

彻底掌握Kotlin

定义泛型类

泛型函数

泛型约束

A

B

C

The background features a faint watermark of the Android robot logo, which is a yellow robot head with a single eye and a smiling mouth, wearing a green shirt with a white "A" on it.

ANdROID



定义泛型类

定义泛型类

- 泛型类的构造函数可以接受任何类型。
- MagicBox类指定的泛型参数由放在一对<>里的字母T表示，T是个代表item类型的占位符。MagicBox类接受任何类型的item作为主构造函数值(item: T)，并将item值赋给同样是T类型的subject私有属性。

```
1  class MagicBox<T>(item: T) {  
2      private var subject: T = item  
3  }  
4  class Boy(val name: String, val age: Int)  
5  class Dog(val weight: Int)  
6  
7  fun main() {  
8      val box1: MagicBox<Boy> = MagicBox(Boy(name: "Jack", age: 20))  
9      val box2: MagicBox<Dog> = MagicBox(Dog(weight: 20))  
10 }
```

注意

➤ 泛型参数通常用字母T（代表英文type）表示，当然，想用其他字母，甚至是英文单词都是可以的。不过，其他支持泛型的语言都在用这个约定俗成的T，所以建议你继续用它，这样写出的代码别人更容易理解。

A large, semi-transparent Android robot head is positioned in the center-left of the slide. It has a yellow body, a white face with black eyes and a small smile, and a red antenna on top. The word "ANDROID" is written in a white, sans-serif font to the right of the robot's head.

ANDROID



泛型函数

泛型函数

- 泛型参数也可以用于函数。
- 定义一个函数用于获取元素，当且仅当MagicBox可用时，才能获取元素。

```
1  class MagicBox<T>(item: T) {  
2      var available = false  
3      private var subject: T = item  
4  
5      fun fetch(): T? {  
6          return subject.takeIf { available }  
7      }  
8  }  
9  class Boy(val name: String, val age: Int)  
10  
11 class Dog(val weight: Int)  
12 fun main() {  
13     val box1: MagicBox<Boy> = MagicBox(Boy(name: "Jack", age: 20))  
14     val box2: MagicBox<Dog> = MagicBox(Dog(weight: 20))  
15     box1.available = true  
16     box1.fetch()?.run { this: Boy  
17         println("you find $name")  
18     }  
19 }
```

多泛型参数

➤ 泛型函数或泛型类也可以有多个泛型参数。

```
1  class MagicBox<T>(item: T) {
2      var available = false
3      private var subject: T = item
4
5      fun fetch(): T? {
6          return subject.takeIf { available }
7      }
8      fun <R> fetch(subjectModFunction: (T) -> R): R? {
9          return subjectModFunction(subject).takeIf { available }
10     }
11 }
12 fun main() {
13     val box1: MagicBox<Boy> = MagicBox(Boy(name: "Jack", age: 20))
14     box1.available = true
15     //男孩变成了男人，返回
16     val man: Man? = box1.fetch() { it: Boy
17         Man(it.name, it.age.plus(other: 10))
18     }
19     man?.let { println("${it.name}, ${it.age}") }
20 }
```

The background features a faint watermark of the Android robot logo, which is a yellow robot head with two antennae and a smiling mouth, surrounded by a circular path of dots.

ANdROID

 A purple lightbulb icon with yellow rays emanating from it, positioned to the left of the title text.

泛型类型约束

泛型类型约束

- 如果要确保MagicBox里面只能装指定类型的物品，如Human类型，怎么办？

```
1  class MagicBox<T : Human>(item: T) {
2      var available = false
3      var subject: T = item
4
5      fun fetch(): T? {
6          return subject.takeIf { available }
7      }
8  }
9
10 fun main() {
11     val box1: MagicBox<Human> = MagicBox(Boy(name: "Jack", age: 20))
12 }
```

vararg关键字与get函数

- MagicBox能存放任何类型的Human实例，但一次只能放一个，如果需要放入多个实例呢？

```
1  class MagicBox<T : Human>(vararg item: T) {  
2      var available = false  
3      var subject: Array<out T> = item  
4  
5      fun fetch(index: Int): T? {  
6          return subject[index].takeIf { available }  
7      }  
8  
9      fun <R> fetch(index: Int, subjectModFunction: (T) -> R): R? {  
10         return subjectModFunction(subject[index]).takeIf { available }  
11     }  
12 }
```

[]操作符取值

➤ 想要通过[]操作符取值，可以重载运算符函数get函数。

```
1  class MagicBox<T : Human>(vararg item: T) {
2      var available = false
3      var subject: Array<out T> = item
4
5      operator fun get(index: Int): T? = subject[index].takeIf { available }
6
7      fun fetch(index: Int): T? {
8          return subject[index].takeIf { available }
9      }
10
11     fun <R> fetch(index: Int, subjectModFunction: (T) -> R): R? {
12         return subjectModFunction(subject[index]).takeIf { available }
13     }
14 }
```