**Java5: enum type**

**Introduction**

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain..

The keyword ‘enum’ is used to declare new enumeration types in Java mostly Like in C and C++. An enum type is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.

**Why enum?**

Enumerations serve the purpose of representing a group of named constants in a programming language. Examples include natural enumerated types like Season names,the planets,month names, days of the week, colors, directions, etc.

enums are used when we know all possible values at compile-time, such as choices on a menu, rounding modes, command-line flags, etc. It is not necessary that the set of constants in an enum type stay fixed for all time.

In Java (from 1.5), enums are represented using enum data type. Java enums are more powerful than [C/C++ enums](https://www.geeksforgeeks.org/enumeration-enum-c/). In Java, we can also add variables, methods, and constructors to it. The main objective of enum is to define our own data types (Enumerated Data Types).

**What is Enumeration in Java?**

**enum**, introduced in Java 5, is a special data type that consists of a set of pre-defined named values separated by commas. These **named values** are also known as **elements** or **enumerators** or **enum instances**. Since the values in the enum type are constant, you should always represent them in UPPERCASE letters as a convention (because **final** internally).

You can use an **enum** type when you need a fixed set of pre-defined constant values that are known at the compile-time itself. Example can be ***Seasons of a year*.**

**Why Do We Need Enumeration?**

Java supports two types of Data Types:

1. Inbuilt Data Types- int, float, double, char, etc.
2. User-Defined Datatypes

Sometimes inbuilt data types are not sufficient. Let’s suppose that we have data of different data types required to be stored in a single variable. In such a situation, inbuilt data types won’t fulfill the need. That’s why there is a requirement for user-defined Data Types, and enum in Java is one of them.

**Java enum Strings**

Before you learn about enum strings, make sure to know about [Java enum](https://www.programiz.com/java-programming/enums).

In Java, we can get the string representation of enum constants using the toString() method or the name() method. We can change the default string representation of enum constants by overriding the toString() method.

**Characteristics make enum a ‘special’ class!**

* enum constants cannot be overridden
* enum doesn’t support the creation of objects
* enum can’t extend other classes

**Syntax**

enum enumerated\_type\_name  
 {  
 VALUE\_1, VALUE\_2, VALUE\_3, … [ ; ]

}

Note: enum can be defined only inside a top-level class or interface or in a static context. It should not be inside a method.

**enum Class**

One thing you must have noticed in the above example (see comments) is that we did not use a **“new”** keyword while instantiating. Although the enumerations define a class **Enum** and the constructors as well, they do not use a new keyword.

We can use the enumerators defined in the same way as the primitive. Enum is a generic class and all the enumerations inherit Enum.

**Syntax:**

**class enum< E extends Enum<E> >** // E is the enumeration type.

**How to use enum in Java**

enum can be declared inside or outside (enum keyword example) a class but not inside a method.

**Iterating enum through Loops**

Here, we will discuss how we can loop through an enum. We have declared enumeration with four enumerators (outside a class). Then we have initialized a for each loop (inside a class) and tried to fetch the value of the enumerators.

**In if-else**

In the below program, we have created an enumeration with three different enumerators and then stored the enumerator in the reference variable for one of the specified enumerators. Then we have started **if** condition check where we have implemented two conditions specified with OR so that if one of those conditions is met then it will print the statement specified in the if condition. Otherwise, it will print the statement specified in the else condition.

**In switch Statement**

In the below program, we have created an enumeration with four enumerators. Then, we stored one of the enumerators in the reference variable. Thereafter, we initialized a Switch statement and checked each of these enumerators.

Upon the occurrence of that particular enumerator, it will print the statement specified in the particular case.

**enum Field**

We can add fields to the Java enum and each enumerator gets these fields. The field value should be assigned to the constructor of the enum.

This sums that Java enum has a constructor that takes an int. This constructor sets the int field. When an enumerator is defined, then that int value (specified for that enumerator) is passed to the constructor.

**enum Constructor**

Enum (as it is also a class) does support constructor to pass data during the creation of enumerators or also known as enum constants.

The main property of an enum constructor is that they are private or private-package. This means that either they are accessible within a class or within a package.

**Constructor overloading in enum**

Just like normal constructors you can also override the constructors of an enum. i.e. you can have the constructor with different parameters.

Example

Following Java program demonstrates the constructor overloading in enumerations.

import java.util.Arrays;

enum Student {

   Krishna("Krishna", "kasyap", "Bhagavatula"), Ravi("Ravi", "Kumar", "pyda"), Archana("Archana", "devraj", "mohonthy");

   private String firstName;

   private String lastName;

   private String middleName;

   private Student(String firstName, String lastName,String middlename){

      this.firstName = firstName;

      this.lastName = lastName;

      this.middleName = middleName;

   }

   private Student(String name) {

      this(name.split(" ")[0], name.split(" ")[1], name.split(" ")[2]);

   }

}

public class ConstructorOverloading{

   public static void main(String args[]) {

      Student stds[] = Student.values();

      System.out.println(Arrays.toString(stds));

   }

}

Output

[Krishna, Ravi, Archana]

**Points to Remember**

1. enums are implicitly **final** subclasses of java.lang.Enum class
2. if an enum is a member of a class, it’s implicitly **static**
3. **new** keyword can not be used to intialize an enum, even within the enum type itself
4. enum constants are implicitly **public static final**
5. you can use enum in switch statement like int or char primitive data type