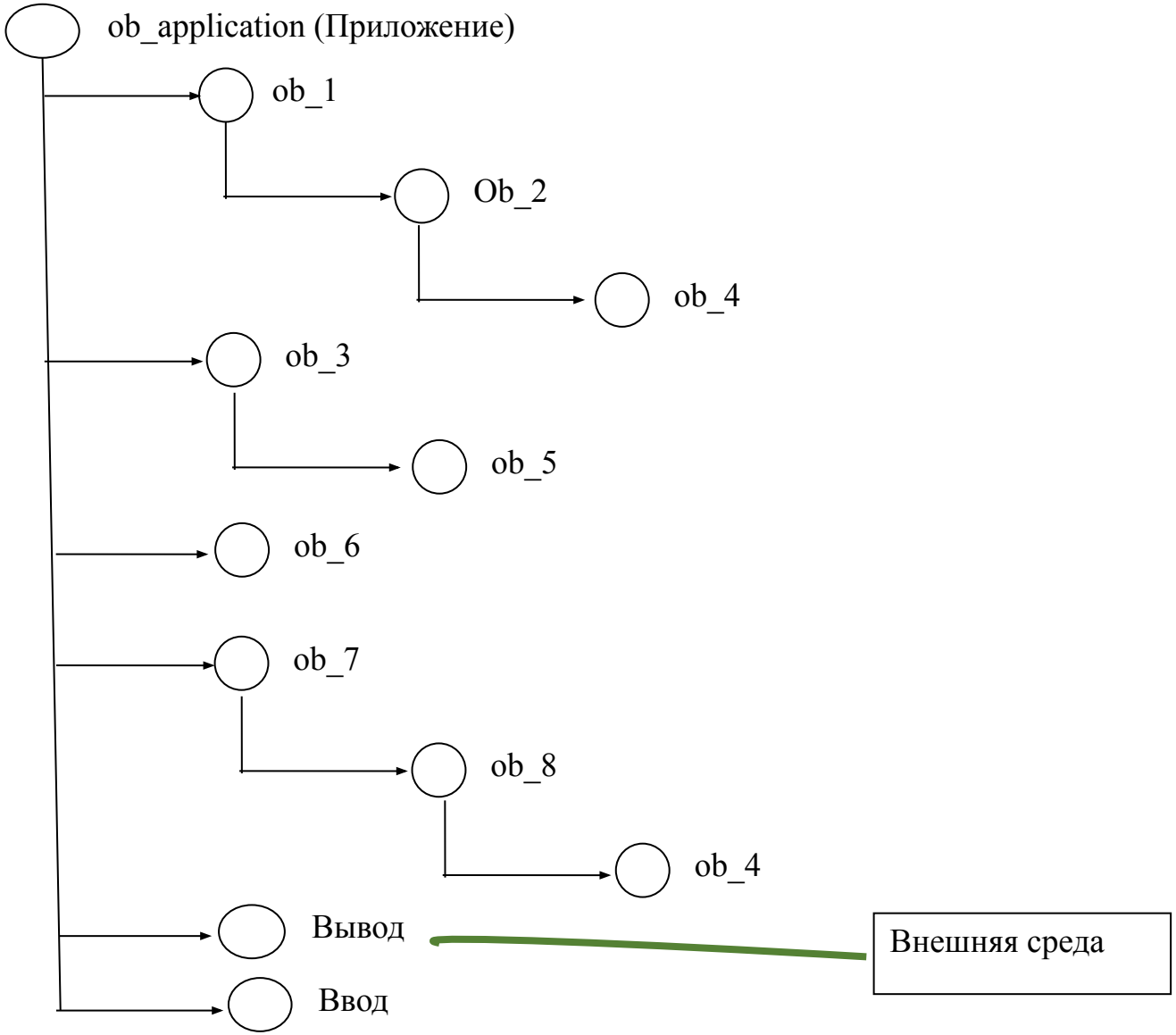


Схема архитектуры программы



Тестирование

№	Входные данные	Фактические выходные данные	Ожидаемые выходные данные
1	Во всех конструкторах объектов задать set_state (1);	Объект root готов к работе Объект ob_1 готов к работе Объект ob_2 готов к работе Объект ob_4 готов к работе Объект ob_3 готов к работе Объект ob_5 готов к работе Объект ob_6 готов к работе Объект ob_7 готов к работе Объект ob_8 готов к работе Объект ob_4 готов к работе	Объект root готов к работе Объект ob_1 готов к работе Объект ob_2 готов к работе Объект ob_4 готов к работе Объект ob_3 готов к работе Объект ob_5 готов к работе Объект ob_6 готов к работе Объект ob_7 готов к работе Объект ob_8 готов к работе Объект ob_4 готов к работе
2	В классе cl_4 в конструкторе объекте задать set_state (0);	Объект root готов к работе Объект ob_1 готов к работе Объект ob_2 готов к работе Объект ob_4 не готов к работе Объект ob_3 готов к работе Объект ob_5 готов к работе Объект ob_6 готов к работе Объект ob_7 готов к работе Объект ob_8 готов к работе Объект ob_4 не готов к работе	Объект root готов к работе Объект ob_1 готов к работе Объект ob_2 готов к работе Объект ob_4 не готов к работе Объект ob_3 готов к работе Объект ob_5 готов к работе Объект ob_6 готов к работе Объект ob_7 готов к работе Объект ob_8 готов к работе Объект ob_4 не готов к работе

Строки

#include <string>
string – коллекция символы char в формате ASCII,

#include <xstring>
wstring –коллекция двухбайтных символов wchar_t, в формате Unicode.

string s_1;

Векторы

#include <vector>
vector – коллекция однотипных переменных,

vector < тип данных > «имя вектора»

Итератор
vector < тип данных > :: iterator «имя итератора»

Примерная заголовочная часть базового класса

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class cl_base   {
public:
    cl_base ( cl_base * p_parent = 0 );

    void      set_object_name ( string    object_name );
    string    get_object_name ( );
    void      set_parent      ( cl_base * p_parent    );
    void      add_child       ( cl_base * p_child     );
    void      delete_child    ( string    object_name );
    cl_base * get_child       ( string    object_name );
    cl_base * get_object      ( string    object_path );

    vector < cl_base * > children;        // ссылки на потомков
    vector < cl_base * > :: iterator  it_child;
 private:
    string       object_name;             // наименование объекта
    cl_base    * p_parent;                // ссылка на головной объект
};
```

Примерная часть реализации базового класса

```
cl_base :: cl_base ( cl_base * p_parent )
{
    set_object_name ( "cl_base" );

    if ( p_parent ) {
        this -> p_parent = p_parent;
        p_parent -> add_child ( this );
    }
}

void cl_base :: set_object_name ( string object_name )
{
    this -> object_name = object_name;
}

void cl_base :: set_parent ( cl_base * p_parent )
{
    if ( p_parent ) {
        this -> p_parent = p_parent;
        p_parent -> add_child ( this );
    }
}
```

Примерная часть реализации базового класса

```
void cl_base :: add_child ( cl_base * p_child )
{
    children.push_back ( p_child );
}

void cl_base :: delete_child  ( string object_name )
{
    if ( children.size ( ) == 0 ) return;

    it_child = children.begin ( );

    while ( it_child != children.end ( ) ) {

        if ( ( * it_child ) -> get_object_name ( ) == object_name ) {

            children.erase ( it_child );
            return;
        }
        it_child ++;
    }
}
```

Примерная часть реализации базового класса

```
cl_base * cl_base :: get_child ( string  object_name )
{
    if ( children.size ( ) == 0 ) return 0;

    it_child = children.begin ( );

    while ( it_child != children.end ( ) ) {

        if ( ( * it_child ) -> get_object_name ( ) == object_name ) {

            return ( * it_child );
        }
        it_child ++;
    }

    return 0;
}
```

Код класса cl_application

```
#include "cl_base.h"

class cl_application : public cl_base
{
public:
    cl_application ( );

    void bild_tree_objects ( );
    int  exec_app          ( );

    void show_object_tree  ( );

private:
    void show_object_next ( cl_base * ob_parent,
int i_level );
};
```

Код метода show_state_next (cl_base * ob_parent)

```
  void cl_application :: show_object_state ( )   {
    show_state_next ( this );
}

void cl_application :: show_state_next ( cl_base * ob_parent )   {
    if ( ob_parent -> get_state ( ) == 1 ) {
        cout << "The object " << ob_parent -> get_object_name ( ) << " is ready" << endl;
    }
    else {
        cout << "The object " << ob_parent -> get_object_name ( ) << " is not ready" << endl;
    }

    if ( ob_parent -> children.size ( ) == 0 ) return;

    ob_parent -> it_child = ob_parent -> children.begin ( );

    while ( ob_parent -> it_child != ob_parent -> children.end ( ) ) {

        show_state_next ( ( * ( ob_parent -> it_child ) ) );
        ob_parent -> it_child ++;
    }
}
```

Вывод дерева иерархии объектов

```
void cl_base :: show_object_tree ( ) {
    int i_level = 0;
    show_object_next ( this, i_level );
}
void cl_base :: show_object_next ( cl_base * ob_parent, int i_level ) {
    string    s_space;
    //-------------------------------------------------------------
    if ( i_level > 0 )  s_space.append ( 4 * i_level, ' ' );
    cout << s_space << ob_parent -> get_object_name ( ) << endl;

    if ( ob_parent -> children.size ( ) == 0 ) return;

    ob_parent -> it_child = ob_parent -> children.begin ( );

    while ( ob_parent -> it_child != ob_parent -> children.end ( ) ) {

        show_object_next ((cl_base*)( * ( ob_parent -> it_child )), i_level + 1 );
        ob_parent -> it_child ++;
    }
}
```

Лабораторные работы

1. Создание базового класса объекта.
2. Создание класса приложения.
3. Конструктивное построение приложения.
   - Вывод на консоль дерева иерархии объектов.
   - Вывод на консоль ветки дерева иерархии объектов.
   - Динамическое добавление объекта в дереве иерархии.
   - Динамическое удаление объекта из дерева иерархии с последующей веткой.
   - Копирование ветки дерева иерархии.
   - Вставка ветки дерева иерархии.
4. Координаты объекта на дереве иерархии, по аналогии XPath.
   - Метод получения ссылки на объект исходя из координаты (пути) объекта в дереве иерархии.
   - Метод получения ссылки на объект исходя из относительной координаты (пути) объекта в дереве иерархии.
5. Объекты ввода и вывода.
6. Простые сигналы между объектами.