Объектно-ориентированное программирование (ООП)

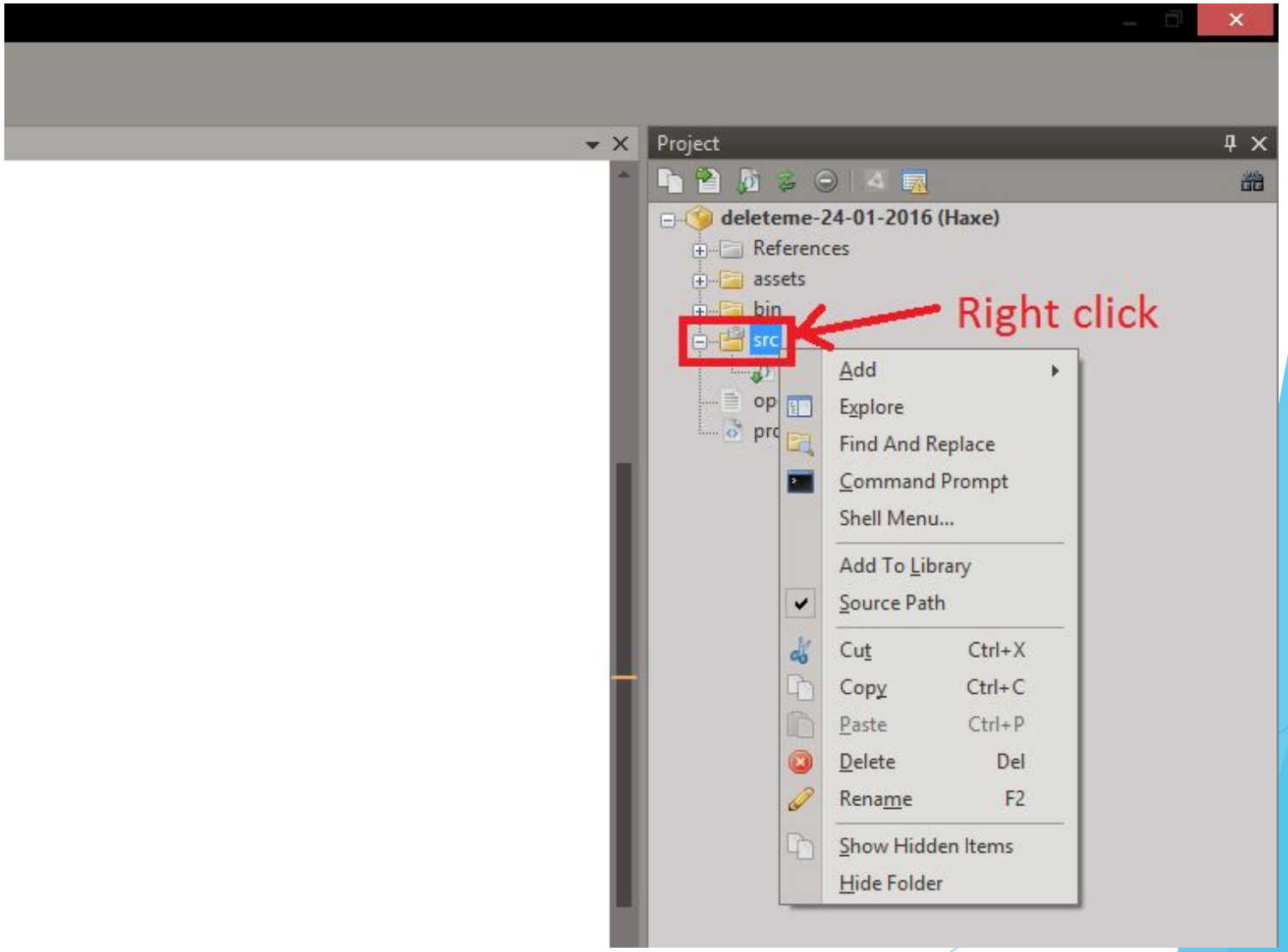
Класс. Объект класса. Конструктор класса. Поля. Методы.

Класс

- Набор полей (переменных) и методов (функций), которые так или иначе работают с этими полями.
- Созданный руками программиста тип данных
- Примеры типичных классов: Sprite, Bitmap, Sound (наверное), SoundChannel, Point.

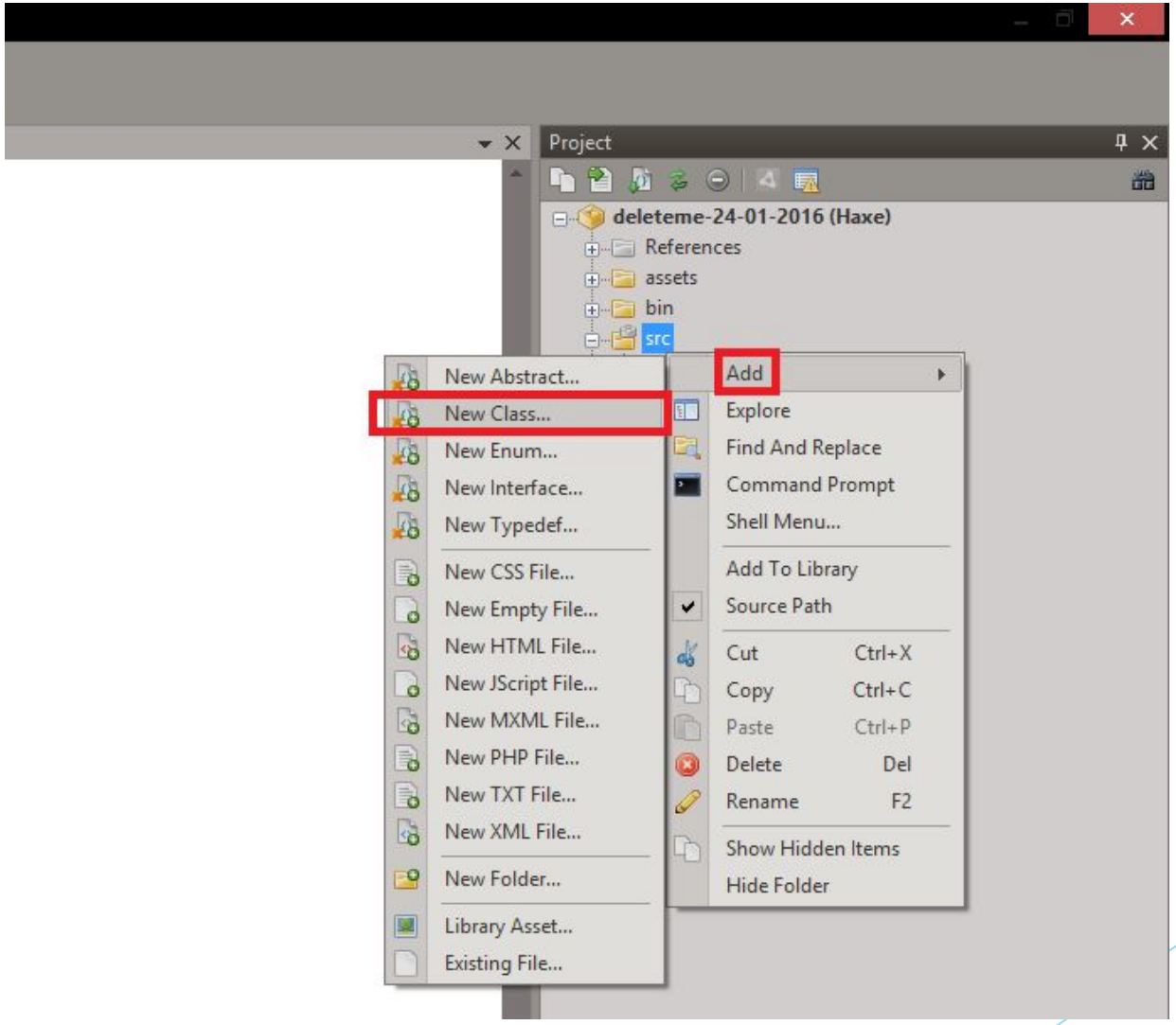
Создаем свой класс

- Шаг 1



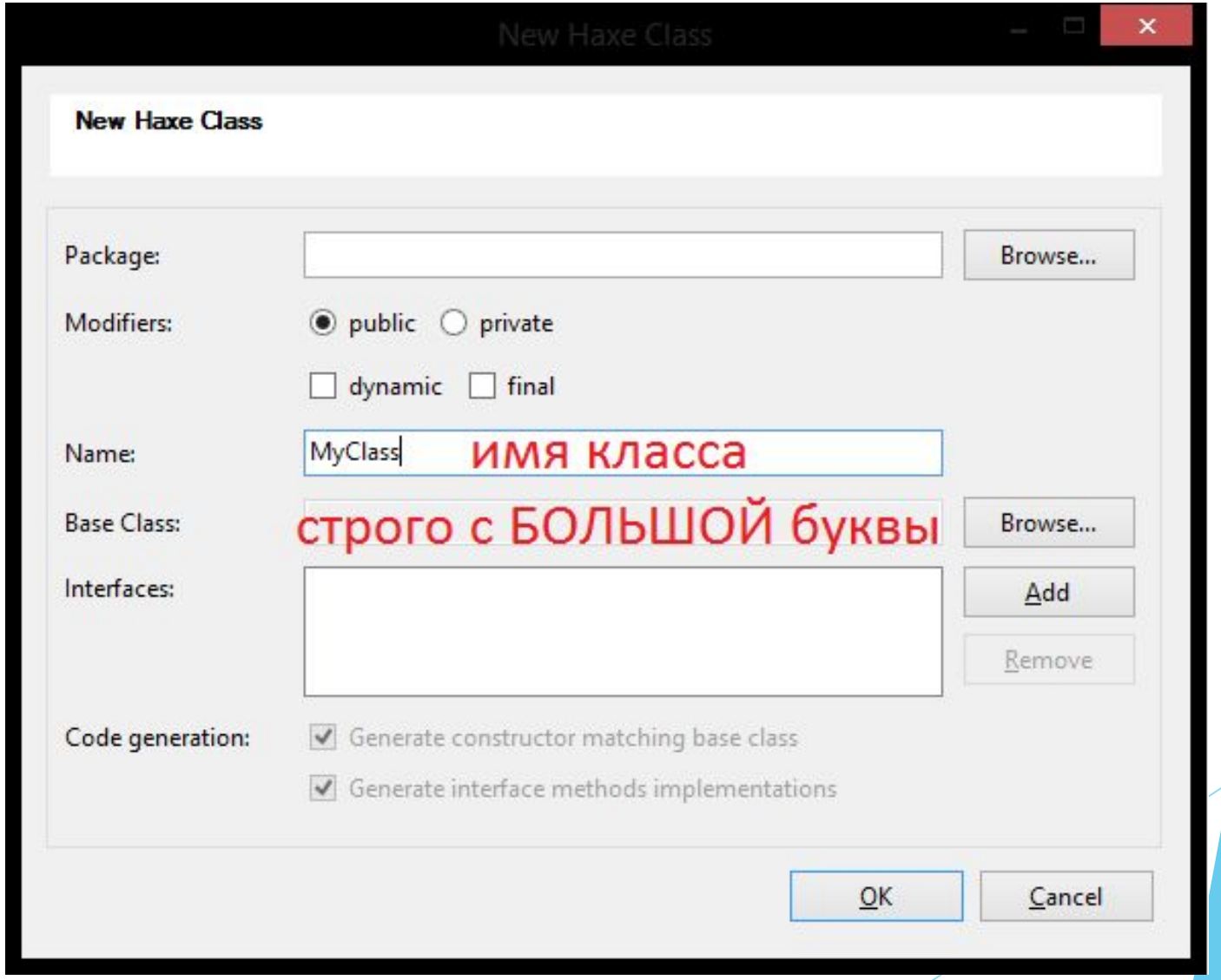
Создаем свой класс

- Шаг 2



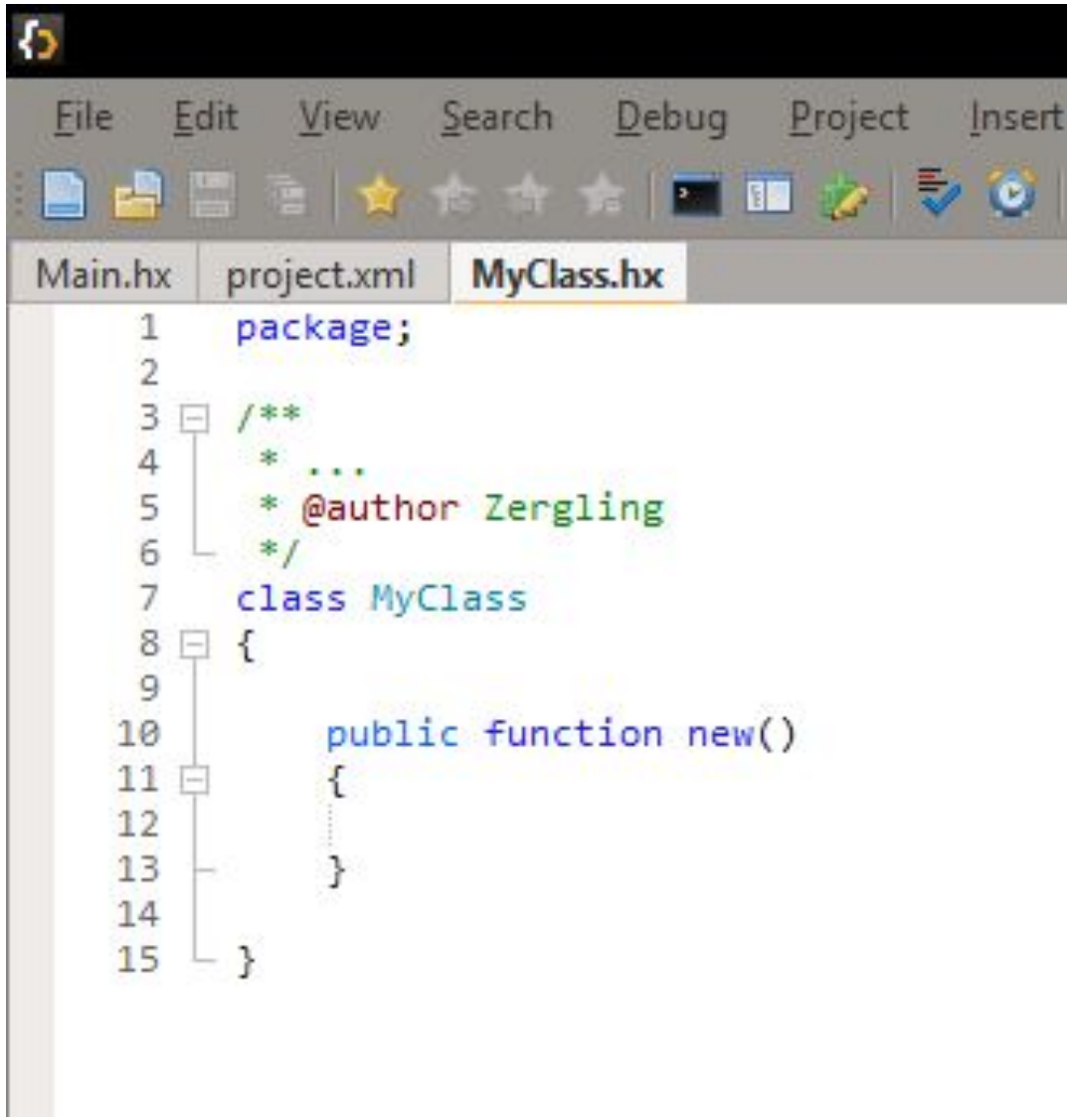
Создаем свой класс

- Шаг 3



Создаем свой класс

- Шаг 4



Объект класса

- Переменная, типом данных которой является **КЛАСС**.
- Когда мы писали var spr: Sprite – мы создавали объект класса Sprite
- Каждый объект класса имеет свойства (поля) и поведение (методы) своего класса, но между собой объекты индивидуальны (в каждый момент времени мы можем воздействовать только на ОДИН объект класса)

```
24  class Main extends Sprite
25  {
26
33
34      var abc: MyClass;
35
36      public function new()
37      {
38          super();
39
40          |
41
61      }
62
63
75
76  }
77
```

Объект класса MyClass

Некоторая аналогия для классов и объектов в реальной жизни.

- Класс – форма для печенья
- Объект класса – конкретная печенька, сделанная по этой форме

Как создаются объекты? (Как происходит выпекание печеньки)

- Процессом создания объекта (начальной инициализации его полей) занимается конструктор

- **Конструктор** – функция, которая начинает свое выполнение при инициализации объекта класса.

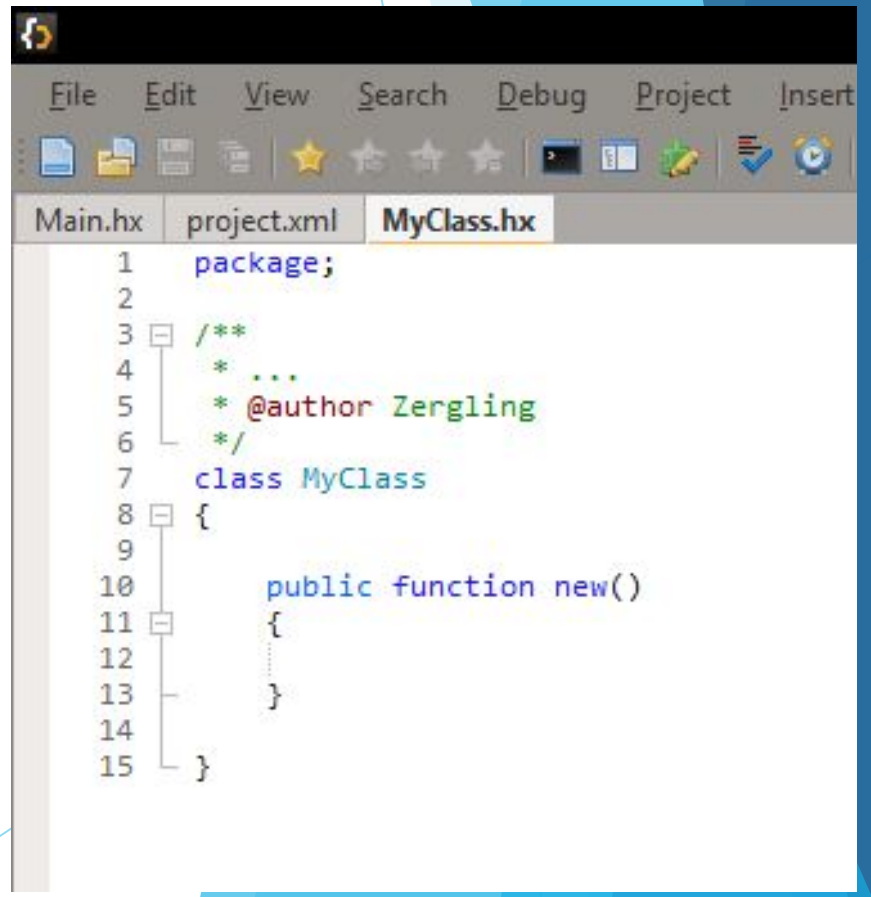
- **ОБЯЗАТЕЛЬНО ДОЛЖЕН БЫТЬ!!!**

- **Конструктор в HaXe** – функция **new** класса. Он начинает свою работу во время выполнения строки *abc = new MyClass();*

  Мы уже наблюдали такую строку, когда писали

  var spr: Sprite;

  spr = new Sprite();



Конструктор класса в HaXe. (Выпекатор печеньки)

- public function new() в каждом классе.

- **ОБЯЗАТЕЛЬНО ДОЛЖЕН БЫТЬ!!!**

- Производит начальную настройку объекта класса.

- Может иметь параметры - значения, передающиеся извне и задающие начальное состояния каких-либо полей. Этот механизм обычно используется тогда, когда один конструктор может создавать почти одинаковые, но немного отличающиеся между собой объекты.

Некоторые фичи классов HaXe.

- Внутри класса можно привязать какие-нибудь EventListener’ы. Этим листнером будут обладать все объекты данного класса.
- При создании листнера в классе надо будет описать его обработчик. Каждый объект будет вызывать этот обработчик.
- Применительно к игрострою: одним классом HaXe можно описать целый экран игры и все его поведение. Довольно удобно.

Небольшой пример

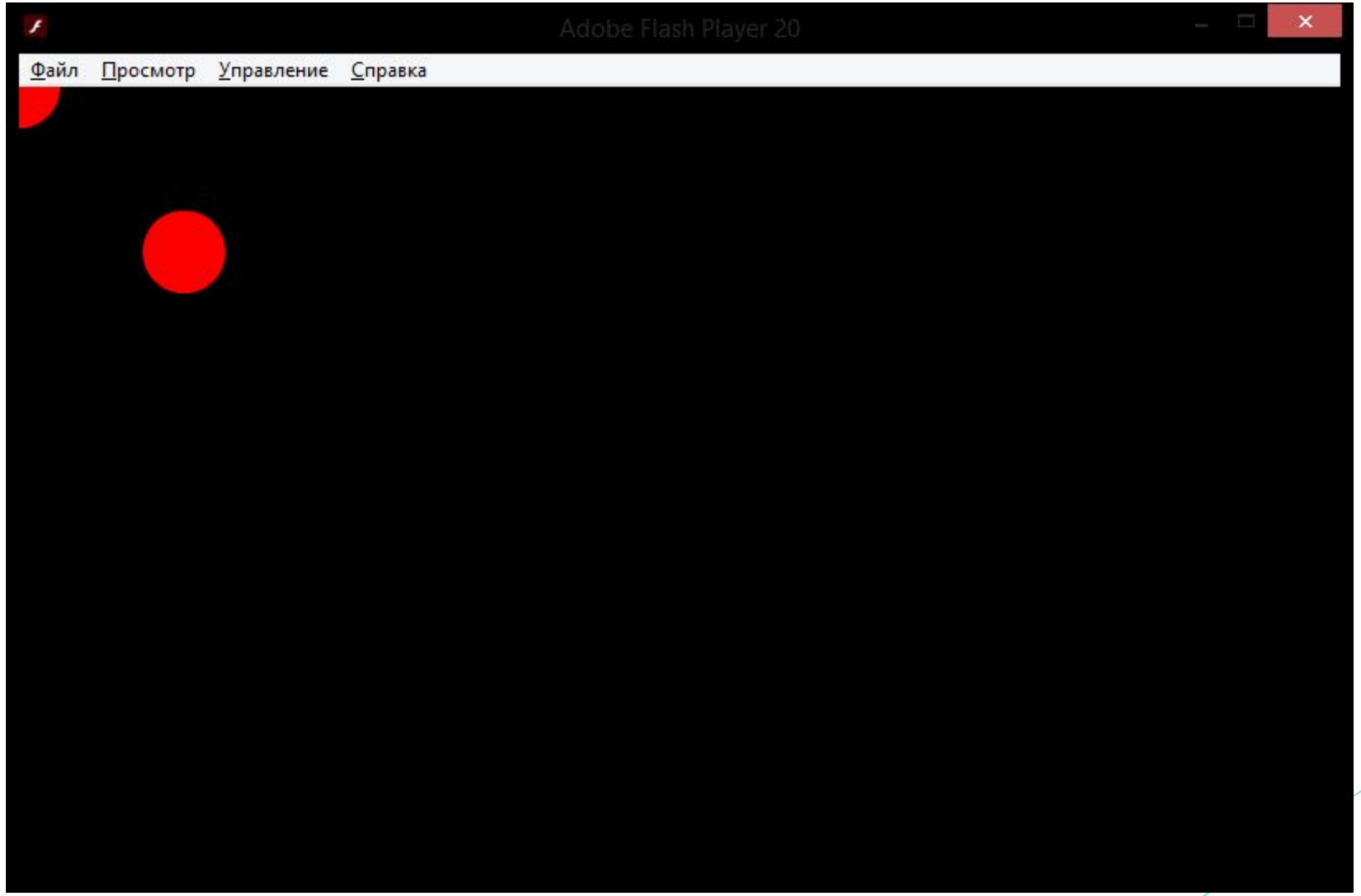
Main.hx | RedCircle.hx

```haxe
package;

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

/**
 * ...
 * @author Zergling
 */
class Main extends Sprite
{
    var rc: RedCircle;
    var redcirlce: RedCircle;

    public function new()
    {
        super();

        // Assets:
        // openfl.Assets.getBitmapData
        rc = new RedCircle();
        addChild(rc);

        redcirlce = new RedCircle();
        redcirlce.x = 100;
        redcirlce.y = 100;
        addChild(redcirlce);
    }

}
```

Main.hx | RedCircle.hx

```haxe
package src;
import openfl.display.Sprite;

/**
 * ...
 * @author Zergling
 */
class RedCircle extends Sprite
{
    var circle: Sprite;

    public function new()
    {
        super();

        circle = new Sprite();
        circle.graphics.beginFill(0xff0000);
        circle.graphics.drawCircle(0, 0, 25);
        addChild(circle);
    }

}
```

Небольшой пример



Еще пример (с листнером)

Main.hx

```haxe
package;

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

class Main extends Sprite
{
    var rc: RedCircle;
    var redcirlce: RedCircle;

    public function new()
    {
        super();

        // Assets:
        // openfl.Assets.getBitmapData
        /*
        rc = new RedCircle();
        addChild(rc);
        */

        redcirlce = new RedCircle();
        redcirlce.x = 100;
        redcirlce.y = 100;
        addChild(redcirlce);
    }

}
```

RedCircle.hx

```haxe
package src;

import openfl.display.Sprite;
import openfl.events.Event;

class RedCircle extends Sprite
{
    var circle: Sprite;
    var frames: Int;

    public function new()
    {
        super();

        circle = new Sprite();
        circle.graphics.beginFill(0xff0000);
        circle.graphics.drawCircle(0, 0, 25);
        addChild(circle);

        frames = 0;

        addEventListener(Event.ENTER_FRAME, Circle_Onframe);
    }

    public function Circle_Onframe(e: Event)
    {
        frames++;

        x = x + Math.cos(frames / 100);
        y = y + Math.sin(frames / 100);
    }

}
```

Важное про классы

- Каждый класс должен уметь убирать за собой мусор. Почти все, что используется в классе, убьется автоматически.
- Листнеры – злые ребята. Их надо отдельно вырубать при удалении объекта класса.
- Для этого надо будет написать отдельную функцию, которая сделает removeEventListener для всех листнеров, которые вешались в ДАННОМ КЛАССЕ.

Области видимости.

- В HaXe есть две области видимости: public и private (не-public), эти модификаторы приписываются к переменным и функциям
- Public говорит о том, что к данной штуке (переменной или функции) можно обратиться извне.
- Private (не-public) говорит о том, что фигушки нам, а не обращение извне.

Main.hx | RedCircle.hx

```haxe
package src;

import openfl.display.Sprite;
import openfl.events.Event;

class RedCircle extends Sprite
{
    var circle: Sprite;
    var frames: Int;

    public var publicInt: Int;
    var nonpublicInt: Int;

    public function new()
    {
        super();

        circle = new Sprite();
        circle.graphics.beginFill(0xff0000);
        circle.graphics.drawCircle(0, 0, 25);
        addChild(circle);

        frames = 0;

        publicInt = 0;
        nonpublicInt = 0;

        addEventListener(Event.ENTER_FRAME, Circle_Onframe);
    }

    public function Circle_Onframe(e: Event)
    {
        frames++;

        x = x + Math.cos(frames / 100);
        y = y + Math.sin(frames / 100);
    }

}
```

Main.hx* | RedCircle.hx

```haxe
package;

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

class Main extends Sprite
{
    var rc: RedCircle;
    var redcirlce: RedCircle;

    public function new()
    {
        super();

        // Assets:
        // openfl.Assets.getBitmapData
        /*
        rc = new RedCircle();
        addChild(rc);
        */

        redcirlce = new RedCircle();
        redcirlce.x = 100;
        redcirlce.y = 100;
        addChild(redcirlce);

        redcirlce.publicInt = 100;
        redcirlce.nonpublicInt = 100;
    }

}
```

Все нормально
ОШИБКА!
ПРОГРАММИСТ - РУКОЖОП

Задание

- Запилить какой-нибудь класс, который будет выбрасывать на сцену что-нибудь визуальное (красный круг, зеленый квадрат или картинку). Создать объект класса. **Убедиться, что все работает.**

- По желанию: добавить туда какой-нибудь EventListener, который будет что-нибудь делать: проигрывать звук, запускать движение объекта, какую-нибудь отсебятину. **Убедиться, что все работает.**

- **Примечание: можно и даже нужно смотреть примеры.**