

Blog

We believe there is something unique at every business that will ignite the fuse of innovation.

Insights (/insights) > Blogs (/insights/blog) > Java Enumerations (/blogs/java-enumerations)

August 3, 2013

Java Enumerations

Written By:

WASEEM QURAISHI (/search#q=Waseem Quraishi)

TAGS

software engineering



Enumerations

Per Sun's Java documentation, a Java enum *"is a type whose fields consist of a fixed set of constants ... you should use enum types any time you need to represent a fixed set of constants."*

- Prior to Java 1.5, there were 2 basic ways to define new types: classes and interfaces
- In some cases neither one of the options was sufficient, especially for a finite set of a specific type of data
- Example of some finite sets include grades, planets, compass directions, countries, genders, and ethnicity types
- These types of construct were possible prior to 1.5 but required a lot of work and were prone to issues.
- Enums remove the need for `public static final` constants.

Defining an Enum

Defining an enum is as simple as creating any other class in Java.

```
public enum Gender {  
    MALE, FEMALE, UNKNOWN (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+unknown)  
};
```

- In this example, Gender is the enumeration type whereas MALE, FEMALE, and UNKNOWN are values for that type.
- Note: The enumerated type identifiers are constants, therefore **UPPERCASE** lettering is the standard convention

Three basic components (at minimum) to create an enumerated type are

1. The `enum` keyword
2. A name for the enumerated type (Gender)
3. A list of allowable values for the type (`MALE` , `FEMALE` , `UNKNOWN`)

In addition to the minimum requirements, an enumeration can also contain the following optional components

1. An interface or set of interfaces that the enum implements
2. Variable definitions
3. Method definitions
4. Value-specific class bodies

Referencing an Enum

Enumerations are referenced in the same manner as any other class because an `enum` is a class. As a result you get all the benefits of a class such as type-safety, compile time checking, and the ability to use them in variable definitions.

```
public class Person {
    private String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) firstName;
    private String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) lastName;
    private Date (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+date) dtOfBirth;
    private Gender gender;

    public Person(String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) fName, String (http://www.google.com/search?hl=en&q=allinurl%3A
        this.firstName = fName;
        this.lastName = lName;
    }

    public void setFirstName(String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) firstName) {
        this.firstname = firstName;
    }

    public String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) getFirstName() {
        return firstName;
    }

    ...

    public void setGender(Gender gender) {
        this.gender = gender;
    }

    public void getGender() {
        return gender;
    }
}
```

As you can see, there is nothing complicated about this. The way of defining and referencing an enumeration in a class is like any other JAVA POJO type.

Using an Enum

Continuing with our Gender and Person example above, let's see an example of how this enumeration can be used in a JAVA program.

```
public class EnumTester {  
    public static void main(String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] args) {  
        Person lisa = new Person ("Lisa", "Simpson");  
        Person homer = new Person ("Homer", "Simpson");  
  
        homer.setGender(Gender.MALE);  
        lisa.setGender(Gender.FEMALE);  
  
        System (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Lisa's gender is: " + lisa.getGender());  
        System (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Homer's gender is: " + homer.getGender());  
    }  
}
```

The output produced from the above call is

Lisa's gender is FEMALE

Homer's gender is MALE

Prior to Java 1.4

As mentioned, enums were introduced in the 1.5 version of Java. Prior to this release, the standard method to represent an enumerated type was using the `int` Enum pattern.

```
public class Gender {  
    public static final int GENDER_MALE = 0;  
    public static final int GENDER_FEMALE = 1;  
    public static final int GENDER_UNKNOWN = 3;  
}
```

At first glance the code seems simpler, but it is prone to many problems including the following:

- Not Type Safe: The gender is just an primitive `int`, therefore it can be set to any acceptable `int` value. In addition, you can perform operations which make no sense such as performing mathematical calculations on the gender enum.
- Namespace: To avoid namespace collisions, the prefixes of the enum `int` constant must be prepended with a string (`GENDER_`).
- Brittle Code: `int` enums are compile time constants. The enums are compiled into classes which use them. If new constants are added or order is changed, the clients need to be recompiled. Although this will not produce a run time error, the behavior will be erratic and undefined.
- Verbosity and/or useful information: Printing an `int` enum produces a number which gives no information pertaining to the type or what it represents.

Additional Notes

As previously noted, Java enumerations are classes therefore receive the benefits of a Java class (see above).

- Enumerations implicitly extend `java.lang.Enum`.
- Each declared values is an instance of that enum class.
- Enums have no public constructor. This prevents the ability to modify enums at run-time.
- Enums values cannot be subclassed. They are essentially final classes.
- Enums can be compared using the `==` or `equals()` method.
- Enums can be compared using the `compareTo()` method because they implement the `java.lang.Comparable` interface.
- Enums override the `toString()` method. This method returns the name of the enumerated value. This method is not `final` therefore can be overridden.
- Enums provide a `valueOf()` method which complements the `toString()` method.
- Using the above example
`Gender.valueOf("MALE")` returns `Gender.MALE`;

- Although this shouldn't be used directly in your code, enums define a `final` instance method called `ordinal()` which returns the position (0 based) of the enumerated type.
- Enums also provide a `values()` method used during the iteration of enumerations.

Next

In future blogs, we'll dive a little deeper into enums to cover the following topics:

1. Inline Enums
2. Iterating Enums
3. Switching Enums
4. Maps of Enums
5. Sets of Enums
6. Adding Methods to Enums
7. Implementing Interfaces with Enums
8. Extending an Enum

Attachments

test (/library/captech/blogs/java-enumerations/test.zip)






CapTech

- Services > (/services)
- Expertise > (/expertise)
- Insights > (/insights)
- Careers > (/careers)
- About Us > (/about)
- Contact > (/contact)

Locations

- Atlanta, GA > (/contact/atlanta)
- Denver, CO > (/contact/denver)
- Baltimore, MD > (/contact/baltimore)
- Orlando, FL > (/contact/orlando)
- Charlotte, NC > (/contact/charlotte)
- Philadelphia, PA > (/contact/philadelphia)
- Chicago, IL > (/contact/chicago)
- Richmond, VA > (/contact/richmond)
- Columbus, OH > (/contact/columbus)
- Washington, DC Metro > (/contact/dc)

Connect With Us

-  https://www.facebook.com/CapTechCareers?_rdrr
-  <https://instagram.com/lifeatcaptech>
-  <https://www.linkedin.com/company/captech-ventures>
-  <https://twitter.com/CapTechListens>
-  <https://www.youtube.com/user/CapTechConsulting>

Tweets



CapTech @CapTechListens

We are thrilled to be a Silver Sponsor of this years' @BestBuddiesCR Friendship Walk on Saturday
captechconsulting.com/news/captech-t...



Oct 20, 2017

©2017 CapTech Ventures, Inc. All Rights Reserved. Legal Notices (/library/captech/legal-notices/legal_notices_may_2016.pdf)