
# ОБОБЩЁННЫЕ МЕТОДЫ

# ОБОБЩЁННЫЕ МЕТОДЫ

```
def ifThenElseInt(cond: Boolean)
                 (t: => Int,  e: => Int): Int =
  if(cond) t else e
```

```
def ifThenElseString(cond: Boolean)
                    (t: => String, e: => String): String =
  if(cond) t else e
```

```
def ifThenElseLong(cond: Boolean)
                  (t: => Long,  e: => Long): Long =
  if(cond) t else e
```

```
def ifThenElse[A](cond: Boolean, t: => A,  e: => A): A =
  if(cond) t else e
```

# ОБОБЩЁННЫЕ МЕТОДЫ

```
def ifThenElse[A](cond: Boolean, t: => A,  e: => A): A =
  if(cond) t else e
```

- Методы могут иметь параметры типа
- Параметры можно использовать в типах параметров или возвращаемого значения
- Такие методы можно переиспользовать для разных типов

```
ifThenElse[String](1 > 2, "one", "two") // :String = "two"
ifThenElse[Int](1 > 2, 1, 2) // :Int = 2
```

# ОБОБЩЁННЫЕ МЕТОДЫ

```
def ifThenElse[A](cond: Boolean, t: => A,  e: => A): A =
  if(cond) t else e
```

- Если компилятор может вывести параметры типа на основе входных параметров, их можно не указывать

```
ifThenElse(1 > 2, "one", "two") // :String = "two"
ifThenElse(1 > 2, 1, 2) // :Int = 2
```

# КОРОТКАЯ ЗАПИСЬ

```
def combineOn[A, B](comb: (B, B) => B)
                   (f: A => B, g: B => A)
                   : (A, A) => A =
  (x, y) => g(comb(f(x), f(y))
```

- Их удобно использовать для методов принимающих функции в качестве параметров

```
val sumStrings =
  combineOn[String, Int](_ + _)(_.toInt, _.toString)
sumStrings("123", "32") //: String = "155"
```

# Мы изучили обобщённые методы

В следующем разделе практика