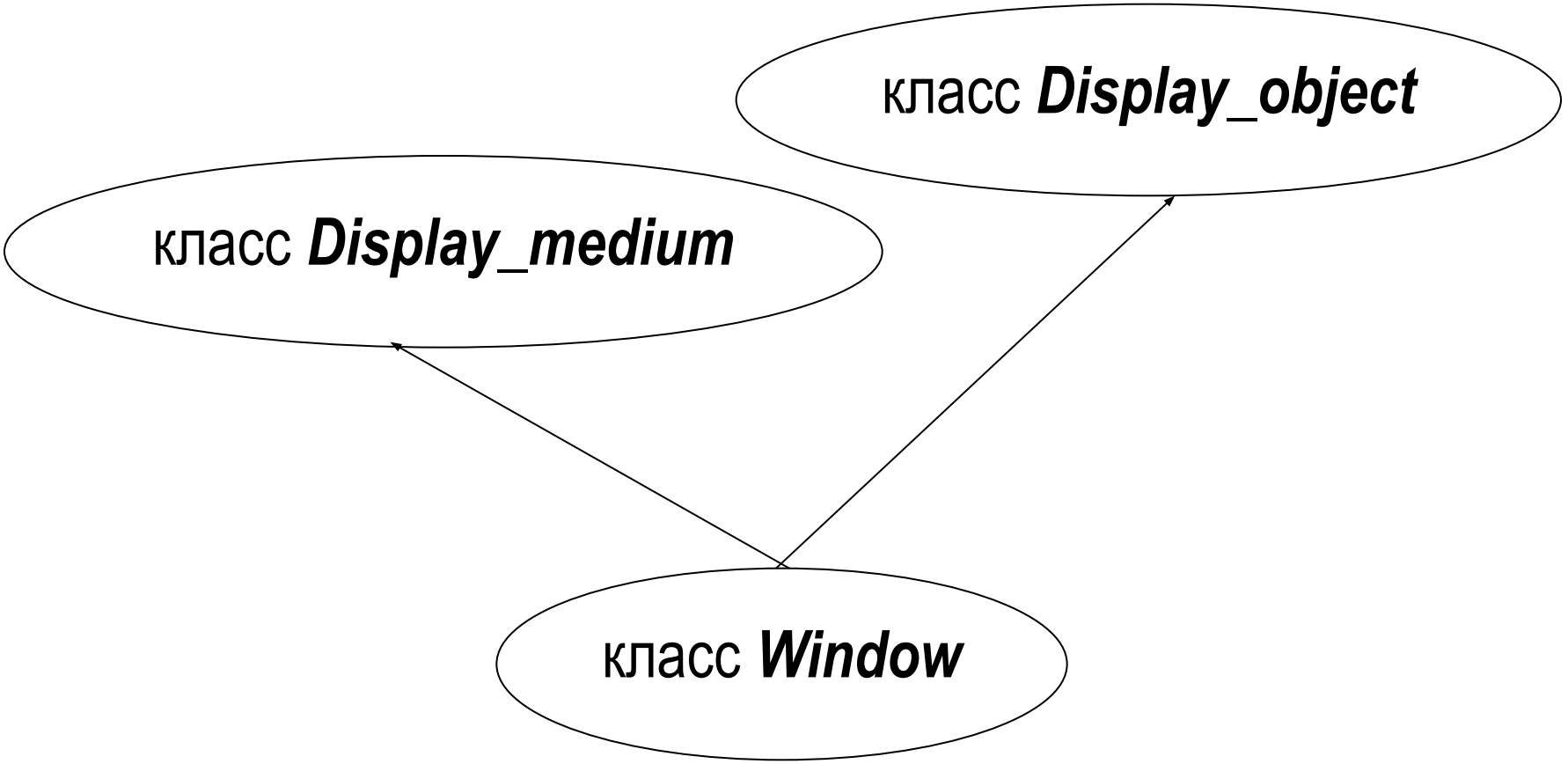
Тема: Множественное наследование

- Постановка задачи
- Синтаксис оформления
- Инвариант абстракции действия
- Возникновение неоднозначностей
- Преодоление неоднозначностей

Постановка проблемы: класс ***Window***

- Бывает, что класс можно считать потомком различных базовых классов
- Окно ***Window*** используется для вывода данных (***Display_medium***)
- Окно ***Window*** само отображается и перемещается на рабочем столе (***Display_object***)

Схема наследования



Класс ***Display_object***

```
class Display_object  {
  public:
     virtual void move( const Point& new_location )=0;
     virtual Point  location( ) const = 0;
      // функция location должна вернуть точку, в которую
       // объект был перемещен в последний раз
};
```

Использование ***Display_object***

```
void bounce( Display_object& d ) {
    Point  at = d.location();
    Point  up = at – Point( 0, 2 );
    Point  down = at + Point( 0, 2 );

    d.move(up);
    d.move(down);
    d.move(up);
    d.move(down);
    d.move(at);
}
```

Класс ***Display_medium***

```
class  Display_medium  {
  public:
    virtual Point size( ) const = 0;
    virtual Point cursor( ) const = 0;
    virtual int move_cursor(const  Point&  p ) = 0;
    virtual Display_char  character( ) const = 0;
    virtual String  line( ) const = 0;
    virtual void add( Display_char  ch ) = 0;
    virtual void add( const  String& s ) = 0;
    virtual void clear( );
  private:
};
```

Класс ***Window***

```
class Window : public Display_medium, Display_object {
  public:
    Window( const Point& upleft, const Point& size,
            const String& title);
    ~Window( );
    void move(const Point& new_upleft);
    Point location() const;
    Point upper_left() const;
    Point lower_right() const;
    Point size() const;
    virtual void change_size(const Point& new_size);
    // если новый размер меньше минимального, то окно
    // уменьшается только до минимального размера
```

Класс ***Window*** (продолжение)

```
    int move_cursor( const Point&  where );
    Point cursor( ) const;
    Display_char character( ) const;
    String line( ) const;
    void add( Display_char c );
    void add( const String& str );
    void clear( );
    void scroll_up( );
    void scroll_down( );
  private :
};
```

Инвариант абстракции действия

- ***location( )*** - чистый виртуальный метод класса ***Display_object***
- Нет *реализации*, но есть *спецификация*, которая задает *инвариант абстракции действия*
- Производные классы обеспечивают реализацию, которая должна соответствовать спецификации
- Если соответствие нельзя обеспечить, то лучше отказаться от наследования

Методы ***move*()** и ***location*()**

```
void Window :: move ( const Point& new_location ) {
    _upper_left = new_location  ;
}

Point Window :: upper_left ( ) const {
    return _upper_left ;
}

Point Window :: location ( ) const {
    return upper_left( ) ;
}
```

Использование объектов ***Window***

```
void  say_hello( Display_medium& m ) {
     m.add( “Hello, world \n” );
}

void main( void ) {
   Window w( Point(2,2), Point(60,10), “test” );
    say_hello( w ) ;
    bounce( w ) ;
}
```

Неоднозначности

- Если класс наследует от различных базовых классов функции с одинаковыми именами и параметрами, то обращения к таким функциям могут быть неоднозначными

Метод ***clear()***

```
class Display_medium {
  public:
     virtual void clear( );
     // ……….
};
```

```
class Display_object {
  public:
     virtual void clear( );
     // ……….
};
```

```
class Window : public Display_medium, Display_object {
  public:
     // нет функции clear ()
};
void main(void) {
   Window w( Point(1,1), Point(10,10), “test” );
   w.clear( );    // ошибка (какая из clear() ? )
}
```

Разрешение неоднозначностей

```
class Window : public Display_medium, Display_object
{
  public:
     // нет функции clear ()
};
void main(void) {
   Window w( Point(1,1), Point(10,10), “test” );
   w.Display_medium::clear( );
   w.Display_object::clear( );
}
```

Предотвращение неоднозначностей

```
class Window : public Display_medium, Display_object
{
  public:
     void clear();
};
void Window::clear() {
   Display_medium::clear( );  Display_object::clear( );
}
void main(void) {
   Window w( Point(1,1), Point(10,10), “test” );
   w.clear( );
}
```

РЕЗЮМЕ

- У класса может быть сразу несколько базовых классов
- Класс должен делать то, что и каждый из базовых классов
- Могут возникнуть неоднозначности
  - разрешаются путем использования операции области видимости
  - предотвращаются подменой в производном классе одноименной функции базовых классов