
# НАСЛЕДОВАНИЕ

# НАСЛЕДОВАНИЕ

```
class A extends P1 with P2 with P3 with ...
trait B extends P1 with P2 with P3 with ...
object C extends P1 with P2 with P3 with ...
case class D() extends P1 with P2 with ...
```

- Любые определения типов могут содержать список базовых типов

# НАСЛЕДОВАНИЕ

```
trait Animal{
  def name: String
}

trait Woofing extends Animal{
  def woof() = println(s"$name говорит: гав")
}
```

- Определения из базовых типов автоматически становятся доступными в их наследниках ...

# НАСЛЕДОВАНИЕ

```
trait Animal{
  def name: String
}

trait Woofing extends Animal{
  def woof() = println(s"$name говорит: гав")
}

val woofer = new Woofing{
  def name = "Барбос"
}

woofer.woof
println(woofer.name)
```

- ... и их экземплярах

# ПЕРЕОПРЕДЕЛЕНИЕ

```
trait Animal{
  def name: String
  def greeting = s"Привет, я - $name"
}

object Pegasus extends Animal {
  val name = "Pegasus"
}

println(Pegasus.greeting)
```

- Абстрактные определения могут быть заменены на конкретные при наследовании
- Def может быть переопределён как **val** или **lazy val**

# ПЕРЕОПРЕДЕЛЕНИЕ

```
trait Animal{
  def name: String
  def greeting = s"Привет, я - $name"
}

object Pegasus extends Animal {
  val name = "Pegasus"
  override def greeting =
   s"Я - $name, крылатый скакун"
}

println(Pegasus.greeting)
```

- Конкретные определения могут быть заменены на другие с помощью ключевого слова **override**

# ПЕРЕОПРЕДЕЛЕНИЕ

```
trait Animal{
  def name: String
  def greeting = s"Привет, я - $name"
}

object Pegasus extends Animal {
  val name = "Pegasus"
  override def greeting =
   s"{$super.greeting}, крылатый скакун"
}

println(Pegasus.greeting)
```

- При переопределении можно сослаться на метод предка через **super**

# НАСЛЕДОВАНИЕ ОТ КЛАССОВ

```
abstract class Animal

trait Greeter

object Pegasus extends Animal with Greeter

// bad
// object Pegasus extends Greeter with Animal
```

- Если в списке наследования есть класс, он должен идти первым в списке наследования

# НАСЛЕДОВАНИЕ ОТ КЛАССОВ

```
abstract class Animal
abstract class Plant

trait Greeter

object Pegasus extends Animal with Greeter

// bad
// object Pegasus extends Animal
//                    with Plant
//                    with Greeter
```

- Можно указывать только один базовый класс

# НАСЛЕДОВАНИЕ ОТ КЛАССОВ

```
abstract class Animal(name: String)

trait Greeter

object Pegasus extends Animal("Pegasus")
                  with Greeter
```

- Если у базового класса есть параметры, вы должны передать их при определении объекта или класса

# НАСЛЕДОВАНИЕ ОТ КЛАССОВ

```
abstract class Animal(name: String)

trait Greeter extends Animal

Pegasus extends Animal("Pegasus")
                    with Greeter
```

- Но если вы определяете trait параметры, передавать не нужно
- Но при определении конкретного наследника этого trait, вам нужно будет обязательно унаследовать этот базовый класс

# ЛИНЕАРИЗАЦИЯ TRAIT

```
trait Animal
trait Greeter extends Animal
trait Mammal extends Animal

Pegasus extends Animal with Greeter with Mammal
```

- Цепочки из trait, унаследованных от одного базового trait, упорядочивают свои реализации особенным образом
- Подробнее на практике

**В этом разделе мы изучили наследование**

В следующем практика