Enumerations

Enumerations was added to Java language in JDK5. **Enumeration** means a list of named constant. In Java, enumeration defines a class type. An Enumeration can have constructors, methods and instance variables. It is created using **enum** keyword. Each enumeration constant is *public*, *static* and *final* by default. Even though enumeration defines a class type and have constructors, you do not instantiate an **enum** using **new**. Enumeration variables are used and declared in much a same way as you do a primitive variable.

How to Define and Use an Enumeration

1. An enumeration can be defined simply by creating a list of enum variable. Let us take an example for list of Subject variable, with different subjects in the list.

```
enum Subject
{
 Java, Cpp, C, Dbms
}
```

2. Identifiers Java, Cpp, C and Dbms are called **enumeration constants**. These are public, static and final by default.
3. Variables of Enumeration can be defined directly without any **new** keyword.

```
Subject sub;
```

4. Variables of Enumeration type can have only enumeration constants as value. We define an enum variable as enum_variable = enum_type.enum_constant;

```
sub = Subject.Java;
```

5. Two enumeration constants can be compared for equality by using the **= =** relational operator.
Example:

```
if(sub == Subject.Java) {
   ...
}
```

Example of Enumeration

```
enum WeekDays
{ sun, mon, tues, wed, thurs, fri, sat }

class Test
{
 public static void main(String args[])
 {
  WeekDays wk;
  wk = WeekDays.sun;
  System.out.println("Today is "+wk);
 }
}
```

Today is sun

Example of Enumeration using switch statement

```
enum Restaurants {
dominos, kfc, pizzahut, paninos, burgerking
}
class Test {
public static void main(String args[])
{
Restaurants r;
r = Restaurants.paninos;
switch(r) {
type name i.e only r, not Restaurants.r
case dominos:
System.out.println("I AM " + r.dominos);
break;
case kfc:
System.out.println("I AM " + r.kfc);
break;
case pizzahut:
System.out.println("I AM " + r.pizzahut);
break;
case paninos:
System.out.println("I AM " + r.paninos);
break;
case burgerking:
System.out.println("I AM " + r.burgerking);
break;
}
}
}
```

I AM paninos

Values( ) and ValueOf( ) method

All the enumerations predefined methods **values()** and **valueOf()**. `values()` method returns an array of enum-type containing all the enumeration constants in it. Its general form is,

```
public static enum-type[ ] values()
```

`valueOf()` method is used to return the enumeration constant whose value is equal to the string passed in as argument while calling this method. It's general form is,

```
public static enum-type valueOf (String str)
```

Example of enumeration using values() and valueOf() methods:

```
enum Restaurants {
dominos, kfc, pizzahut, paninos, burgerking
}
class Test {
public static void main(String args[])
{
Restaurants r;
System.out.println("All constants of enum type Restaurants are:");
Restaurants  rArray[] = Restaurants.values();
for(Restaurants  a : rArray)
System.out.println(a);

r = Restaurants.valueOf("dominos");
System.out.println("I AM " + r);
}
}
```

All constants of enum type Restaurants are: dominos kfc pizzahut paninos burgerking I AM dominos

Points to remember about Enumerations

1. Enumerations are of class type, and have all the capabilities that a Java class has.
2. Enumerations can have Constructors, instance Variables, methods and can even implement Interfaces.
3. Enumerations are not instantiated using **new** keyword.
4. All Enumerations by default inherit **java.lang.Enum** class.

Enumeration with Constructor, instance variable and Method

```
enum Student
{
 John(11), Bella(10), Sam(13), Viraaj(9);
 private int age;
 int getage { return age; }
 public Student(int age)
 {
  this.age= age;
 }
}

class EnumDemo
{
 public static void main( String args[] )
 {
  Student S;
  System.out.println("Age of Viraaj is " +Student.Viraaj.getage()+ "years");
 }
}
```

Age of Viraaj is 9 years

In this example as soon as we declare an enum variable(*Student S*), the constructor is called once, and it initializes age for every enumeration constant with values specified with them in parenthesis.