TOUR OF SCALA

UPPER TYPE BOUNDS

In Scala, type parameters and abstract type members may be constrained by a type bound. Such type bounds limit the concrete values of the type variables and possibly reveal more information about the members of such types. An *upper type bound* `T <: A` declares that type variable `T` refers to a subtype of type `A`. Here is an example that demonstrates upper type bound for a type parameter of class `PetContainer`:

```
abstract class Animal {
 def name: String
}

abstract class Pet extends Animal {}

class Cat extends Pet {
  override def name: String = "Cat"
}

class Dog extends Pet {
  override def name: String = "Dog"
}

class Lion extends Animal {
  override def name: String = "Lion"
}

class PetContainer[P <: Pet](p: P) {
  def pet: P = p
}

val dogContainer = new PetContainer[Dog](new Dog)
val catContainer = new PetContainer[Cat](new Cat)
```

```
// this would not compile
val lionContainer = new PetContainer[Lion](new Lion)
```

The `class PetContainer` takes a type parameter `P` which must be a subtype of `Pet`. `Dog` and `Cat` are subtypes of `Pet` so we can create a new `PetContainer[Dog]` and `PetContainer[Cat]`. However, if we tried to create a `PetContainer[Lion]`, we would get the following Error:

`type arguments [Lion] do not conform to class PetContainer's type parameter bounds [P <: Pet]`

This is because `Lion` is not a subtype of `Pet`.

← previous next →

Contributors to this page:

ckipp01 mlachkar erikvanzijst ashawley dwijnand f0ster Philippus sake92 heathermiller