

[Home](#) [Java](#) [Programs](#) [OOps](#) [String](#) [Exception](#) [Multithreading](#) [Collections](#)

Java Enums

The **Enum in Java** is a data type which contains a fixed set of constants.

It can be used for days of the week (SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, and SATURDAY), directions (NORTH, SOUTH, EAST, and WEST), season (SPRING, SUMMER, WINTER, and AUTUMN or FALL), colors (RED, YELLOW, BLUE, GREEN, WHITE, and BLACK) etc. According to the Java naming conventions, we should have all constants in capital letters. So, we have enum constants in capital letters.

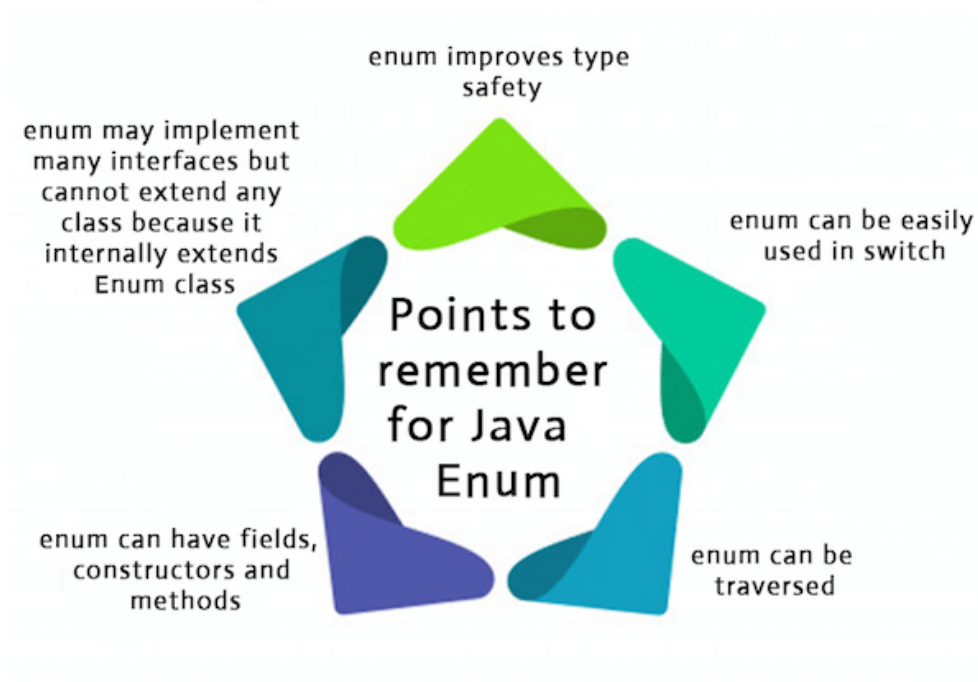
Java Enums can be thought of as classes which have a fixed set of constants (a variable that does not change). The Java enum constants are static and final implicitly. It is available since JDK 1.5.

Enums are used to create our own data type like classes. The **enum** data type (also known as Enumerated Data Type) is used to define an enum in Java. Unlike C/C++, enum in Java is more *powerful*. Here, we can define an enum either inside the class or outside the class.

Java Enum internally inherits the *Enum class*, so it cannot inherit any other class, but it can implement many interfaces. We can have fields, constructors, methods, and main methods in Java enum.

Points to remember for Java Enum

- Enum improves type safety
- Enum can be easily used in switch
- Enum can be traversed
- Enum can have fields, constructors and methods
- Enum may implement many interfaces but cannot extend any class because it internally extends Enum class



Simple Example of Java Enum

```
class EnumExample1{  
    //defining the enum inside the class  
    public enum Season { WINTER, SPRING, SUMMER, FALL }  
    //main method  
    public static void main(String[] args) {  
        //traversing the enum  
        for (Season s : Season.values())  
            System.out.println(s);  
    }  
}
```

Test it Now

Output:

```
WINTER  
SPRING  
SUMMER  
FALL
```

Let us see another example of Java enum where we are using value(), valueOf(), and ordinal() methods of Java enum.

```
class EnumExample1{  
    //defining enum within class  
    public enum Season { WINTER, SPRING, SUMMER, FALL }  
}
```

↑ SCROLL TO TOP [method](#)

```
public static void main(String[] args) {  
    //printing all enum  
    for (Season s : Season.values()){  
        System.out.println(s);  
    }  
    System.out.println("Value of WINTER is: "+Season.valueOf("WINTER"));  
    System.out.println("Index of WINTER is: "+Season.valueOf("WINTER").ordinal());  
    System.out.println("Index of SUMMER is: "+Season.valueOf("SUMMER").ordinal());  
}
```

Output:

```
WINTER  
SPRING  
SUMMER  
FALL  
Value of WINTER is: WINTER  
Index of WINTER is: 0  
Index of SUMMER is: 2
```

Note: Java compiler internally adds values(), valueOf() and ordinal() methods within the enum at compile time. It internally creates a static and final class for the enum.

What is the purpose of the values() method in the enum?

The Java compiler internally adds the values() method when it creates an enum. The values() method returns an array containing all the values of the enum.

What is the purpose of the valueOf() method in the enum?

The Java compiler internally adds the valueOf() method when it creates an enum. The valueOf() method returns the value of given constant enum.

What is the purpose of the ordinal() method in the enum?

The Java compiler internally adds the ordinal() method when it creates an enum. The ordinal() method returns the index of the enum value.

Defining Java Enum

The enum can be defined within or outside the class because it is similar to a class. The semicolon (;) at the end of the enum constants are optional. For example:

```
enum Season { WINTER, SPRING, SUMMER, FALL }
```

Or,

```
enum Season { WINTER, SPRING, SUMMER, FALL; }
```

Both the definitions of Java enum are the same.

Java Enum Example: Defined outside class

```
enum Season { WINTER, SPRING, SUMMER, FALL }  
class EnumExample2{  
    public static void main(String[] args) {  
        Season s=Season.WINTER;  
        System.out.println(s);  
    }  
}
```

Test it Now

Output:

```
WINTER
```

Java Enum Example: Defined inside class

```
class EnumExample3{  
    enum Season { WINTER, SPRING, SUMMER, FALL; }//semicolon(;) is optional here  
    public static void main(String[] args) {  
        Season s=Season.WINTER;//enum type is required to access WINTER
```

↑ SCROLL TO TOP);

```
}}
```

Test it Now

Output:

```
WINTER
```

Java Enum Example: main method inside Enum

If you put main() method inside the enum, you can run the enum directly.

```
enum Season {  
    WINTER, SPRING, SUMMER, FALL;  
    public static void main(String[] args) {  
        Season s=Season.WINTER;  
        System.out.println(s);  
    }  
}
```

Output:

```
WINTER
```

Initializing specific values to the enum constants

The enum constants have an initial value which starts from 0, 1, 2, 3, and so on. But, we can initialize the specific value to the enum constants by defining fields and constructors. As specified earlier, Enum can have fields, constructors, and methods.

```

class EnumExample4{
enum Season{
WINTER(5), SPRING(10), SUMMER(15), FALL(20);

private int value;
private Season(int value){
this.value=value;
}
}

public static void main(String args[]){
for (Season s : Season.values())
System.out.println(s+ " "+s.value);

}}

```

Test it Now

Output:

```

WINTER 5
SPRING 10
SUMMER 15
FALL 20

```

Constructor of enum type is private. If you don't declare private compiler internally creates private constructor.

```

enum Season{
WINTER(10),SUMMER(20);
private int value;
Season(int value){
this.value=value;
}
}

```

Internal code generated by the compiler for the above example of enum type

```

final class Season extends Enum
{
public static Season[] values()
{

```

↑ SCROLL TO TOP 1[])\$VALUES.clone();

```

    }
    public static Season valueOf(String s)
    {
        return (Season)Enum.valueOf(Season, s);
    }
    private Season(String s, int i, int j)
    {
        super(s, i);
        value = j;
    }
    public static final Season WINTER;
    public static final Season SUMMER;
    private int value;
    private static final Season $VALUES[];
    static
    {
        WINTER = new Season("WINTER", 0, 10);
        SUMMER = new Season("SUMMER", 1, 20);
        $VALUES = (new Season[] {
            WINTER, SUMMER
        });
    }
}

```

Can we create the instance of Enum by new keyword?

No, because it contains private constructors only.

Can we have an abstract method in the Enum?

Yes, Of course! we can have abstract methods and can provide the implementation of these methods.

Java Enum in a switch statement

We can apply enum on switch statement as in the given example:

Example of applying Enum on a switch statement

```

class EnumExample5{
    enum Day{ SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY}
    public static void main(String args[]){
        Day day=Day.MONDAY;

        switch(day){
            case SUNDAY:

```

↑ SCROLL TO TOP


```
System.out.println("sunday");  
break;  
case MONDAY:  
System.out.println("monday");  
break;  
default:  
System.out.println("other day");  
}  
}}
```

Test it Now

Output:

monday

← Prev

Next →

 **For Videos Join Our Youtube Channel: [Join Now](#)**

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



↑ SCROLL TO TOP

Learn Latest Tutorials



Splunk tutorial
Splunk



SPSS tutorial
SPSS



Swagger tutorial
Swagger



T-SQL tutorial
Transact-SQL



Tumblr tutorial
Tumblr



React tutorial
ReactJS



Regex tutorial
Regex



Reinforcement learning tutorial
Reinforcement Learning



R Programming tutorial
R Programming



RxJS tutorial
RxJS



React Native tutorial
React Native



Python Design Patterns
Python Design Patterns



Python Pillow tutorial
Python Pillow




Python Turtle tutorial
Python Turtle




Keras tutorial
Keras


Preparation




Aptitude
Aptitude




Logical Reasoning
Reasoning



Verbal Ability
Verbal Ability




Interview Questions
Interview Questions




Company Interview Questions
Company Questions


Trending Technologies




Artificial Intelligence Tutorial
Artificial




AWS Tutorial
AWS



Selenium tutorial
Selenium










Cloud Computing tutorial
Cloud Computing



Hadoop tutorial
Hadoop

↑ SCROLL TO TOP

| | | | | |
|--|---|---|---|---|
|  ReactJS Tutorial ReactJS |  Data Science Tutorial Data Science |  Angular 7 Tutorial Angular 7 |  Blockchain Tutorial Blockchain |  Git Tutorial Git |
|  Machine Learning Tutorial Machine Learning |  DevOps Tutorial DevOps | | | |

B.Tech / MCA

| | | | | |
|--|--|--|--|--|
|  DBMS tutorial DBMS |  Data Structures tutorial Data Structures |  DAA tutorial DAA |  Operating System tutorial Operating System |  Computer Network tutorial Computer Network |
|  Compiler Design tutorial Compiler Design |  Computer Organization and Architecture Computer Organization |  Discrete Mathematics Tutorial Discrete Mathematics |  Ethical Hacking Tutorial Ethical Hacking |  Computer Graphics Tutorial Computer Graphics |
|  Software Engineering Tutorial Software Engineering |  html tutorial Web Technology |  Cyber Security tutorial Cyber Security |  Automata Tutorial Automata |  C Language tutorial C Programming |
|  C++ tutorial C++ |  Java tutorial Java |  .Net Framework tutorial .Net |  Python tutorial Python |  List of Programs Programs |
|  Control Systems tutorial Control System |  Data Mining Tutorial Data Mining |  Data Warehouse Tutorial Data Warehouse | | |

