Enums

An enumeration is a list of named constants.

In Java, an enumeration defines a class type.

By making enumerations into classes, the capabilities of the enumeration are greatly expanded.

An enumeration is created using the enum keyword.

Enum declaration can be done outside a Class or inside a Class but not inside a Method

We can declare main() method inside enum. Hence we can invoke enum directly from the Command Prompt.

```
/* internally above enum Color is converted to (Check Example.java)
class Color
{
        public static final Color Red = new Color();
        public static final Color Blue = new Color();
        public static final Color Green = new Color();
}*/
```

Enum and Inheritance :

-All enums implicitly extend java.lang.Enum class. As a class can only extend one parent in Java, so an enum cannot extend anything else.

-An enum cannot be a superclass.

-toString() method is overridden in java.lang.Enum class, which returns enum constant name.

-enum can implement many interfaces.

Two enumeration constants can be compared for equality by using the == relational operator.

values(), ordinal() and valueOf() methods :

These methods are present inside java.lang.Enum.

-values() method can be used to return all values present inside enum.

-Order is important in enums.By using ordinal() method, each enum constant index can be found, just like array index.

-valueOf() method returns the enum constant of the specified string value, if exists.

<u>enum and constructor :</u>

-enum can contain constructor and it is executed separately for each enum constant at the time of enum class loading.

-We can’t create enum objects explicitly and hence we can’t invoke enum constructor directly.

-And the constructor cannot be the public or protected it must have private or default modifiers.

-Why? if we create public or protected, it will allow initializing more than one objects.

-This is totally against enum concept.

enum and methods :

enum can contain concrete methods only i.e. no any abstract method.

You can compare for equality an enumeration constant with any other object by using equals( ), which overrides the equals( ) method defined by Object.

Although equals( ) can compare an enumeration constant to any other object, those two objects will be equal only if they both refer to the same constant,within the same enumeration.

Simply having ordinal values in common will not cause equals( ) to return true if the two constants are from different enumerations. Remember, you can compare two enumeration references for equality by using ==.