Java Enum Tutorial with examples

BY CHAITANYA SINGH | FILED UNDER: JAVA TUTORIALS

An enum is a special type of data type which is basically a collection (set) of constants. In this tutorial we will learn how to use enums in Java and what are the possible scenarios where we can use them.

This is how we define Enum

```
public enum Directions{
  EAST,
  WEST,
  NORTH,
  SOUTH
}
```

Here we have a variable **Directions** of **enum type**, which is a collection of four constants EAST, WEST, NORTH and SOUTH.

How to assign value to a enum type?

```
Directions dir = Directions.NORTH;
```

The variable dir is of type Directions (that is a enum type). This variable can take any value, out of the possible four values (EAST, WEST, NORTH, SOUTH). In this case it is set to NORTH.

Use of Enum types in if-else statements

This is how we can use an enum variable in a if-else logic.

```
/* You can assign any value here out of
 * EAST, WEST, NORTH, SOUTH. Just for the
 * sake of example, I'm assigning to NORTH
 */
Directions dir = Directions.NORTH;

if(dir == Directions.EAST) {
  // Do something. Write your logic
} else if(dir == Directions.WEST) {
     // Do something else
  } else if(dir == Directions.NORTH) {
     // Do something
    } else {
        /* Do Something. Write logic for
         * the remaining constant SOUTH
         */
      }
```

Enum Example

This is just an example to demonstrate the use enums. If you understand the core part and basics, you would be able to write your own logic based on the requirement.

```
public enum Directions{
   EAST,
   WEST,
   NORTH,
   SOUTH
}
public class EnumDemo
{
   public static void main(String args[]){
 Directions dir = Directions.NORTH;
 if(dir == Directions.EAST) {
     System.out.println("Direction: East");
 } else if(dir == Directions.WEST) {
     System.out.println("Direction: West");
   } else if(dir == Directions.NORTH) {
       System.out.println("Direction: North");
       } else {
  System.out.println("Direction: South");
       }
   }
}
```

Output:

```
Direction: North
```

Use of Enum in Switch-Case Statements

Here is the example to demonstrate the use of enums in switch-case statements.

```
public enum Directions{
   EAST,
   WEST,
   NORTH,
   SOUTH
}
public class EnumDemo
{
   Directions dir;
   public EnumDemo(Directions dir) {
      this.dir = dir;
   }
   public void getMyDirection() {
     switch (dir) {
       case EAST:
          System.out.println("In East Direction");
          break;

       case WEST:
          System.out.println("In West Direction");
          break;

       case NORTH:
          System.out.println("In North Direction");
          break;

       default:
          System.out.println("In South Direction");
          break;
     }
   }

    public static void main(String[] args) {
        EnumDemo obj1 = new EnumDemo(Directions.EAST);
        obj1.getMyDirection();
        EnumDemo obj2 = new EnumDemo(Directions.SOUTH);
        obj2.getMyDirection();
    }
}
```

Output:

```
In East Direction
In South Direction
```

How to iterate through an Enum variable

```
class EnumDemo
{
    public static void main(String[] args) {
     for (Directions dir : Directions.values()) {
         System.out.println(dir);
     }
    }
}
```

This code would display all the four constants.

Enum Fields and Methods

Lets take an example first then we will discuss it in detail:

```
public enum Directions{
  EAST ("E"),
  WEST ("W"),
  NORTH ("N"),
  SOUTH ("S")
  ;
  /* Important Note: Must have semicolon at
   * the end when there is a enum field or method
   */
  private final String shortCode;

  Directions(String code) {
      this.shortCode = code;
  }

  public String getDirectionCode() {
      return this.shortCode;
  }
}
public class EnumDemo
{
    public static void main(String[] args) {
     Directions dir = Directions.SOUTH;
     System.out.println(dir.getDirectionCode());
     Directions dir2 = Directions.EAST;
     System.out.println(dir2.getDirectionCode());
    }
}
```

Output:

```
S
E
```

As you can see in this example we have a field `shortCode` for each of the constant, along with a method `getDirectionCode()` which is basically a getter method for this field. When we define a constant like this `EAST ("E")`, it calls the enum constructor (Refer the constructor `Directions` in the above example) with the passed argument. This way the passed value is set as an value for the field of the corresponding enum's constant [EAST("E") **=>** Would call constructor Directions("E") **=>** this.shortCode = code **=>** this.shortCode = "E" **=>** shortCode field of constant EAST is set to "E"].

Important points to Note:

1) While defining Enums, the constants should be declared first, prior to any fields or methods.
2) When there are fields and methods declared inside Enum, the list of enum constants must end with a semicolon(;).

Enjoyed this post? Try these related posts

1. **Java Annotations tutorial with examples**
2. **Sorting float array in Java example**
3. **Sort an Array in Descending (Reverse) Order – Java**
4. **Java Regular Expressions (java regex) Tutorial with examples**
5. **Java Arrays**
6. **Java Autoboxing and Unboxing with examples**

Comments

zsee says
APRIL 26, 2015 AT 7:25 AM

short and to the point. Loving it…

Reply

Luqman says
SEPTEMBER 7, 2015 AT 11:43 AM

thanks a lot

Reply

Ram says
MAY 30, 2016 AT 10:02 AM

if we are using public declaration for the enum Datatype. we must put that inside our class..

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

POST COMMENT