



**КЛАССЫ**

# ОПРЕДЕЛЕНИЕ КЛАССА

```
class Dog{  
  
}
```

- **Класс** — это шаблон для создания объектов

# ОПРЕДЕЛЕНИЕ КЛАССА

```
class Dog{  
    val name = "Gray"  
    def woof() = println("гав")  
}
```

- Его тело может содержать методы, значения и определения типов

# ОПРЕДЕЛЕНИЕ КЛАССА

```
class Dog{  
    val name = "Gray"  
    def woof() = println(s"$name говорит: гав")  
}
```

- Они могут ссылаться друг на друга

# ОПРЕДЕЛЕНИЕ КЛАССА

```
class Dog
```

- Тело класса можно оставить пустым

# СОЗДАНИЕ ЭКЗЕМПЛЯРА

```
class Dog{  
    val name = "Gray"  
    def woof() = println(s"$name говорит: гав")  
}  
  
val dog = new Dog
```

# СОЗДАНИЕ ЭКЗЕМПЛЯРА

```
class Dog{  
    val name = "Gray"  
    def woof() = println(s"$name говорит: гав")  
}  
  
val dog = new Dog  
  
println(dog.name)  
dog.woof()
```

# ПАРАМЕТРЫ

```
class Dog(name: String){  
    def woof() = println(s"$name говорит: гав")  
}  
  
val dog = new Dog("Gray")  
dog.woof()
```

- Вы можете определять параметры. Все параметры должны быть переданы при создании объекта



# ПАРАМЕТРЫ

```
class Dog(val name: String){  
    def woof() = println("$name говорит: гав")  
}  
  
val dog = new Dog("Gray")  
println(dog.name)  
dog.woof()
```

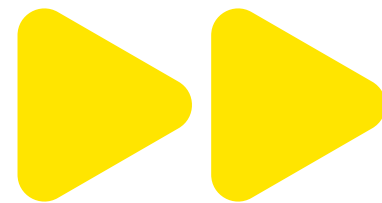
- Параметры могут быть определены как значения
- Тогда на них можно явно ссылаться как на члены объекта

# ИНИЦИАЛИЗАЦИЯ

```
class Dog(name: String){  
    println("$name родился!!!")  
  
    def woof() = println(s"$name говорит: гав")  
}  
  
val dog = new Dog("Gray") // Gray родился!!!  
dog.woof()
```

- Классы могут содержать блоки инициализации. Они будут выполняться каждый раз, когда создаётся экземпляр

**В этом разделе  
мы изучили классы**



**В следующем  
практика**