Extending Enums in Java

1. Overview

The enum type, introduced in Java 5, is a special data type that represents a group of constants.

Using enums, we can define and use our constants in the way of type safety. It brings compile-time checking to the constants. Furthermore, it allows us to use the constants in the switch-case statement.

In this tutorial, we'll discuss extending enums in Java, including adding new constant values and new functionalities.

Java enums are enhanced with generics support and with the ability to add methods to individual items, a new JEP shows. Since both features can be delivered with the same code change, they are bundled together in the same JEP. The change only affects the Java compiler, and therefore no runtime changes are needed.

An enum with generics support would allow us to indicate the set of available keys together with their associated type:

```
public enum Key<T> {
    HOST<String>,
    PORT<Integer>,
    SCORE<Double>}
public interface PropertiesStore {
    public <T> void put(Key<T> key, T value);
    public <T> T get(Key<T> key);}
```

Now these keys can be safely retrieved and stored from and into the properties store, since expressions like the following will fail to compile:

```
put(PORT, “not a number”); // error, type mismatch: PORT is Key<Integer>
                          // “not a number” is String
```

2. Enums and Inheritance

When we want to extend a Java class, we typically create a subclass. In Java, enums are classes as well.

In this section, we'll see if we can inherit an enum, as we do with regular Java classes.

2.1. Extending an Enum Type

First, let's look at an example, so we can quickly understand the problem:

```
public enum BasicStringOperation {
    TRIM("Removing leading and trailing spaces."),
    TO_UPPER("Changing all characters into upper case."),
    REVERSE("Reversing the given string.");
    private String description;
    // constructor and getter
}
```

As the code above shows, we have an enum, BasicStringOperation, that contains three basic string operations.

Now let's say we want to add some extension to the enum, such as MD5_ENCODE and BASE64_ENCODE. We may come up with this straightforward solution:

```
public enum ExtendedStringOperation extends BasicStringOperation {
    MD5_ENCODE("Encoding the given string using the MD5 algorithm."),
    BASE64_ENCODE("Encoding the given string using the BASE64 algorithm.");
    private String description;
    // constructor and getter
}
```

However, when we attempt to compile the class, we'll see the compiler error:

Cannot inherit from enum BasicStringOperation

2.2. Inheritance Is Not Allowed for Enums

Let's figure out why we received a compiler error.

When we compile an enum, the Java compiler does some magic to it:

- **It turns the enum into a subclass of the abstract class java.lang.Enum**
- **It compiles the enum as a final class**

For example, if we disassemble our compiled BasicStringOperation enum using javap, we'll see it's represented as a subclass of java.lang.Enum<BasicStringOperation>:

```
$ javap BasicStringOperation
public final class com.baeldung.enums.extendenum.BasicStringOperation
    extends
java.lang.Enum<com.baeldung.enums.extendenum.BasicStringOperation
> {
  public static final
com.baeldung.enums.extendenum.BasicStringOperation TRIM;
  public static final
com.baeldung.enums.extendenum.BasicStringOperation TO_UPPER;
  public static final
com.baeldung.enums.extendenum.BasicStringOperation REVERSE;
}
```

As we know, we can't inherit a final class in Java. Moreover, even if we could create the ExtendedStringOperation enum to inherit BasicStringOperation, our ExtendedStringOperation enum would extend two classes: BasicStringOperation and java.lang.Enum. That is to say, it would become a multiple inheritance situation, which isn't supported in Java.

Read More : https://www.baeldung[.]com/java-extending-enums