Java Enum

**Enum in java** is a data type that contains 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) etc. The java enum constants are static and final implicitly. It is available from JDK 1.5.

Java Enums can be thought of as classes that have fixed set of constants.

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

1. **class** EnumExample1{
2. **public** **enum** Season { WINTER, SPRING, SUMMER, FALL }
4. **public** **static** **void** main(String[] args) {
5. **for** (Season s : Season.values())
6. System.out.println(s);
8. }}

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=EnumExample1)

Output:WINTER

SPRING

SUMMER

FALL

What is the purpose of values() method in 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.

Defining Java enum

The enum can be defined within or outside the class because it is similar to a class.

Java enum example: defined outside class

1. **enum** Season { WINTER, SPRING, SUMMER, FALL }
2. **class** EnumExample2{
3. **public** **static** **void** main(String[] args) {
4. Season s=Season.WINTER;
5. System.out.println(s);
6. }}

Output:WINTER

Java enum example: defined inside class

1. **class** EnumExample3{
2. **enum** Season { WINTER, SPRING, SUMMER, FALL; }//semicolon(;) is optional here
3. **public** **static** **void** main(String[] args) {
4. Season s=Season.WINTER;//enum type is required to access WINTER
5. System.out.println(s);
6. }}

Output:WINTER

Initializing specific values to the enum constants

The enum constants have initial value that 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.

Example of specifying initial value to the enum constants

1. **class** EnumExample4{
2. **enum** Season{
3. WINTER(5), SPRING(10), SUMMER(15), FALL(20);
5. **private** **int** value;
6. **private** Season(**int** value){
7. **this**.value=value;
8. }
9. }
10. **public** **static** **void** main(String args[]){
11. **for** (Season s : Season.values())
12. System.out.println(s+" "+s.value);
14. }}

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.**

Can we create the instance of enum by new keyword?

|  |
| --- |
| No, because it contains private constructors only. |

Can we have abstract method in enum?

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

Java enum in switch statement

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

Example of applying enum on switch statement

1. **class** EnumExample5{
2. **enum** Day{ SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY}
3. **public** **static** **void** main(String args[]){
4. Day day=Day.MONDAY;
6. **switch**(day){
7. **case** SUNDAY:
8. System.out.println("sunday");
9. **break**;
10. **case** MONDAY:
11. System.out.println("monday");
12. **break**;
13. **default**:
14. System.out.println("other day");
15. }
16. }}