**Java Enum:**

In this tutorial I will explain what is enum, how to use enum in different areas of a Java program and an example program on it.

An Enum type is a special data type which is added in Java 1.5 version. It is an abstract class in Java API which implements Clonable and Serializable interfaces. It is used to define collection of constants. When you need a predefined list of values which do not represent some kind of numeric or textual data, at this moment, you have to use an enum.

Common examples include days of week (values of SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, and SATURDAY), directions, and months in a year. Etc.

You can declare simple Java enum type with a syntax that is similar to the Java class declaration syntax. Let’s look at several short Java enum examples to get you started.

**Syntax:**

|  |
| --- |
| public enum Day {  SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY  } |

Or

|  |
| --- |
| public enum Month{  JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER  } |

Enums are constants; they are by default static and final. That’s why the names of an enum type's fields are in uppercase letters.

Make a note that the enum keyword which is used in place of class or interface. The Java enum keyword signals to the Java compiler that this type definition is an enum.

You can refer to the constants in the enum above like this:

|  |
| --- |
| Day day = Day.MONDAY; |

Here the ‘day’ variable is of the type ‘Day’ which is the Java enum type defined in the example above. The ‘day’ variable can take one of the ‘Day’ enum constants as value (SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY). In this case ‘day’ is set to MONDAY.

If you use enum instead of integers or String codes, you increase compile-time checking and avoid errors from passing in invalid constants, and you document which values are legal to use.

**How to use a Java enum type in various areas in a Java Program:**

We have seen how to declare simple Java enum types, let’s take a look at how to use them in various areas. We have to use a Java enum type in a variety of situations, including in a Java 5 for loop, in a switch statement, in an if/else statement, and more. For simple, Enum comparison, there are 3 ways to do it. They are, Switch-Case Statement, == Operator, .equals() method. Like that there are other places where you have to use Enum.

Let's take a look at how to use our simple enum types with each of these Java constructs.

1. **Enum in IF statements:**

We know Java Enums are constants. Sometimes, we have a requirement to compare a variable of Enum constant against the possible other constants in the Enum type. At this moment, we have to use IF statement as follows.

|  |
| --- |
| Day day = ----- //assign some Day constants to it.  If(day ==Day.MONDAY){  ….//your code  } else if (day == Day.TUESDAY){  ….//your code  } else if (day == Day.WEDNESDAY){  ….//your code  } else if (day == Day. THURSDAY){  ….//your code  } else if (day == Day.FRIDAY){  ….//your code  } else if (day == Day.SATURDAY){  ….//your code  } else if (day == Day.SUNDAY){  ….//your code  } |

Here, you can use “.equals()” instead of “==”. But, my preference is to use “==” because, it will never throws NullPointerException, safer at compile-time, runtime and faster.

The code snippet compares the ‘day’ variable against each of the possible other Enum constants in the ‘Day’ Enum.

1. **Enums in Switch Statements:**

Just assume that your Enum have lot of constants and you need to check a variable against the other values in Enum. For a small Enum IF Statement will be OK. If you use same IF statement for lot of Enum constants then our program will be increased and it is not a standard way to write a Java program. At this Situation, use Switch Statement instead of IF statement.

You can use enums in switch statements as follows:

|  |
| --- |
| Day day = ... //assign some Day constant to it  switch (day) {  case SUNDAY : //your code; break;  case MONDAY //your code; break;  case TUESDAY : //your code; break;  case WEDNESDAY : //your code; break;  case THURSDAY: //your code; break;  case FRIDAY : //your code; break;  case SATURDAY : //your code; break;  } |

Here, give your code in the comments, “//your code”. The code should be a simple Java operation or a method call..etc.

1. **Enum Iteration in for-each loop:**

You can obtain an array of all the possible values of a Java enum type by calling its static values() method. All enum types get a static values() method automatically by the Java compiler. Here is an example of iterating all values of an enum:

|  |
| --- |
| for (Day day : Day.values()) {  System.out.println(day);  }  If you run this code, then you will get the following output:  SUNDAY  MONDAY  TUESDAY  WEDNESDAY  THURSDAY  FRIDAY  SATURDAY |

1. **Enum fields:**

It is possible to add fields to Java Enum. Each constant enum value gets these fields. The field values must be supplied to the constructor of the enum when defining the constants. Here is an example:

|  |
| --- |
| public enum Day {  SUNDAY(1), MONDAY(2), TUESDAY(3), WEDNESDAY(4), THURSDAY(5), FRIDAY(6), SATURDAY(7);  private final int dayCode;  private Day(int dayCode) {  this.dayCode = dayCode;  }  } |

The example above has a constructor which takes an int. The enum constructor sets the int field. When the constant enum values are defined, an int value is passed to the enum constructor.

The enum constructor must be either private or package scope (default). You cannot use public or protected constructors for a Java enum.

1. **Enum Methods:**

It is possible to add methods to Java Enum. Here I am showing an example that how to do it.

|  |
| --- |
| public enum Day {  SUNDAY(1), MONDAY(2), TUESDAY(3), WEDNESDAY(4), THURSDAY(5), FRIDAY(6), SATURDAY(7);  private final int dayCode;  private Day(int dayCode) {  this.dayCode = dayCode;  public int getDayCode() {  return this.dayCode;  }    } |

It is possible to call an Enum method through a reference to one of the constant values. The following code will show you how to call Java Enum method.

|  |
| --- |
| Day day = Day.MONDAY;  System.out.println(day.getDayCode()); //which gives 2.Because the dayCode field for the Enum constant MONDAY is 2. |

Some details to know about enum:

* Java enums extend the java.lang.Enum class implicitly, so your enum types cannot extend another class. Please refer the following code. This code is from Java API.

|  |
| --- |
| **public** **abstract** **class** Enum<E **extends** Enum<E>> **implements** Comparable<E>,Serializable {  // Enum class code in Java API  } |

* If a Java enum contains fields and methods, the definition of fields and methods must always come after the list of constants in the enum. Additionally, the list of enum constants must be terminated by a semicolon;
* Java does not support multiple [inheritance](http://crunchify.com/understanding-java-annotation-annotation-examples/), and enum also does not support for multiple inheritance.
* Enums are type-safe. Enum has their own name-space. It means your enum will have a type for example “Company” in below example and you cannot assign any value other than specified in Enum Constants.
* Enum constants are implicitly static and final and cannot be changed once created.
* Enum can be safely compare using:

Switch-Case Statement

== Operator

.equals() method

* You cannot create instance of enums by using new operator in Java because constructor of Enum in [Java](http://crunchify.com/category/java-web-development-tutorial/) can only be private and Enums constants can only be created inside Enums itself.
* Instance of Enum in Java is created when any Enum constants are first called or referenced in code
* An enum can be declared outside or inside a class, but NOT in a method.
* An enum declared outside a class must NOT be marked static, final , [abstract](http://crunchify.com/what-is-an-abstract-class-and-abstract-method-in-java-when-should-i-use-it/), protected , or private
* Enums can contain [constructors](http://crunchify.com/create-simple-pojo-and-multiple-java-reflection-examples/), methods, variables, and constant class bodies.
* Enum constructors can have arguments, and can be [overloaded](http://crunchify.com/java-method-overriding-examples-and-concepts-overriding-rules/).
* Enum constructors can have arguments, and can be [overloaded](http://crunchify.com/java-method-overriding-examples-and-concepts-overriding-rules/).
* The constructor for an enum type must be package-private or private access. It automatically creates the constants that are defined at the beginning of the enum body. You cannot invoke an enum constructor yourself

**Example Program:**

|  |
| --- |
| package com.pamu.test;  /\*\*This is an Enum which indicates possible days in a week \*\*/  public enum Day {  SUNDAY, MONDAY, TUESDAY, WEDNESDAY,THURSDAY, FRIDAY, SATURDAY;  } |
| package com.pamu.test;  /\*\* This class is an example of Enum \*\*/  public class EnumTest {  Day day;  public EnumTest(Day day) {  this.day = day;  }  //method to get descriptions of the respected days in a switch-statement.  public void getDayDescription() {  switch (day) {  case MONDAY:  System.out.println("Monday is First day in a week.");  break;  case FRIDAY:  System.out.println("Friday is a partial working day.");  break;  case SATURDAY:  System.out.println("Saturday is a Weekend.");  break;  case SUNDAY:  System.out.println("Sunday is a Weekend.");  break;  default:  System.out.println("Midweek days are working days.");  break;  }  }  public static void main(String[] args) {  EnumTest day1 = new EnumTest(Day.MONDAY);  day1.getDayDescription();  EnumTest day2 = new EnumTest(Day.TUESDAY);  day2.getDayDescription();  EnumTest day3 = new EnumTest(Day.WEDNESDAY);  day3.getDayDescription();  EnumTest day5 = new EnumTest(Day.FRIDAY);  day5.getDayDescription();  EnumTest day6 = new EnumTest(Day.SATURDAY);  day6.getDayDescription();  EnumTest day7 = new EnumTest(Day.SUNDAY);  day7.getDayDescription();  }  }  The output is,  Monday is First day in a week.  Midweek days are working days.  Midweek days are working days.  Friday is a partial working day.  Saturday is a Weekend.  Sunday is a Weekend. |

Here I wrote this example by using Switch-Statement. However, you can try this example by using different approaches. This is according to your requirement.

Please take this example which is written in Eclipse.

