# Angelika Langer Enum<E extends Enum<E>> decoding

Ask Question

As per my Previous Question, I am reading the article from Angelika Dissecting Enum. Except for the points that a type can only be instantiated for its subtypes and the subtypes do inherit some common methods, I am not able to understand the article.

1. What is the meaning of abstract Enum class declared in this way? How is it helpful?
2. The document in the last part has described three aspects, can someone explain them in easier terms to me?
3. I do see in the code sketch the Enum class is declaring the `compareTo` method. When Enum is implicitly implementing `Comparable interface`. Why do it needs to define its own `compareTo` method?
4. Seems like it is a concept of recursive generics. What does recursive generics exactly mean? After doing a bit of R&D and understanding my last question answer, I understand that it forces the class to be parameterized on itself.

Still, a detailed explanation would be useful.

java generics

edited May 23 '17 at 12:05
Community ♦
1 1

asked Aug 26 '13 at 19:45
benz
1,878 4 25 45

"Why didn't it say something like..." - because that's not valid Java ;) – Oliver Charlesworth Aug 26 '13 at 19:47

Honestly speaking my emphasis is not that. I know its not valid java. My point is to understand the whole concept @OliCharlesworth – benz Aug 26 '13 at 19:50

2 In C++ the idiom is known as Curiously recurring template pattern. See also the following page for a reference to Java: en.wikipedia.org/wiki/Talk:Curiously_recurring_template_pattern – nosid Aug 26 '13 at 19:51

@nosid: except that in Java it is not useful. (neither is it safe in C++) – newacct Aug 27 '13 at 6:51

## 2 Answers

I think the main benefit of declaring generic types as `Type<E extends Type<E>>` is that such generic classes will make subclasses to inherit methods which return or accept arguments with subtype's type. Such methods in `java.lang.Enum` are:

```
public final int compareTo( E o) { ... }
public final Class< E > getDeclaringClass() { ... }
```

So, if we declare the enum `Color`, that implicitly means:

```
public class Color extends Enum<Color>
```

so in this instantiation of `Enum` the type paramater `E` is assigned the type argument `Color`, so the above methods will look like these:

```
public final int compareTo(Color o) { ... }
public final Class<Color> getDeclaringClass() { ... }
```

answered Aug 27 '13 at 15:59
Katona
**3,423** 15 23

Thanks @Katona, i understood the concept after thoroughly reading. thankyou very much. Still i am looking for point 2 and 4 clarification. – benz Aug 27 '13 at 16:32

Everything you say here applies equally if it's declared as `class Type<E>` – newacct Jan 23 '14 at 8:00

When saying something like `Enum<Color extends Enum<Color>>`, that sounds like you are declaring a generic type parameter `Color` that makes sure that it extends `Enum` with a type parameter matching `Color`.

But that isn't where generic type parameters for a class are declared. You must declare them next to the class name; you can only use them later in the `extends` clause. E.g.

```
//                 Use "extends" here  ...                   not here.
public class MyClass<E extends MyClass<E>> extends MySuperClass<E>
```

In this example, you are declaring the *class* `Color` to be the value of the generic type parameter that is already defined on `Enum` .

answered Aug 26 '13 at 19:52

rgettman
**140k** 20 188 269