

**What is an enum in Java, and how do you declare it?**

An enum in Java is a special type that has a fixed set of constant values. You can declare an enum using the enum keyword followed by the name of the enum and a list of constants inside curly braces. For example:

enum Season {  
 SPRING, SUMMER, AUTUMN, WINTER;  
}

**What are the advantages of using enums over constants?**

Some advantages of using enums over constants are :

* Enums improve safety by preventing invalid values from being assigned to variables of enum type.
* Enums can be placed inside or outside a class depending on their scope and visibility.
* Enums can be easily used in switch statements without requiring explicit casting or conversion.
* Enums cannot extend any class or be extended by any class, which ensures type safety and prevents inheritance issues.
* Objects of enum cannot be created using new operator, which prevents duplication and memory waste.
* Enums can have variables, constructors, and methods like regular classes.

**How do you access and iterate over the enum constants?**

To access an enum constant, you can use the dot (.) operator after the name of the enum. For example:

Season season = Season.SPRING;

To iterate over all the enum constants, you can use the values() method that returns an array of all the constants in the enum. For example:

for (Season s : Season.values()) {  
 System.out.println(s);  
}

**How do you compare two enum constants using the == operator?**

To compare two enum constants using the == operator, you just need to use it like any other primitive type. For example:

if (season == Season.SPRING) {  
 System.out.println("It's spring!");  
}

This works because each enum constant is a singleton object that has only one instance in memory.

**How do you add custom methods and fields to an enum?**

To add custom methods and fields to an enum, you need to define them after the list of constants and separate them with a semicolon (;). You also need to provide a constructor for initializing the fields if they are not static. For example:

enum Planet {  
 MERCURY(3.303e+23, 2.4397e6),  
 VENUS(4.869e+24, 6.0518e6),  
 EARTH(5.976e+24, 6.37814e6),  
 MARS(6.421e+23, 3.3972e6),  
 JUPITER(1.9e+27 ,7.1492e7),  
 SATURN(5.688e+26 ,6.0268e7),  
 URANUS(8.686e+25 ,2.5559e7),  
 NEPTUNE(1.024e+26 ,2.4746e7);  
  
// fields  
private final double mass; // in kilograms  
private final double radius; // in meters  
  
// constructor  
Planet(double mass, double radius) {  
 this.mass = mass;  
 this.radius = radius;  
}  
  
// method  
public double getSurfaceGravity() {  
 return G \* mass / (radius \* radius);  
}  
}

**How do you implement a switch statement using an enum?**

To implement a switch statement using an enum, you need to use the name of the enum as the expression and match each case with an enum constant without specifying its name again. For example:

switch (season) {  
case SPRING:  
 System.out.println("Flowers bloom");  
 break;  
case SUMMER:  
 System.out.println("Sun shines");  
 break;  
case AUTUMN:  
 System.out.println("Leaves fall");  
 break;  
case WINTER:  
 System.out.println("Snow falls");  
 break;  
default:  
 System.out.println("Invalid season");  
}

**How do you use enums with annotations and generics?**

To use enums with annotations and generics, you need to follow some rules:

* Annotations can only take primitive types or String as parameters, so you cannot pass an enum directly as an annotation parameter.
* However, you can pass an array of enums as an annotation parameter by using curly braces ({}) around them.
* Generics can take any reference type as parameters, including enums.
* However, you cannot create generic arrays with enums because arrays are covariant while generics are invariant.

**How do you get the ordinal value of an enum constant?**

You can use the ordinal() method that returns an int value representing the position of the enum constant in the declaration. For example:

enum Color {  
 RED, GREEN, BLUE;  
}  
  
Color c = Color.GREEN;  
System.out.println(c.ordinal()); // prints 1

**How do you get the name of an enum constant?**

You can use the name() method that returns a String value representing the name of the enum constant as declared. For example:

enum Color {  
 RED, GREEN, BLUE;  
}  
  
Color c = Color.BLUE;  
System.out.println(c.name()); // prints BLUE

**How do you convert a String value to an enum constant?**

You can use the valueOf() method that takes a String parameter and returns an enum constant with the same name. For example:

enum Color {  
 RED, GREEN, BLUE;  
}  
  
String s = "RED";  
Color c = Color.valueOf(s);  
System.out.println(c); // prints RED

**How do you implement an interface using an enum?**

You can implement an interface using an enum by providing a semicolon (;) after the list of constants and then implementing the interface methods for each constant. For example:

interface Operation {  
 double apply(double x, double y);  
}  
  
enum MathOperation implements Operation {  
 PLUS {   
 public double apply(double x, double y) { return x + y; }  
 },  
 MINUS {   
 public double apply(double x, double y) { return x - y; }  
 },  
 TIMES {   
 public double apply(double x, double y) { return x \* y; }  
 },  
 DIVIDE {   
 public double apply(double x, double y) { return x / y; }  
 };  
}

[Enum](https://medium.com/tag/enum?source=post_page-----63c1f69ae9d9---------------enum-----------------)