

# Объектно-ориентированное программирование (ООП)

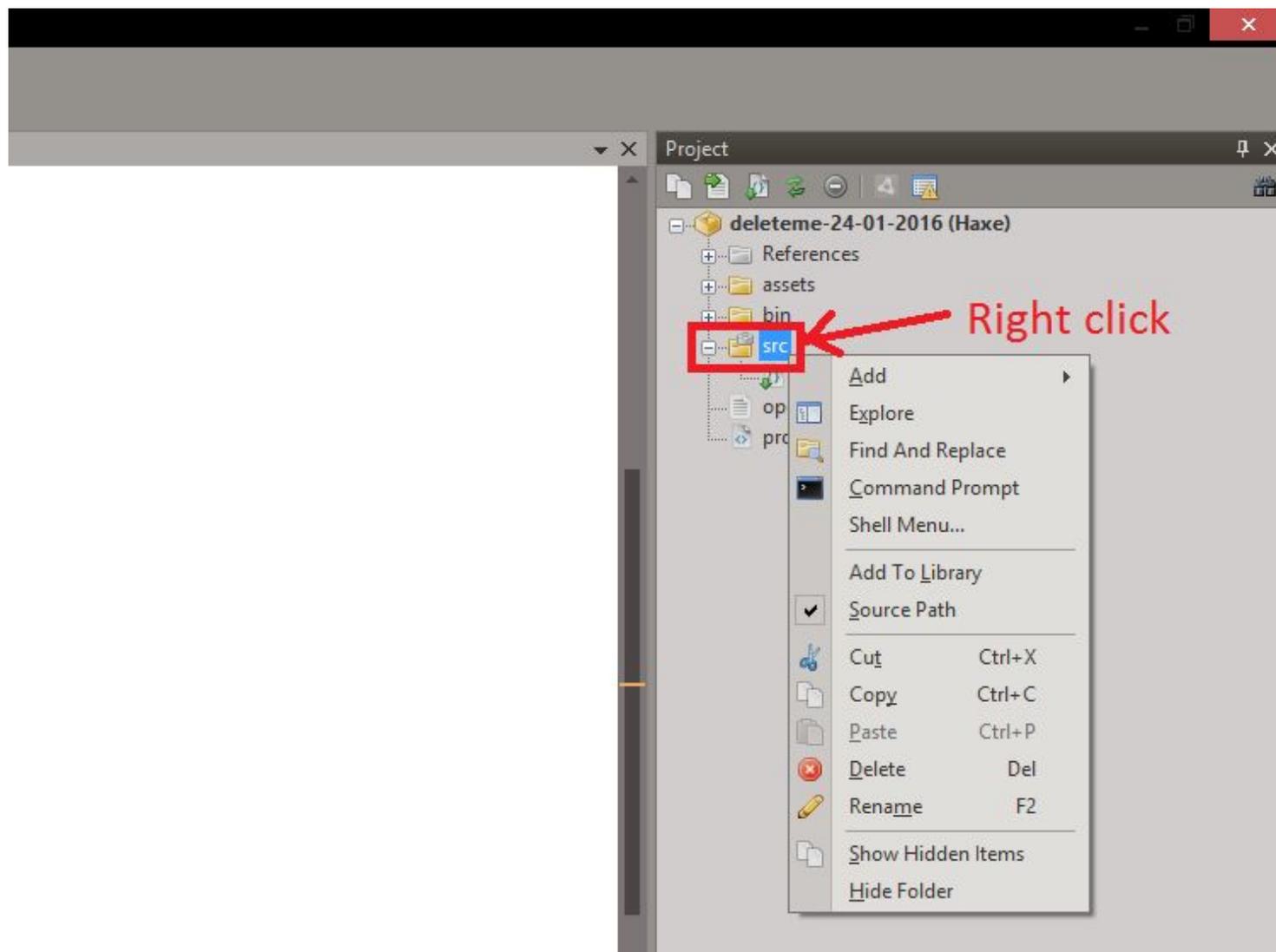
Класс. Объект класса. Конструктор класса. Поля. Методы.

# 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.

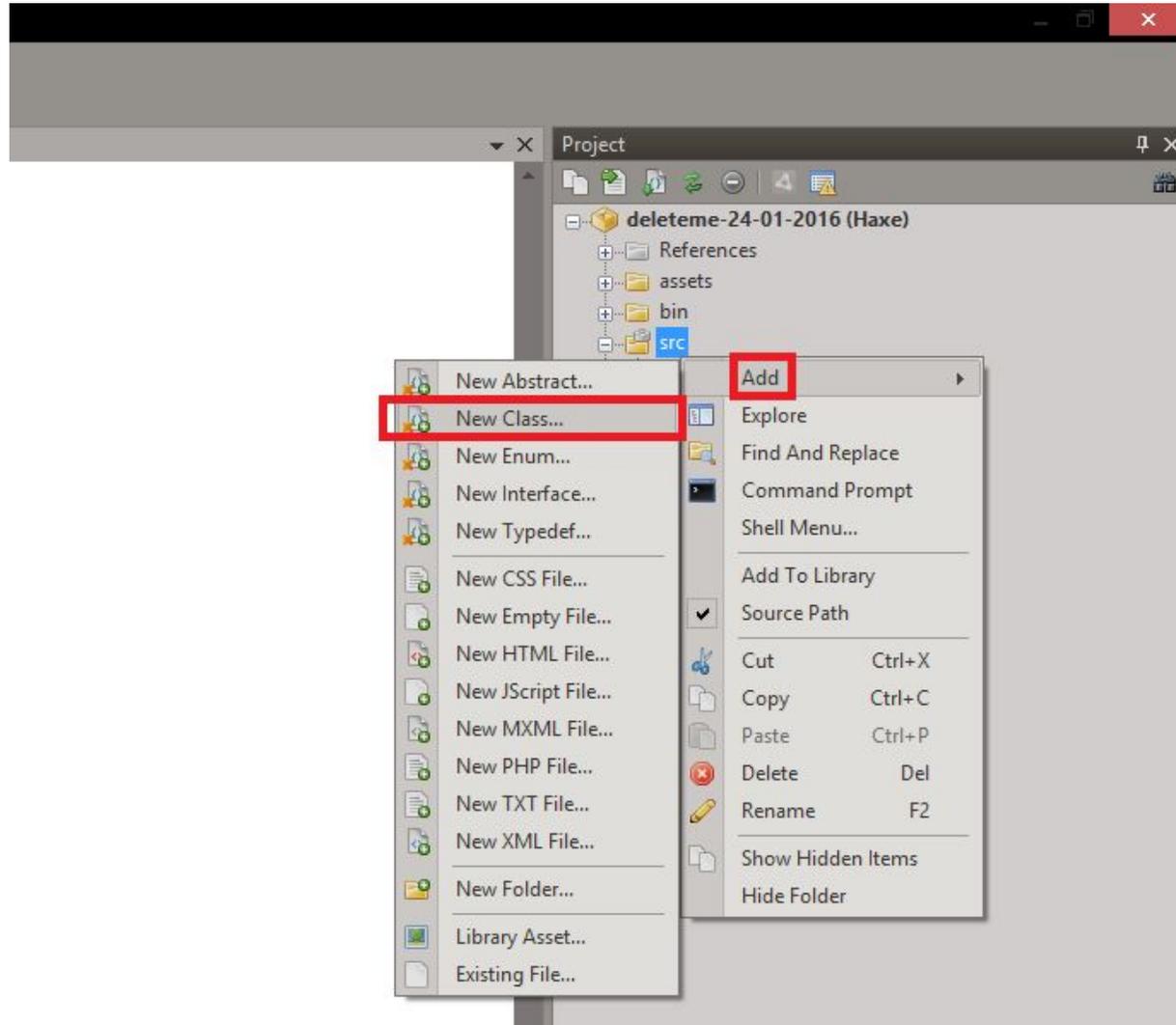
# Создаем свой класс

## ▶ Шаг 1



# Создаем свой класс

## ▶ Шаг 2



# Создаем свой класс

## ▶ Шаг 3

**New Haxe Class**

Package:  Browse...

Modifiers:  public  private  
 dynamic  final

Name:  **ИМЯ КЛАССА**

Base Class:  **строго с БОЛЬШОЙ буквы** Browse...

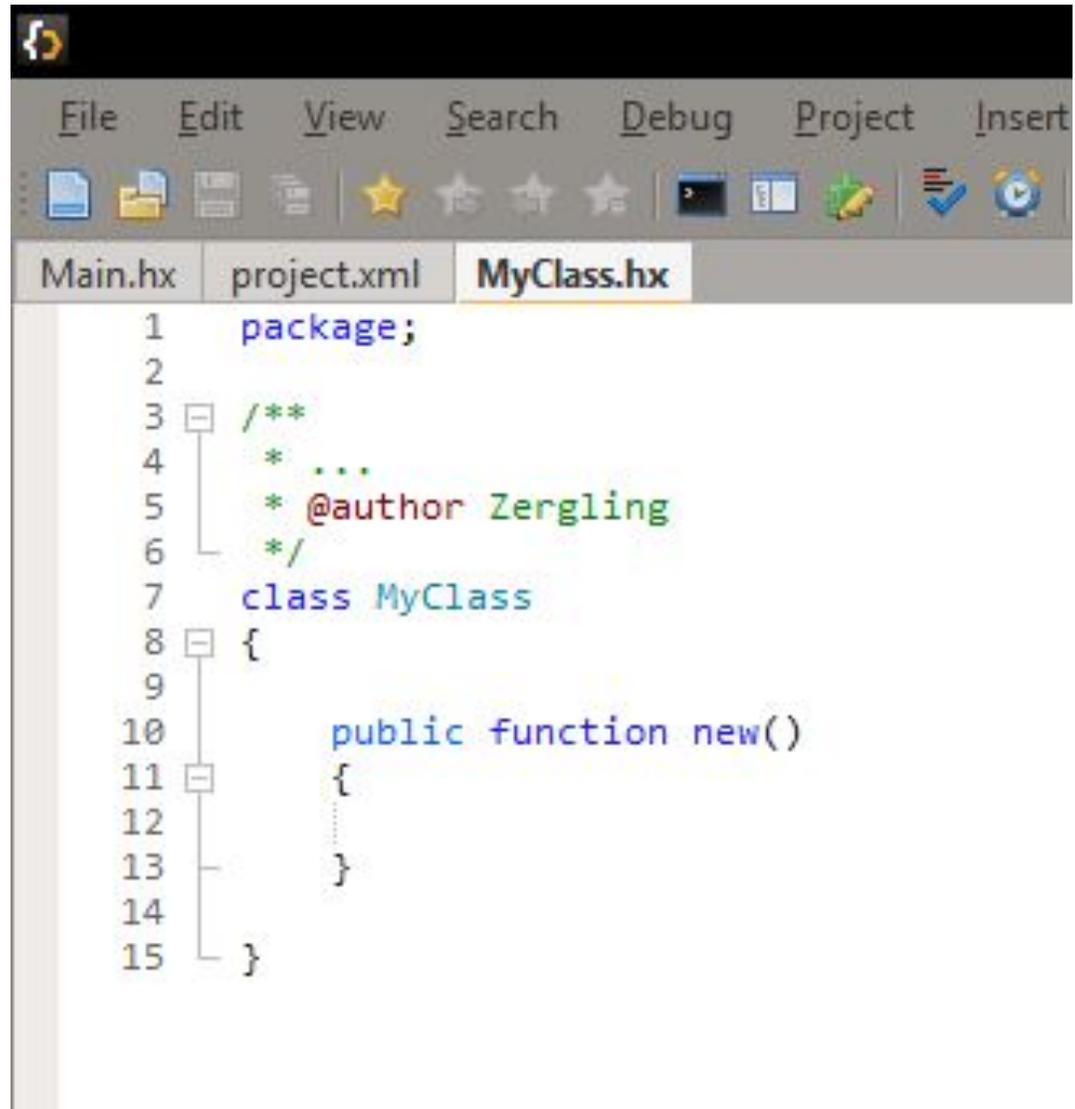
Interfaces:  Add Remove

Code generation:  Generate constructor matching base class  
 Generate interface methods implementations

OK Cancel

# Создаем свой класс

## ▶ Шаг 4



```
1 package;
2
3 /**
4  * ...
5  * @author Zergling
6  */
7 class MyClass
8 {
9
10     public function new()
11     {
12         ...
13     }
14
15 }
```

# 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
35
36
37 -
38     super();
39
40     |
41 +
61 }
62
63 +
75
76 }
77
```

`var abc: MyClass;`

Объект класса `MyClass`

# Некоторая аналогия для классов и объектов в реальной жизни.

- ▶ Класс - форма для печенья
- ▶ Объект класса - конкретная печенье, 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.

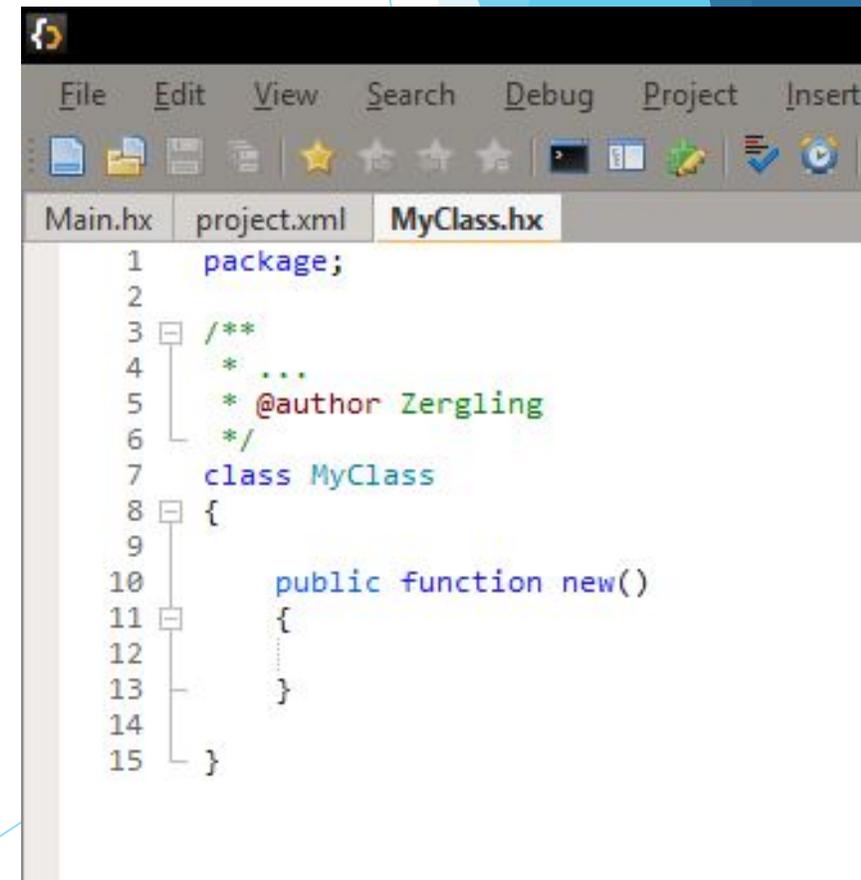
### ▶ **ОБЯЗАТЕЛЬНО ДОЛЖЕН БЫТЬ!!!**

- ▶ 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();
```



```
1 package;
2
3 /**
4  * ...
5  * @author Zergling
6  */
7 class MyClass
8 {
9
10     public function new()
11     {
12         ...
13     }
14
15 }
```

# Конструктор класса в NaHe. (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 фишки классов NaHe.

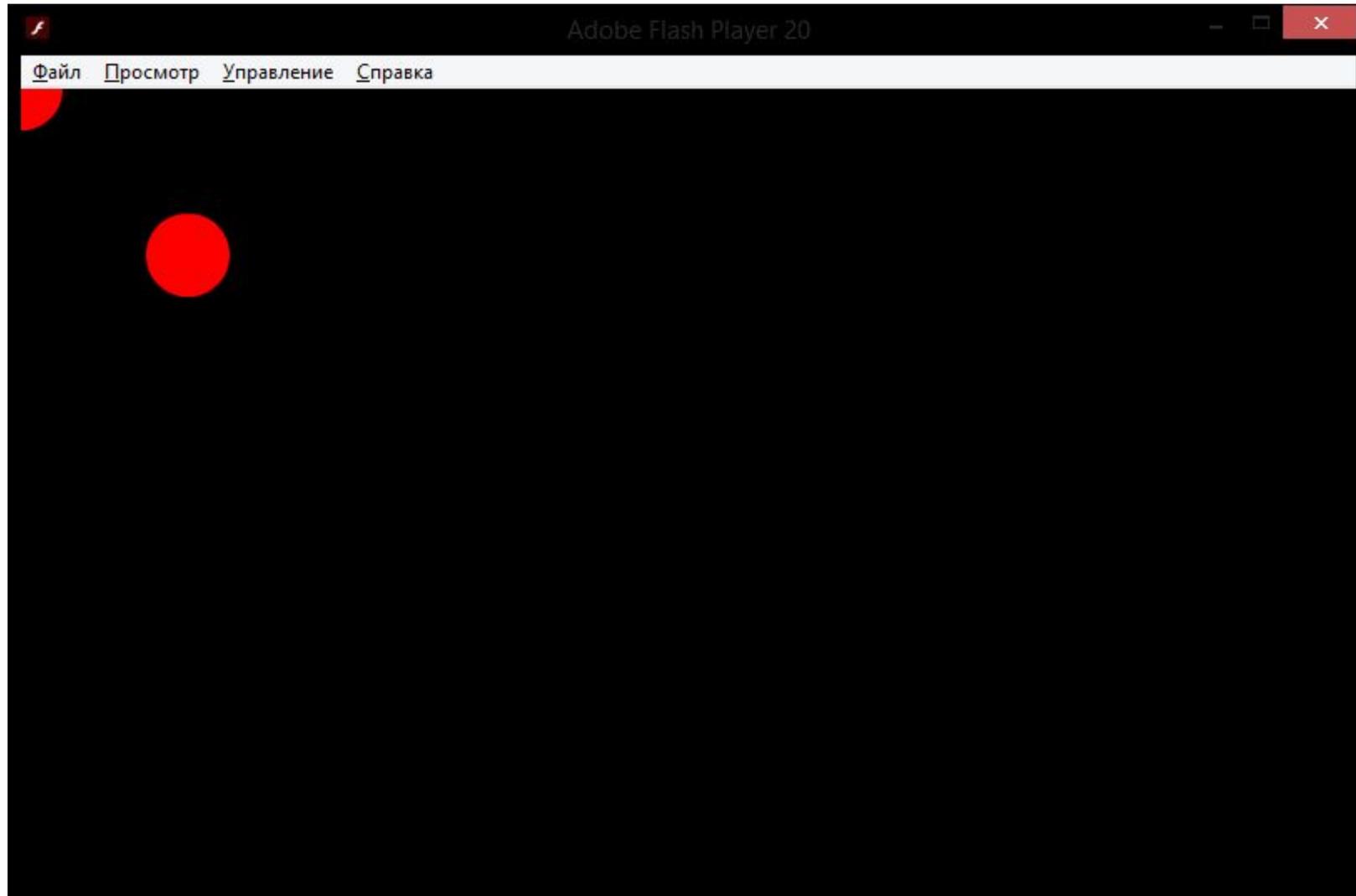
- ▶ Внутри класса можно привязать какие-нибудь EventListener'ы. Этим листнером будут обладать все объекты данного класса.
- ▶ При создании листнера в классе надо будет описать его обработчик. Каждый объект будет вызывать этот обработчик.
- ▶ Применительно к игрострою: одним классом NaHe можно описать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

# Небольшой пример

```
Main.hx  RedCircle.hx
1  package;
2
3  import openfl.display.Sprite;
4  import openfl.Lib;
5  import src.*;
6
7  /**
8   * ...
9   * @author Zergling
10  */
11  class Main extends Sprite
12  {
13      var rc: RedCircle;
14      var redcir1ce: RedCircle;
15
16      public function new()
17      {
18          super();
19
20          // Assets:
21          // openfl.Assets.getBitmapData
22          rc = new RedCircle();
23          addChild(rc);
24
25          redcir1ce = new RedCircle();
26          redcir1ce.x = 100;
27          redcir1ce.y = 100;
28          addChild(redcir1ce);
29      }
30
31  }
```

```
Main.hx  RedCircle.hx
1  package src;
2  import openfl.display.Sprite;
3
4  /**
5   * ...
6   * @author Zergling
7   */
8  class RedCircle extends Sprite
9  {
10     var circle: Sprite;
11
12     public function new()
13     {
14         super();
15
16         circle = new Sprite();
17         circle.graphics.beginFill(0xff0000);
18         circle.graphics.drawCircle(0, 0, 25);
19         addChild(circle);
20     }
21
22 }
```

# Небольшой пример



# Еще пример (с листнером)

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

```
Main.hx RedCircle.hx
1 package src;
2
3 import openfl.display.Sprite;
4 import openfl.events.Event;
5
6 class RedCircle extends Sprite
7 {
8     var circle: Sprite;
9     var frames: Int;
10
11    public function new()
12    {
13        super();
14
15        circle = new Sprite();
16        circle.graphics.beginFill(0xff0000);
17        circle.graphics.drawCircle(0, 0, 25);
18        addChild(circle);
19
20        frames = 0;
21
22        addEventListener(Event.ENTER_FRAME, Circle_Onframe);
23    }
24
25    public function Circle_Onframe(e: Event)
26    {
27        frames++;
28
29        x = x + Math.cos(frames / 100);
30        y = y + Math.sin(frames / 100);
31    }
32
33 }
```

# 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 NaHe 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
1  package src;
2
3  import openfl.display.Sprite;
4  import openfl.events.Event;
5
6  class RedCircle extends Sprite
7  {
8      var circle: Sprite;
9      var frames: Int;
10
11     public var publicInt: Int;
12     var nonpublicInt: Int;
13
14     public function new()
15     {
16         super();
17
18         circle = new Sprite();
19         circle.graphics.beginFill(0xff0000);
20         circle.graphics.drawCircle(0, 0, 25);
21         addChild(circle);
22
23         frames = 0;
24
25         publicInt = 0;
26         nonpublicInt = 0;
27
28         addEventListener(Event.ENTER_FRAME, Circle_Onframe);
29     }
30
31     public function Circle_Onframe(e: Event)
32     {
33         frames++;
34
35         x = x + Math.cos(frames / 100);
36         y = y + Math.sin(frames / 100);
37     }
38
39 }

```

```

Main.hx*  RedCircle.hx
1  package;
2
3  import openfl.display.Sprite;
4  import openfl.Lib;
5  import src.*;
6
7  class Main extends Sprite
8  {
9      var rc: RedCircle;
10     var redcirclce: RedCircle;
11
12     public function new()
13     {
14         super();
15
16         // Assets:
17         // openfl.Assets.getBitmapData
18         /*
19         rc = new RedCircle();
20         addChild(rc);
21         */
22
23         redcirclce = new RedCircle();
24         redcirclce.x = 100;
25         redcirclce.y = 100;
26         addChild(redcirclce);
27
28         redcirclce.publicInt = 100;
29         redcirclce.nonpublicInt = 100;
30     }
31
32 }
33

```

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 посмотреть примеры.