ОПЦИОНАЛЬНЫЕ ЗНАЧЕНИЯ

ОПЦИОНАЛЬНЫЕ ЗНАЧЕНИЯ

```
Option[A]
```

Означает, что в данном месте может быть значение типа A, а может не быть ничего

Для этого у него есть ровно два подтипа

```
Some[A]
```

Контейнер, содержащий значение

```
None
```

Объект, означающий отсутствие значения

ОПЦИОНАЛЬНЫЕ ЗНАЧЕНИЯ

- Option[A] используется для решения проблем null reference
- Всюду, где для обозначения возможного отсутствия значения может использоваться пустая ссылка, в scala используется Option[A]
- В отличие от пустой ссылки, Option[A] заставляет потребителя явно задуматься, что делать, если значения нет

ОПЦИОНАЛЬНЫЕ ЗНАЧЕНИЯ

```
def divide(x: Int, y: Int): Option[Int] =
    if(y == 0) None else Some(x / y)
```

ОПЦИОНАЛЬНЫЕ ЗНАЧЕНИЯ

```
def divide(x: Int, y: Int): Option[Int] =
    if(y == 0) None else Some(x / y)

def showDivide(x: Int, y: Int): String =
  divide(x, y) match {
    case Some(d) => s"$x = $d * $y"
    case None => "null division"
  }
```

ЗАМЕНА ПУСТОГО ЗНАЧЕНИЯ

на значение по умолчанию: getOrElse

```
divide(7, 0).getOrElse(1)
```

на другое опциональное значение: orElse

```
divide(7, 0).orElse(divide(7, 2))
```

ПРЕОБРАЗОВАНИЕ

обычной функцией: map

```
divide(7, 4).map(x => x + 6)
```

функцией, возвращающей опциональное значение: flatMap

```
divide(17, 3).flatMap(x => divide(x, 3))
```

ФИЛЬТРАЦИЯ

предикатом: filter

```
divide(7, 4).filter(x => x > 2)
```

частичной функцией: collect

```
divide(17, 3).collect{
    case x if x > 4 => x + 4
}
```

ОПЦИОНАЛЬНЫЕ ЗНАЧЕНИЯ

```
def divide(x: Int, y: Int): Option[Int] =
    if(y == 0) None else Some(x / y)

def showDivide(x: Int, y: Int): String =
  divide(x, y)
    .map(d => s"$x = $d * $y")
    .getOrElse("null division")
```

В этом разделе мы изучили опциональные значения

В следующем рассмотрим Either