Enumeration (enum)

15/5/11

→ We can use enum to define a group of named constants

Ex!:① enum month
{
JAN, FEB, MAR ----- DEC(;) → optional.
}

② enum Bear
{
KF, KO, RC, FO(;) → optional
}

→ By using enum we can define our own data types

→ enum Concept introduced in 1.5v.

→ When compared with old languages enum Java's enum is more powerfull.

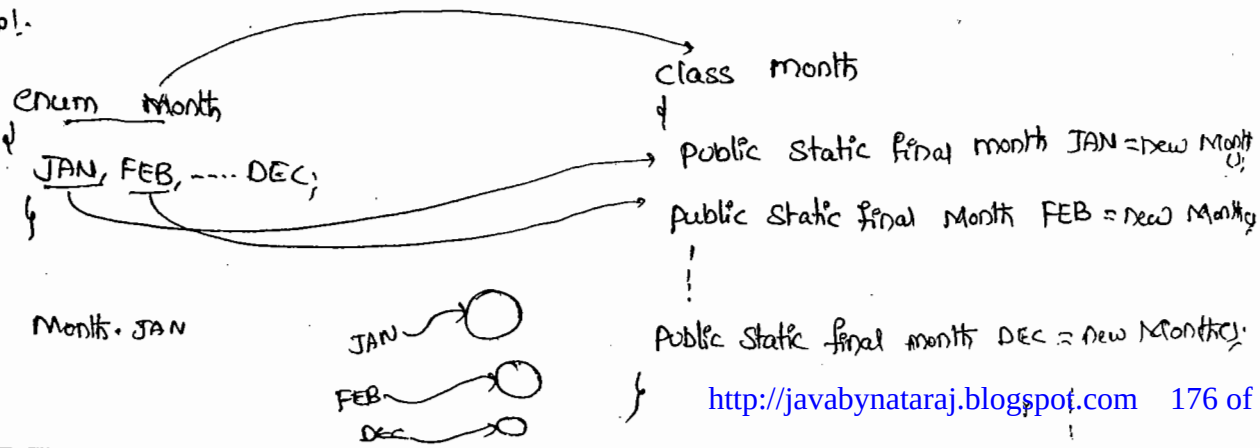
Internal implementation of enum :-

→ enum Concept internally implemented by using class Concept.

→ every enum constant is a reference variable to enum type object.

→ every enum constant is always public static final by default.

Eg!.



Declaration and usage of enum :-

Ex:-

```
enum Beer
{
 KF, KO, RC, FO;
}
Class Test
{
 P.S.V.m(String[] args)
 {
   Beer b1 = Beer.KF;
   S.o.pln(b1); // KF
 }
}
```

→ We can declare enum either with in the class or outside of the class but not inside a method.

→ If we are taying to declare enum with in a method we will get Compiletime Error.

Ex!-

```
enum X
{
}
Class Y
{
}
```
✓

```
Class Y
{
  enum X
  {
  }
}
```
✓

```
Class Y
{
  public void m1()
  {
    enum X
    {
    }
  }
}
```
✗

C.E:- enum types must not be local

→ if we declare enum outside the class the applicable modifiers are public, default, strictfp.

→ if we declare enum with in a class the applicable modifiers are public, default, strictfp, private, protected, static.

enum Vs Switch Statement :-

→ until 1.4v the allowed data-types for switch argument are byte, short, char, int.

→ But from 1.5v onwards in addition to above the corresponding wrapper classes (Byte, Short, Char, Integer,) + enum type also allowed

```
Switch (  )
{
}
```

1.4 V	1.5V	1.7V
byte short char int	Byte Short Character Integer enum	String

→ Hence from 1.5 version onwards we can use enum as argument to switch statement.

Ex:-

```
enum Beer
{
  KF, KO, RC, FO;
}
class Test
{
```

```
p.s.v.m(-->
{
 Beer b1 = Beer.Rc;
 switch (b1)
 {
  Case KF:
        S.o.pln(" It is children's brand);
        break;
  Case KO:
        S.o.pln(" It is too lite");
        break;
  Case Rc:
        S.o.pln(" It is challengers brand");
        break;
  Case FO:
        S.o.pln(" Buy one get one");
        break;
  default:
        S.o.pln(" other brands not recommended to
                                    take");
    }
  }
 }
```

Op:- It is challengers brand.

→ If we are passing enum type as argument to switch statement every case label should be a valid enum constant.

Eg:-
```
enum Beer
{
  KF, KO, RC, FO;
}
Beer b1 = Beer.KF;
Switch(b1)
{
  Case KF: ✓
  Case KO: ✓
  Case RC: ✓
  Case KALYANI: ✗ C.E:- UnQualified Enumeration Constant name required
}
}
```

enum Vs Inheritance :-

→ every enum in Java is direct child class of java.lang.Enum

→ As every enum is always extending java.lang.Enum there is no chance of extending any other enum (because Java won't provide support for multiple inheritance).

→ As every enum is always final implicitly we can't create child enum for our enums.

→ because of above reasons we can conclude inheritance concept is not applicable for enums Explicitly.

→ But enum can implement any no. of interfaces at a time.

Ex:- ①

```
enum x
{
}
enum y extends x
{
}
```

C.E:-

Cannot inherit from final x

enum types are not extensible

②

```
enum x extends java.lang.Enum
{
}
```

C.E:-

③

```
enum x
{
}
class y extends x
{
}
```

C.E1:- Cannot inherit from final x

C.E2:- enum types are not extensible

④

```
class x
{
}
enum y extends x
{
}
```

C.E:-

⑤

```
interface x
{
}
enum y implements x
{
}
```

java.lang.Enum :-

↦ every enum in java ~~should~~ is always direct child class of java.lang.Enum class.

↦ The power of enum is inheriting from this class only to our enum classes.

(Javap java.lang.Enum)

↦ It is an abstract class & direct child class of Object class.

↦ this class implements Comparable & Serializable Interfaces. hence every enum in java is bydefault Serializable and Comparable.

values() method :-

→ we can use values() method to list out all values of enum.

eg:- `Beer[] b = Beer.values();`

ordinal() method :-

→ with in the enum the order of constants is important we can specify its order by using ordinal value.

→ we can find ordinal value of enum constant by using ordinal method.

```
Public int ordinal();
```

→ Ordinal value is zero-based.

Eg:-

```
enum Beer
{
  KF, KO, RC, FO;
}
```

```
class Test
{
  p.s.v.m(String[] args)
  {
    Beer[] b = Beer.values();
    for(Beer b1 : b)
    {
      S.o.pln(b1+"----"+b1.ordinal());
    }
  }
}
```

o/p :-
KF ---- 0
KO ---- 1
RC ---- 2
FO ---- 3

Enum class Constructors & Speciality of Java enum :-

→ When Compared with old languages enum, Java enum is more Poweful because in addition to Constants we Can take Variables, methods, Constructors e.t.c... which may not possible in old languages. This extra facility is due to internal implementation of enum Concept which is class based.

→ Inside enum we Can declare main() method & hence we Can invoke enum class directly from Command prompt.

eg:- ① enum Fish
```
{
  STAR, GOLD, GUPPY, APOLLO, KILLER;   → mandatory.
  p.s.v.m(String[] args)
  {
    S.o.pln("ENUM MAIN METHOD");
  }
}
```

> Javac fish.java

> Java Fish

o/p!- Enum main method.

→ In addition to Constant if we want to take any extra members Compulsory List of Constants should be in the 1st Line & should ends with ";"

ex!- ① enum Color
{
RED, GREEN, BLUE;
Public void m1()
{
}
}

② enum Color
{
Public void m1()
{
}
RED, GREEN, BLUE;
}

③ enum Color
{
RED, GREEN, BLUE;
public void m1()
{
}
}

④ enum Color
{
Public void m1()
{
}
}

⑤ enum Color
{
}

→ Inside enum with out having Constant we can't to take any extra members, but Empty enum is always valid.

ex!.

enum Color
{
public void m1()
{
}
}

enum Color
{
}

Enum class Constructors :-

→ with-in Enum we can take Constructors also.

→ Enum class Constructors will be executed automatically at the time of Enum class loading. Hence because Enum Constants will be created at the time of class loading only.

→ We can't invoke Enum Constructors explicitly

Ex!-

```
enum Beer
{
  KF, KO, RC, FO;
  Beer()
  {
    S.o.pln("Constructor");
  }
}
Class Test
{
  Public static void main(String[] args)
  {
    Beer b1 = Beer.KF;
    S.o.pln(b1);
  }
}
```

o/p!-

```
Constructor
Constructor
Constructor
Constructor
KF
```

→ We Can't Create Objects of Enum explicitly & hence we Can't Call Constructors directly.

```
Beer b = new Beer();   X
```

C.E!-
Enum types may not be instantiated.

Ex:-
```
enum Beer
{
KF(75), KO(90), RC(90), FO;
int price;
Beer(int price)
{
this.price = price;
}
Beer()
{
this.price = 65;
}
Public int getprice()
{
return price;
}}
Class Test
{
p.s.v.m(String[] args)
{
Beer() b = Beer.values();
for(Beer b1 : b)
{
S.o.pln(b1 + "....." + b1.getprice());
}}}
```

KF ⇒ p.s.f. Beer KF = new Beer();

KF(100) ⇒ p.s.f. Beer KF = new Beer(100);

KF(100, "Gold", "Bitter")
⇒ p.s.f. Beer KF = new Beer(100, "Gold", "Bitter")

o/p!-
```
KF ..... 75
KO ..... 90
RC ..... 70
FO ..... 65
```

→ Within the enum we can take instance & static methods but we can't to take abstract methods

→ every enum constant represents an object hence what ever the methods we can apply on normal Java object we can apply those on enum constants also.

ex:-

Q) which of the following expressions are valid

✓ ① Beer.KF.equals(Beer.RC) // false

✓ ② Beer.KF.hashCode() // —

✓ ③ Beer.KF == Beer.RC ⟶ false

X ④ Beer.KF > Beer.RC

✓ ⑤ Beer.KF.ordinal > Beer.RC.ordinal

Case(1):-

```
Package pack1;
public enum Fish;
{
STAR, GUPPY, APOLLO;
}
```

```
Package pack2;
Class Test1
{
P.S.V.m(—)
{
S.o.pln(STAR);
}
}
```

```
import static pack1.Fish.STAR;
(or)
import static pack1.fish.*;
```

```
Package pack3;
--
Class Test2
{
 P.S.V.m(--)
 {
  Fish f = Fish.STAR;
  S.o.pln(f);
 }
}
```

```
import pack1.Fish;
    (or)
import pack1.*;
```

```
package pack4;
---
class Test3
{
 P.S.V.m(--)
 {
  Fish f = Fish.STAR;
  S.o.pln(Guppy);
 }
}
```

```
import pack1.Fish (or)
       import pack1.*;

import static pack1.Fish.Guppy;
          (or)
import static pack1.Fish.*;
```

Case2 :-

```
enum Color
{
  BLUE, RED
   {
     public void info()
     {
       S.o.pln(" Dangerous Color");
     }
   }, GREEN;

  public void info()
  {
   S.o.pln(" universal Color");
  }
}
```

```
class Test
{
  p.s.v.m(------>)
  {
    Color[] c = Color.values();
    for(Color c1 : c)
    {
      c1.Enfo();
    }
  }
}
```

O/P :- universal Color
Dangerous Color
universal Color.

Enum vs Enum vs Enumeration :-

Enum :-

→ It is a Keyword which can be used to define a group of named constants.

Enum :-

→ It is a class present in java.lang package which acts as a base class for all Java enums

Enumeration :-

→ It is an Interface present in java.util package, which can be used for retrieving objects from Collection one by one.