Contents

[Enums 1](#_Toc124366221)

[Enum Class in Java 3](#_Toc124366222)

[Methods of Java Enum Class 4](#_Toc124366223)

[1. Java Enum ordinal() 4](#_Toc124366224)

[2. Enum compareTo() 5](#_Toc124366225)

[3. Enum toString() 5](#_Toc124366226)

[4. Enum name() 5](#_Toc124366227)

[5. Java Enum valueOf() 5](#_Toc124366228)

[6. Enum values() 5](#_Toc124366229)

[Java enum Constructor 6](#_Toc124366230)

# Enums

In Java, an enum (short for enumeration) is a type that has a fixed set of constant values. We use the enum keyword to declare enums. For example,

enum Size {

SMALL, MEDIUM, LARGE, EXTRALARGE

}

Here, we have created an enum named Size. It contains fixed values SMALL, MEDIUM, LARGE, and EXTRALARGE

These values inside the braces are called enum constants (values).

enum Size {

SMALL, MEDIUM, LARGE, EXTRALARGE

}

class Main {

public static void main(String[] args) {

System.out.println(Size.SMALL);

System.out.println(Size.MEDIUM);

}

}

Output

SMALL

MEDIUM

enum Size {

SMALL, MEDIUM, LARGE, EXTRALARGE

}

class Test {

Size pizzaSize;

public Test(Size pizzaSize) {

this.pizzaSize = pizzaSize;

}

public void orderPizza() {

switch(pizzaSize) {

case SMALL:

System.out.println("I ordered a small size pizza.");

break;

case MEDIUM:

System.out.println("I ordered a medium size pizza.");

break;

default:

System.out.println("I don't know which one to order.");

break;

}

}

}

class Main {

public static void main(String[] args) {

Test t1 = new Test(Size.MEDIUM);

t1.orderPizza();

}

}

All enums implicitly extend java.lang.Enum. Since Java does not support multiple [inheritance](https://crunchify.com/understanding-java-annotation-annotation-examples/), an enum cannot extend anything else.

Enum constants are implicitly static and [final](https://crunchify.com/in-java-how-to-perform-file-search-operation-using-java-nio-file-interface-tutorial-on-file-and-directory-operations/) and can not be changed once created.

You can not create instance of enums by using new operator in Java because constructor of Enum in [Java](https://crunchify.com/category/java-tutorials/) 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](https://crunchify.com/missing-maven-settings-xml-file-for-your-eclipse-what-if-you-need-two-settings-xml-file-for-work-personal-workspace/) 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](https://crunchify.com/what-is-an-abstract-class-and-abstract-method-in-java-when-should-i-use-it/), protected , or private

Enums can contain [constructors](https://crunchify.com/create-simple-pojo-and-multiple-java-reflection-examples/), methods, variables, and constant class bodies.

enum constructors can NEVER be invoked directly in code. They are always called [automatically](https://crunchify.com/how-to-use-expiringmap-maven-java-utility-to-remove-expired-objects-from-map-automatically-complete-java-tutorial/) when an enum is initialized.

# Enum Class in Java

In Java, enum types are considered to be a special type of class

An enum class can include methods and fields just like regular classes.

enum Size {

constant1, constant2, …, constantN;

// methods and fields

}

When we create an enum class, the compiler will create instances (objects) of each enum constants. Also, all enum constant is always public static final by default.

enum Size{

SMALL, MEDIUM, LARGE, EXTRALARGE;

public String getSize() {

// this will refer to the object SMALL

switch(this) {

case SMALL:

return "small";

case MEDIUM:

return "medium";

case LARGE:

return "large";

case EXTRALARGE:

return "extra large";

default:

return null;

}

}

public static void main(String[] args) {

// call getSize()

// using the object SMALL

System.out.println("The size of the pizza is " + Size.SMALL.getSize());

}

}

Since Size is an enum class, the compiler automatically creates instances for each enum constants.

Here inside the main() method, we have used the instance SMALL to call the getSize() method

## Methods of Java Enum Class

There are some predefined methods in enum classes that are readily available for use.

### 1. Java Enum ordinal()

The ordinal() method returns the position of an enum constant. For example,

ordinal(SMALL)

// returns 0

### 2. Enum compareTo()

The compareTo() method compares the enum constants based on their ordinal value. For example,

Size.SMALL.compareTo(Size.MEDIUM)

// returns ordinal(SMALL) - ordinal(MEDIUM)

### 3. Enum toString()

The toString() method returns the string representation of the enum constants. For example,

SMALL.toString()

// returns "SMALL"

### 4. Enum name()

The name() method returns the defined name of an enum constant in string form. The returned value from the name() method is final. For example,

name(SMALL)

// returns "SMALL"

### 5. Java Enum valueOf()

The valueOf() method takes a string and returns an enum constant having the same string name. For example,

Size.valueOf("SMALL")

// returns constant SMALL.

### 6. Enum values()

The values() method returns an array of enum type containing all the enum constants. For example,

Size[] enumArray = Size.value();

The main difference between name() and toString() is that name() is a final method, so it cannot be overridden. The toString() method returns the same value that name() does by default, but toString() can be overridden by subclasses of Enum.

public enum WeekDay {

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY;

public String toString() {

return name().charAt(0) + name().substring(1).toLowerCase();

}

}

## Java enum Constructor

In Java, an enum class may include a constructor like a regular class. These enum constructors are either

* private - accessible within the class  
  or
* package-private - accessible within the package

enum Size {

// enum constants calling the enum constructors

SMALL("The size is small."),

MEDIUM("The size is medium."),

LARGE("The size is large."),

EXTRALARGE("The size is extra large.");

private final String pizzaSize;

// private enum constructor

private Size(String pizzaSize) {

this.pizzaSize = pizzaSize;

}

public String getSize() {

return pizzaSize;

}

}

class Main {

public static void main(String[] args) {

Size size = Size.SMALL;

System.out.println(size.getSize());

}

}

Since the constructor is private, we cannot access it from outside the class. However, we can use enum constants to call the constructor.

In the Main class, we assigned SMALL to an enum variable size. The constant SMALL then calls the constructor Size with string as an argument.