enum

enum is a keyword like class, which is used to create one new Data Type with possible set of values.

Notes:
#1 enum is a keyword (JDK1.5)
#2 enum is a set of possible values
#3 enum is used to create one DataType
#4 syntax is :

```
  enum  <enum_name> {
    Possible values in all upper case;
  }
```

#5 Every value in enum by default "public static final"
#6 value itself a variable

examples:

```
enum Gender {
     MALE,FEMALE ;
}
enum TimeMode{
     AM,PM,NOON;
}
enum ExamResult { PASS,FAIL,ABSENT; }

enum StudentGrade{ A,B,C; }

enum TicketStatus {
     CNF,RAC,WL,PQWL,GWL;
}
enum CricketType{
     BAT,BOWL,WK,AL;
}
enum BakingModes{
     NET,MOBILE,TELE;
}
enum UserRoles{
     ADMIN,EMPLOYEE,CUSTOMER;
}
enum IndCDNotes{
	C10,C20,C50,C100,C200,C500,C2000;
}
enum AcceptedResults{
	YES,NO,MAYBE;
}
```

#7 To get all possible values in enum use method "values()" that returns same enum type array, use for-each loop to print them.

#8 Every enum value will be identified using unique index that is called as "ordinal". We can get this using "ordinal()" method. Starts from zero.

-----------Example---------

```
//creating enum
package com.app;

enum Grade{ A,B,C; }

class Test {
	public static void main(String[] srs) {
		//reading one enum value "enum.value"
		System.out.println(Grade.A);
		//reading all enum values...
		Grade[] pgs=Grade.values();

		//finding no.of values in enum
		System.out.println(pgs.length);
		//display one by one with index.
		for(Grade pg:pgs){
			System.out.println(pg + "," + pg.ordinal());
		}
	}
}
```

#9 We can use static import, so that exact variable can be accessed from enum without using Enum.Variable format (Use directly eum variable only)
ex:

```
package com.app;
```

```
import static java.lang.annotation.ElementType.FIELD;
class Test {
     public static void main(String[] args) {
          System.out.println(FIELD);
     }
}
```

** Here ElementType is a enum.

Annotation

Annotations are Tags given to java code that provides information to either Programmer or Pre-defined Program(Compiler, JVM, Framework, Container etc...)

a. These are introduced in JDK 1.5
b. To create annotation we need to provide
  1.Name of annotation
  2.Target= Where it is applicable in code
  3.Retention=When Annotation should work
c. Before Annotations concept we were using Marker interfaces or XML coding
d. Annotation reduces programmer work (coding/configuration/code check)
e. Target possible values are provided using "ElementType" enum.

```
enum ElementType {
      TYPE, FIELD, METHOD,PARAMETER,CONSTRUCTOR,
      LOCAL_VARIABLE,ANNOTATION_TYPE,PACKAGE
}
```

Type= class/interface levels
Field=Instance/static variabl levels
Method=Method level
ANNOTATION_TYPE=Another annotation level

f. Retention possible values are provided using "RetentionPolicy" enum

```
enum RetentionPolicy {
      SOURCE,CLASS,RUNTIME
}
```

g. Here Retention and Target are Annotations used to create new Annotation such kind annotations are called meta annotations.

//1. create Annotation(creator)(f/w)
//2. Define Processor class(creator)(f/w)

//3. Use Annotation(Programmer/User)

-------------Example code-----------

```
package com.app;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.ElementType.METHOD;
import static
java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Retention;
import java.lang.annotation.Target;

//1.creating one annotation
@Target({TYPE,METHOD})
@Retention(RUNTIME)
@interface Product{

}

//2. processing annotation
class ProductProcess{
	public static void process(Class<?> c){
		Product
p=(Product)c.getAnnotation(Product.class);
		if(p==null)
		throw new RuntimeException("No Product Annotation
provided");
		else
		System.out.println("Your class having Product
Annotation");
	}
}
```

```
//3. using annotation
@Product
class Sample{  }

//4. Testing Annotation
public class Test{
     public static void main(String[] args) {
          ProductProcess.process(Sample.class);
     }
}
```

Arguments of Annotations:-

To Provide data to Annotation Processor class we use Arguments. Arguments can be optional or required. If Argument has default value then it is optional else it is required.

Syntyax:

```
@interface Annotation_Name{
 //optional attribute
 DataType methodName() default value;

 //required attirbute
 DataType methodName() ;
}
```

----------Example code----------

```
package com.app;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;


enum ObjType{
     NEW,USED,INTO;
}

//1. creating annotation
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@interface Product{
     //attributes
     int    prodId(); //required
     String objName() default "NO";//optional
     ObjType  objType() default ObjType.NEW;//optional
}

//2. Processor class for Annotation
class ProductProcessor{
     public static void process(Class<?> c){
          Product 
p=(Product)c.getAnnotation(Product.class);
          if(p==null)
               throw new RuntimeException("No Product 
Annotation is provided");
          else{
               System.out.println("Having Product 
Annotation:");
               System.out.println("Id:  " + p.prodId());
               System.out.println("NAME:" + p.objName());
               System.out.println("Type:" + p.objType());
          }
     }
}

//3. Using annotation
```

```
@Product(prodId=6,objName="SM",objType=ObjType.USED)
class Smaple{

}

//4.Testing Annotation
public class Test{
	public static void main(String[] args) {
		ProductProcessor.process(Smaple.class);
	}
}
```

FB Group: https://www.facebook.com/groups/thejavatemple/

Email : javabyraghu@gmail.com