## Enum in Java is a keyword, a feature which is used to represent fixed number of well-known values in Java, For example, Number of days in Week, Number of planets in Solar system etc. Enumeration (Enum) in Java was introduced in JDK 1.5

### Benefits of using Enums in Java

## 1) Enum is type-safe you can not assign anything else other than predefined Enum constants to an Enum variable. It is a compiler error to assign something else, unlike the public static final variables used in [Enum int pattern](http://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html) and [Enum String pattern](http://java67.blogspot.com/2015/03/top-40-core-java-interview-questions-answers-telephonic-round.html). 2) Enum has its own namespace. 3) The best feature of Enum is you can use Enum in Java inside Switch statement like int or char primitive data type. We will also see an example of [using java enum in switch statement](http://java67.blogspot.com/2012/09/how-to-use-java-enum-in-switch-case-example.html) in this java enum tutorial. 4) Adding new constants on [Enum in Java](http://java67.blogspot.com/2014/04/what-java-developer-should-know-about-Enumeration-type-in-Java.html) is easy and you can add new constants without breaking the existing code. Why use Enum? (advantage of enum)

* Enumeration being used as a singleton pattern so it gives thread safety benefits.
* When you deal with serialization enum is suitable options because it gives you guaranteed to be unique just check them using == operator.
* Being lazy programmer enum avoids a lot of boilerplate code.
* It is easy to use with java collections.
* It allow us to store additional value with each enum constant.
* For batter readability.
* Enum inherits all the methods of Object class and abstract class Enum. So you can use its methods for comparable, reflection, serialization etc. If you just declare a static constant instead of Enum, you can’t do this.
* Enums are iterable like collections. (Using values() method)
* Ability to use values in switch case statements without restriction.

Why not use Enum? (disadvantages of enum)

* It is much like class it’s extend java.lang.Enum so you can’t extend other enum.
* Another potential problem is you don’t get along with Expression Language (EL) in JSP.
* There are things that can be done in normal classes but maybe you can not do with enum class because of it is a special class. For example, accessing a static field in the constructor that not possible with enum.
* When you working with JSP then you can not be accessing enums nor calling enum constants because it’s not supported (which is possible after EL version 3.0).

The Java enum constants are static and final essentially. It’s likewise in Enum:

public enum RatingEnum {

EXCEEDS\_EXPECTATIONS(6), EXCELLENT(5), SUPERB(4), NICE(3), NEEDS\_IMPROVEMENT(2), POOR(1);

private int value;

public int getRating() {

return value;

}

private RatingEnum(int value) {

this.value = value;

}

}

## How to Iterate Java Enumeration

### Using Java 1.5+

for(RatingEnum rating: RatingEnum.values()){

System.out.println(rating);

}

### Using Java 8

Stream.of(RatingEnum.values()).forEach(System.out::println);

**Question 1) Can Enum implement interface in Java?**

Yes, Enum can implement interface in Java. Since enum is a type, similar to class and interface, it can implement interface. This gives a lot of flexibility to use Enum as specialized implementation in some cases.

**Question 2) Can Enum extends class in Java?**

No, Enum can not extend class in Java. Surprised, because I just said it's a type like a class or interface in Java. Well, this is why this question is a good follow-up question of *previous Enum interview question*. Since all Enum by default extend abstract base class java.lang.Enum, obviously they can not extend another class, because [Java doesn't support multiple inheritance for classes](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html). Because of extending java.lang.Enum class, all enum gets methods like ordinal(), values() or valueOf().

**Question 9) Can we use Enum with TreeSet or TreeMap in Java?**

This is really interesting question on Java Enum, I would love to ask this to gauge knowledge of Enum. Until you know about java.lang.Enum and has looked it's code, it's more likely that you don't know that **Enum implements Comparable interface**, which is main requirement to be used in Sorted Collection like [TreeSet and TreeMap](http://java67.blogspot.com/2012/08/difference-between-treemap-and-treeset-java.html). Since Enum by default impalement Comparable interface, they can be safely used inside TreeSet or TreeMap in Java.

Question 3) How do you create Enum without any instance? Is it possible without compile time error?  
yes, you can create Enum without any instance in Java, say for creating a utility class. This is another innovative way of using Enum in Java. Here is the code

**public** **enum** MessageUtil{

; *// required to avoid compiler error, also signifies no instance*

**public** **static** **boolean** isValid() {

**throw** **new** UnsupportedOperationException("Not supported yet.");

}

}

Read more: <http://www.java67.com/2013/07/15-java-enum-interview-questions-amswers-for-experienced-programmers.html#ixzz4xEgzUWy6>

**Question 4) Can we override toString() method for Enum? What happens if we don't?**

Ofcourse [you can override toString in Enum](http://javarevisited.blogspot.sg/2012/09/override-tostring-method-java-tips-example-code.html), as like any other class it also extends java.lang.Object and has toString() method available, but even if you don't override, you will not going to regret much, because abstract base class of enum does that for you and return name, which is name of the enum instance itself. here is the code of toString() method from Enum class :

**public** String toString() {

**return** name;

}

name is set, when compiler emit code for creating enum in response to instance declaration in enum class itself, along with setting ordinal, as visible in this constructor of enum from java.lang.Enum class :

**protected** Enum(String name, **int** ordinal) {

**this**.name = name;

**this**.ordinal = ordinal;

}

This is the only constructor of creating enum, which is called by code, generated by compiler in response to enum type declaration in Java program.

Read more: <http://www.java67.com/2013/07/15-java-enum-interview-questions-amswers-for-experienced-programmers.html#ixzz4xEe9T4FF>

**Question 5) Can we create instance of Enum outside of Enum itself? If Not, Why?**

No, you can not create enum instances outside of Enum boundry, because Enum doesn't have any [public constructor](http://java67.blogspot.sg/2012/12/how-constructor-chaining-works-in-java.html), and compiler doesn't allow you to provide any public constructor in Enum. Since compiler generates lot of code in response to enum type declaration, it doesn’t allow public constructors inside Enum, which enforces declaring enum instances inside Enum itself.

Question 6) Can we declare Constructor inside Enum in Java?  
Yes, you can, but remember you can only declare either private or package-private constructor inside enum. public and protected constructors are not permitted inside enum.

**Question 8) What does ordinal() method do in Enum?**

Ordinal method returns the order in which Enum instance are declared inside Enum. For example in a DayOfWeek Enum, you can declare days in order they come e.g.

**public** **enum** DayOfWeek{

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;

}

here if we call DayOfWeek.MONDAY.ordinal() it will return 0, which means it's the first instance. This ordering can be very useful to represent actual real world ordering i.e. declaring TUESDAY after MONDAY, ensures that it came after MONDAY and before WEDNESDAY. Similarly you can use enum to represent Month of year in the order they come e.g. FEBRUARY after JANUARY and before MARCH. All user defined enum inherit this method from java.lang.Enum [abstract class](http://javarevisited.blogspot.sg/2013/04/10-abstract-class-and-interface-interview-question-java-answers.html), and it's set by compiler, when it internally call protected constructor of java.lang.Enum, which accepts name and ordinal.

**Question 10) What is difference between ordinal() and compareTo() in Enum?**

This is follow-up of previous question on Java Enum. Actually, compareTo() mimic ordering provided by ordinal() method, which is the natural order of Enum. In short Enum constraints are compared in the order they are declared. Also, worth remembering is that enum constants are only comparable to other enum constants of the same enum type. Comparing enum constant  of one type to another type will result in compiler error.

### **How do you create an enum from a String value?**

Function valueOf(String) is used to convert a string to enum.

//Converting String to Enum

Season season = Season.valueOf("FALL");

Function name() is used to find String value of an enum.

//Converting Enum to String

System.out.println(season.name());//FALL

### **What is an Enum Ordinal?**

Java assigns default ordinals to an enum in order. However, it is not recommended to use ordinals to perform logic.

//Default ordinals of enum

// By default java assigns ordinals in order

System.out.println(Season.WINTER.ordinal());//0

System.out.println(Season.SPRING.ordinal());//1

System.out.println(Season.SUMMER.ordinal());//2

System.out.println(Season.FALL.ordinal());//3

### **How do you compare two Enums?**

Values of Enum’s can be compared using == or the equals function.

//Comparing two Enums

Season season1 = Season.FALL;

Season season2 = Season.FALL;

System.out.println(season1 == season2);//true

System.out.println(season1.equals(season2));//true

### **Can you use a Switch Statement around an Enum?**

Example below shows how we can use a switch around an enum.

//Using switch statement on an enum

public int getExpectedMaxTemperature() {

switch (this) {

case WINTER:

return 5;

case SPRING:

case FALL:

return 10;

case SUMMER:

return 20;

}

return -1;// Dummy since Java does not recognize this is possible }