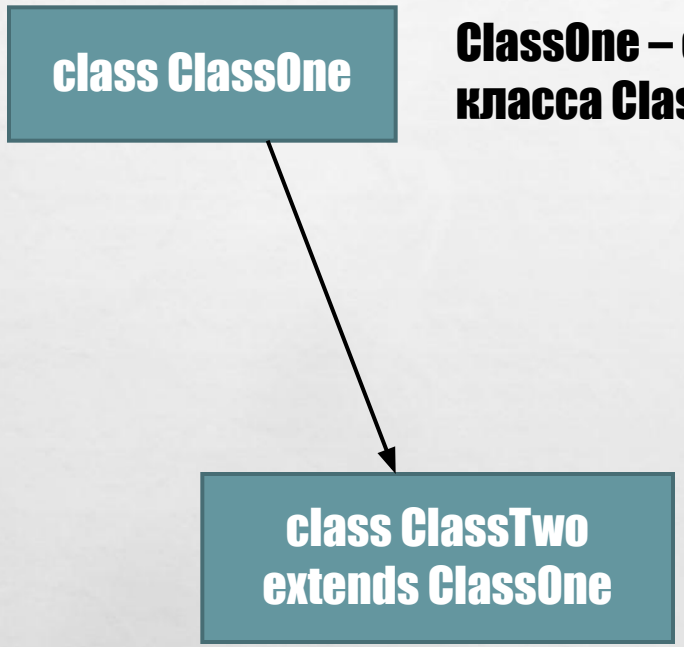
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

НАСЛЕДОВАНИЕ. ЧТО ЭТО И ЗАЧЕМ ЭТО.

НАСЛЕДОВАНИЕ

- НАСЛЕДОВАНИЕ – ЭТО СПОСОБНОСТЬ КЛАССОВ СОБИРАТЬСЯ В ВЕРТИКАЛЬНУЮ ИЕРАРХИЮ (ИЕРАРХИЮ НАСЛЕДОВАНИЯ)
- САМЫЙ ВЕРХНИЙ КЛАСС В ИЕРАРХИИ – САМЫЙ ОБЩИЙ.
- САМЫЙ НИЖНИЙ КЛАСС В ИЕРАРХИИ – САМЫЙ КОНКРЕТНЫЙ.
- КОГДА КАКОЙ-ТО КЛАСС Б ОБЪЯВЛЯЕТСЯ НАСЛЕДНИКОМ ДРУГОГО КЛАССА А – ТО Б СРАЗУ ПОЛУЧАЕТ ВСЕ СВОЙСТВА И МЕТОДЫ (ПЕРЕМЕННЫЕ И ФУНКЦИИ) ОТ А (Б НАСЛЕДУЕТ ОТ А ВСЕ ЕГО СОДЕРЖИМОЕ)

НАСЛЕДОВАНИЕ СХЕМАТИЧНО



ClassOne – супер-класс класса ClassTwo

ClassOne – класс-папашка класса ClassTwo

ClassTwo – подкласс класса ClassOne

ClassTwo – класс-сыночек класса ClassOne

НАСЛЕДОВАНИЕ В КОДЕ

Создадим два класса. ClassOne – папашка, ClassTwo – сыночек.

Main.hx | ClassOne.hx | ClassTwo.hx

```
1   package src.classes;
2   import openfl.display.Sprite;
3   import openfl.Lib;
4
5   class ClassOne extends Sprite
6   {
7       var spr1: Sprite;
8
9       public function new()
10      {
11          super();
12
13          spr1 = new Sprite();
14          spr1.graphics.beginFill(0xff0000);
15          spr1.graphics.drawRect( -20, -20, 40, 40);
16          addChild(spr1);
17      }
18
19  }
```

Main.hx | ClassOne.hx | ClassTwo.hx

```
1   package src.classes;
2   import openfl.display.Sprite;
3   import openfl.Lib;
4   import src.classes.*;
5
6   class ClassTwo extends ClassOne
7   {
8       //var spr1: Sprite; - унаследовано от ClassOne
9       var spr2: Sprite;
10
11      public function new()
12      {
13          super();
14
15          spr2 = new Sprite();
16          spr2.graphics.beginFill(0xffff00);
17          spr2.graphics.drawCircle(0, 0, 10);
18          addChild(spr2);
19      }
20
21  }
```

НАСЛЕДОВАНИЕ В КОДЕ

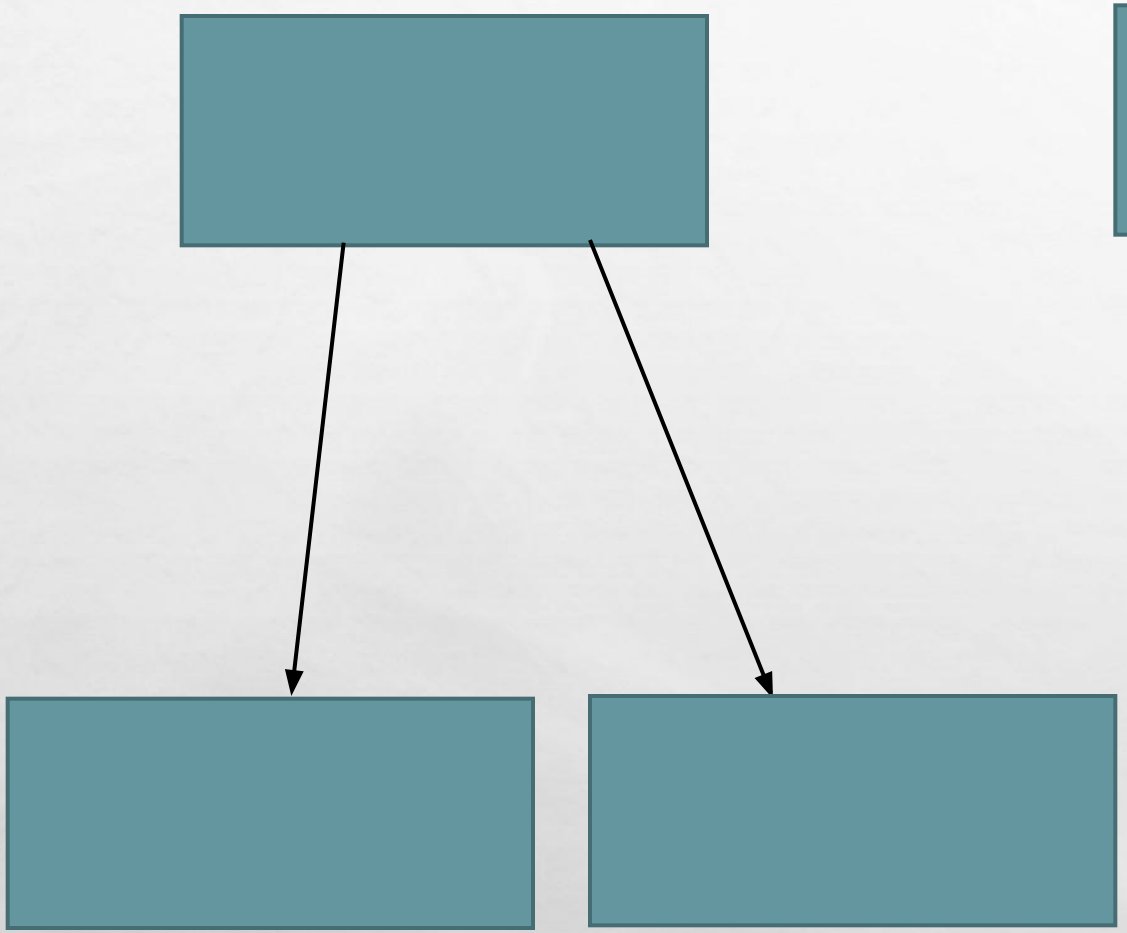
Main.hx | ClassOne.hx | ClassTwo.hx

```
1  package;
2
3  import openfl.display.Sprite;
4  import openfl.Lib;
5  import src.classes.*;
6
7  class Main extends Sprite
8  {
9      var obj1: ClassOne;
10     var obj2: ClassTwo;
11
12     public function new()
13     {
14         super();
15
16         // Assets:
17         // openfl.Assets.getBitmapData("img/assetname.jpg");
18
19         obj1 = new ClassOne();
20         obj1.x = 100;
21         obj1.y = 100;
22         addChild(obj1);
23
24         obj2 = new ClassTwo();
25         obj2.x = 200;
26         obj2.y = 100;
27         addChild(obj2);
28     }
29
30 }
31
```

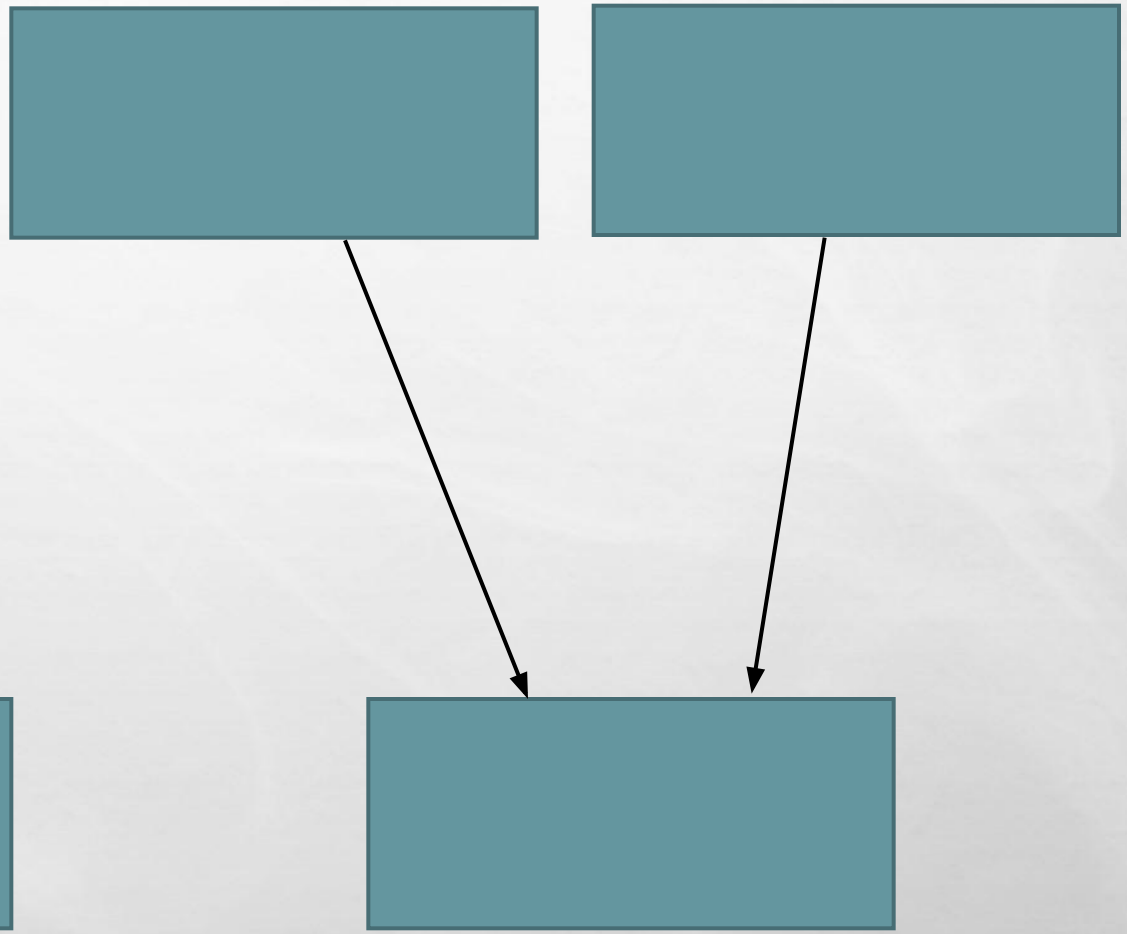
Создадим по одному объекту каждого из классов и посмотрим, что получится.

НАСЛЕДОВАНИЕ В КОДЕ

Картинка №1



Картинка №2



- КАРТИНКА №1 – У ОДНОГО ПАПАШКИ НЕСКОЛЬКО СЫНОЧКОВ. ЭТО НОРМАЛЬНАЯ СИТУАЦИЯ, ОНА ВОЗМОЖНА НА НАХЕ И ДРУГИХ ЯЗЫКАХ ВЫСОКО УРОВНЯ. КЛАССЫ СЫНОЧКИ В ТАКОЙ СИТУАЦИИ НИКАК НЕ ПЕРЕСЕКАЮТСЯ. ОНА ДАЖЕ НЕ ЗНАЮТ О СУЩЕСТВОВАНИИ ДРУГ ДРУГА.

- КАРТИНКА №2 – У ОДНОГО СЫНОЧКА НЕСКОЛЬКО ПАПАШЕК. ЭТА СИТУАЦИЯ НОСИТ НАЗВАНИЕ МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ. ТАКОЕ ВОЗМОЖНО НА C++, НО НЕВОЗМОЖНО НА JAVA И НАХЕ. ДЛЯ МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ НА JAVA И НАХЕ ИСПОЛЬЗУЮТ ИНТЕРФЕЙСЫ.

РЕАЛИЗАЦИЯ КАРТИНКИ №1.

СОЗДАДИМ CLASSTHREE, КОТОРЫЙ ТОЖЕ БУДЕТ НАСЛЕДНИКОМ CLASSONE, НО СО СВОИМ СОБСТВЕННЫМ РАСШИРЕНИЕМ.

Main.hx | ClassOne.hx | ClassTwo.hx | **ClassThree.hx**

```
1   package src.classes;
2   import openfl.display.Sprite;
3   import openfl.Lib;
4   import src.classes.*;
5
6   class ClassThree extends ClassOne
7   {
8       var spr3: Sprite;
9
10      public function new()
11      {
12          super();
13
14          spr3 = new Sprite();
15          spr3.graphics.beginFill(0x00ff00);
16          spr3.graphics.drawCircle(0, 0, 30);
17          addChild(spr3);
18      }
19
20  }
```

РЕАЛИЗАЦИЯ КАРТИНКИ №1.

СОЗДАДИМ ОБЪЕКТ КЛАССА CLASSTHREE И ПОСМОТРИМ РЕЗУЛЬТАТ

Main.hx* | ClassOne.hx | ClassTwo.hx | ClassThree.hx

```
1  package;
2
3  import openfl.display.Sprite;
4  import openfl.Lib;
5  import src.classes.*;
6
7  class Main extends Sprite
8  {
9      var obj1: ClassOne;
10     var obj2: ClassTwo;
11     var obj3: ClassThree;
12
13     public function new()
14     {
15         super();
16
17         obj1 = new ClassOne();
18         obj1.x = 100;
19         obj1.y = 100;
20         addChild(obj1);
21
22         obj2 = new ClassTwo();
23         obj2.x = 200;
24         obj2.y = 100;
25         addChild(obj2);
26
27         obj3 = new ClassThree();
28         obj3.x = 300;
29         obj3.y = 100;
30         addChild(obj3);
31     }
32
33 }
34
```

РЕАЛИЗАЦИЯ КАРТИНКИ №1.

ПРО МЕТОДЫ.

- КАЖДЫЙ КЛАСС МОЖЕТ ИМЕТЬ СВОЙ НАБОР МЕТОДОВ (ФУНКЦИЙ).

- КЛАСС-СЫНОЧЕК НАСЛЕДУЕТ ВСЕ МЕТОДЫ КЛАССА-ПАПАШКИ И МОЖЕТ ИХ СВОБОДНО ИСПОЛЬЗОВАТЬ.

Main.hx | **ClassOne.hx** | ClassTwo.hx | ClassThree.hx

```haxe
package src.classes;
import openfl.display.Sprite;
import openfl.Lib;
import openfl.text.TextField;
import openfl.text.TextFormat;

class ClassOne extends Sprite
{
    var spr1: Sprite;
    var field: TextField;

    public function new()
    {
        super();

        spr1 = new Sprite();
        spr1.graphics.beginFill(0xff0000);
        spr1.graphics.drawRect( -40, -40, 80, 80);
        addChild(spr1);

        field = null;
    }

    public function MakeField()
    {
        var format = new TextFormat();
        format.font = "Arial";
        format.color = 0x000000;
        format.size = 20;

        field = new TextField();
        field.defaultTextFormat = format;
        field.x = -40;
        addChild(field);
    }

    public function SaySomething()
    {
        field.text = "One";
    }
}
```

Main.hx | ClassOne.hx | ClassTwo.hx | ClassThree.hx

```haxe
package;

import openfl.display.Sprite;
import openfl.Lib;
import src.classes.*;

class Main extends Sprite
{
    var obj1: ClassOne;
    var obj2: ClassTwo;
    var obj3: ClassThree;

    public function new()
    {
        super();

        obj1 = new ClassOne();
        obj1.x = 100;
        obj1.y = 100;
        addChild(obj1);
        obj1.MakeField();
        obj1.SaySomething();

        obj2 = new ClassTwo();
        obj2.x = 200;
        obj2.y = 100;
        addChild(obj2);
        obj2.MakeField();
        obj2.SaySomething();

        obj3 = new ClassThree();
        obj3.x = 300;
        obj3.y = 100;
        addChild(obj3);
        obj3.MakeField();
        obj3.SaySomething();
    }

}
```


Adobe Flash Player 20
Файл Просмотр Управление Справка
One
One
One

ПЕРЕГРУЗКА МЕТОДОВ

- КЛАСС-СЫНОЧЕК МОЖЕТ ПЕРЕГРУЖАТЬ МЕТОДЫ КЛАССА-ПАПАШКИ (ПЕРЕОПРЕДЕЛЯТЬ ИХ), ЗАМЕНЯЯ ПОВЕДЕНИЕ ПАПАШКИ НА СВОЕ СОБСТВЕННОЕ.

- КЛАСС-СЫНОЧЕК ПОМИМО ПЕРЕГРУЗКИ МОЖЕТ СОДЕРЖАТЬ СВОИ СОБСТВЕННЫЕ МЕТОДЫ, О КОТОРЫХ ПАПАШКА НЕ БУДЕТ ДАЖЕ ПОДОЗРЕВАТЬ. ИСПОЛЬЗОВАТЬ ЭТИ МЕТОДЫ ПАПАШКА ТОЖЕ НЕ СМОЖЕТ.

Main.hx | ClassOne.hx | **ClassTwo.hx** | ClassThree.hx

```haxe
package src.classes;
import openfl.display.Sprite;
import openfl.Lib;
import src.classes.*;
import openfl.text.TextField;
import openfl.text.TextFormat;

class ClassTwo extends ClassOne
{
    //var spr1: Sprite; - унаследовано от ClassOne
    var spr2: Sprite;

    public function new()
    {
        super();

        spr2 = new Sprite();
        spr2.graphics.beginFill(0xffff00);
        spr2.graphics.drawCircle(0, 0, 30);
        addChild(spr2);
    }

    override public function SaySomething()
    {
        // super.SaySomething();

        field.text = "Two";
    }

}
```

Main.hx | ClassOne.hx | ClassTwo.hx | **ClassThree.hx**

```haxe
package src.classes;
import openfl.display.Sprite;
import openfl.Lib;
import src.classes.*;
import openfl.text.TextField;
import openfl.text.TextFormat;

class ClassThree extends ClassOne
{
    var spr3: Sprite;

    public function new()
    {
        super();

        spr3 = new Sprite();
        spr3.graphics.beginFill(0x00ff00);
        spr3.graphics.drawCircle(0, 0, 30);
        addChild(spr3);
    }

    override public function SaySomething()
    {
        // super.SaySomething();

        field.text = "Three";
    }

}
```


Adobe Flash Player 20
Файл Просмотр Управление Справка
One
Two
Three

ЕЩЕ ПРО ПЕРЕГРУЗКУ

- ЗАГОЛОВКИ ДОЛЖНЫ СОВПАДАТЬ! Т.Е. ЕСЛИ У ВАС В ПАПАШКЕ ЕСТЬ

  PUBLIC FUNCTION F()

- И В СЫНОЧКЕ ХОТИТЕ ПЕРЕГРУЗИТЬ ЭТОТ МЕТОД, ЗАГОЛОВОК ДОЛЖЕН

  ВЫГЛЯДЕТЬ ВОТ ТАК

  OVERRIDE PUBLIC FUNCTION F()

- ЕСЛИ БУДЕТ

  OVERRIDE PUBLIC FUNCTION F(*КАКИЕ-ТО АРГУМЕНТЫ*)

- Я НЕ ЗНАЮ, ЧТО ПРОИЗОЙДЕТ, НО ЭТО СТОПУДОВА БУДЕТ НЕ ПЕРЕГРУЗКА!

ЗАДАНИЕ

- СДЕЛАТЬ ПРИМЕР, ОПИСАННЫЙ В ПРЕЗЕНТАШКЕ. ПРОВЕРИТЬ, ЧТО ВСЕ РАБОТАЕТ.

- ДЛЯ ХАРДКОРЩИКОВ: В КЛАССАХ CLASSTWO И CLASSTHREE ПЕРЕГРУЗИТЕ МЕТОД MAKEFIELD(). СДЕЛАЙТЕ ТАК, ЧТОБЫ ЭТОТ МЕТОД СОЗДАВАЛ ТЕКСТОВОЕ ПОЛЕ, В КОТОРОМ ТЕКСТ ПЕЧАТАЕТСЯ НЕ ЧЕРНЫМ ЦВЕТОМ (МОЖНО ПОМЕНЯТЬ ЕЩЕ ШРИФТ И РАЗМЕР).