Enumeration(enum)

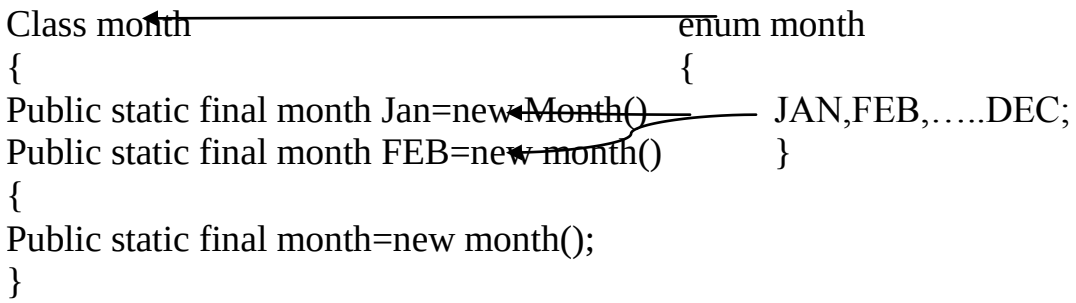
->We can use enum to define a group of named constants
Ex.

1) enum month
   {
   JAN,FEB,MAR,…….DEC;->optional
   }

2) enum Bear
   {
   KF,KO,RC,FO ;->optional
   }
   ->By using enum we can define our own datatype
   ->Enum concept introduced is 1.5v
   ->When compared with old language enum java's enum is more powerfull.

Internal Implementation of enum:-
->Enum concept internally implemented by using class concept.
->Every enum constant is areference variable to enum type object.
->Every enum constant is always public static final by default.

Ex.



Declaration and usage of enum:-

Ex.

```
enum  Beer
{
KF,KO,RC,FO;
}
class Test
{
p.s.v.main(String[] args)
{
Beer b=Beer.KF;
}
}
```

->We can declare enum either with in the class or outside of the class bu not inside a method.
->If we are trying to declare enum with in a method we will get compile time error.

Ex.

```
Enum X          | class  y          |          class y
{               | {                 |  {
Class y         | enum x            |          public void m1()
}               | }                 |  {
}               | }                 |  enum x;
                | }                 |  }}}
```

Write write wrong

Note:enum type must not be local

->If we declare enum outside the class the applicable modifiers are public,default, strict fp
->If we declare enum with in a class the applicable modifiers are public,default,strictfp,private,protected,static.

Enumvs switch statement:-
->Until 1.4v the allowed datatype for seitch arguments are bye,short,char,int.
->But from 1.5v onwards to above the corresponding wrapper classes Byte,char,integer,+enum type also allowed
switch()

{

}

1.4v	1.5v	1.7v
Byte Short Char int	Byte Short Character Integer And Enum	String

->Hencefrom 1.5v version onwards we can use enum as argument to swich statement.

Ex.

```
Enum Beer
{
KF,KO,RC,FO;
}
class test
{
p.s.vmain(String[] args)
{
Beer b1=Beer.RC;
Switch(b1)
{
Case KF:
        s.o.pln("It is children brand");
            break;
case KO:
        s.o.pln("It is too lite");
        break;
case RC:
        s.o.pln("It is challenger brand");
        break;
case FO:
        s.o.pln("buy one get one")
```

```
            break;
    default:
            s.o.pln("after brands not recommended to take");
    }
    }
    }
```

o/p:
It is challengers brand

->If we are passingenum type as arguments to switch statements every case label should be a valid enum constant.

Ex.

```
Enum Beer
{
KF,KO,RC,FO;
}
Beer b1=Beer.KF;
Switch(b1)
{
Case KF;
Case KO;
Case RC;
Case KALYANI:  X        C.E: Unqualified enumeration constant name required
}
}
}
```

Enumvs 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 wan'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 implements any no. of interface at a time.

Ex.

1)
```
Enum x
{
Enum y extends x
{
    //wrong
}
```
C.E:-
Cannot inherit form final x
Enum types not extensible

(2)
```
enum x extends java.lang.enum
{
}
```
//wrong

(1)
```
Enum x
{
        //write
}

class y extends x
{
}
```
C.E1: cannot inherit from final x

(4)
```
class x
{
}

enum y extends x
{
}
```
//wrong

C.E2: enumtypes are not extensible

(5)

```
Interface x
{


}
Enum y implements x
{
                                        //write
}
```

Java.lang.Enum:-

->every enum in java is always direct child class of java.lang.enum.class.

->The power of enum is inheriting from this class only to our enum class.

->It is an abstract class and direct child class of object class.

->This class implements comparable and serializable interface. Hence every enum in java is bytedefaultserializable and comparable.

Values() method:-

->With in the enum the orders of constants is important we can specify its orders by using original value. ->We can find ordinal value of enum constants by using ordinal method.

publicint ordinal();

->Ordinal value is zero-based

Ex.

```
Enum Beer
{
KF,KO,RC,FO;
}
class Test
{
p.s.v.main(String[] args)
{
Beer[] b=Beer.value();
For(Beer b1=b)
{
s.o.pln(b1+"……."+b1.ordinal());
}
}
}
```

o/p:-

KF….0

KO……1

RC…..2

FO…..3

Enum class Constructors and speciality of java enum:-

->When compared with old languages enum,Javaenum is more powerful because in addition to constants we can take variables, methods, constructors etc… Which may not possible in old language this extra facility is due to internal implementations of enum concept which is class based.

->Inside enum we can declare main() methods and hence we can invoke enum class directly from command prompt.

Ex.

```
Enum fish
{
STAR,GOLD,GUPPY,APOLLO,KILLER;          mandatory
p.s.v.main(String[] args)
{
s.o.pln(“”ENUM MAIN METHODS”);
}
}
```

```
>javac fish.java
>java fish
```

o/p:- enum main method

->Inside enumwith out having constants we can’t to take any extra members, but empty enum is always valid

```
Ex. Enum color
{
```

```
Public void m1()
{
        //wrong
}

Enum color
{
        //write
}
```

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

```
Ex1.
Enum Beer
{
KF,KO,RC,FO;
Beer()
{
s.o.pln("constructors");
}
}
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 and hence we can't call constructors directly.
Beer b=new Beer(); X or wrong

C.E:
Enumtypes may not be instantiated

```
Ex. Enum Beer
{
KF(75),KO(90),RC(70),FO;
int price;
Beer(int price)
{
this.price=price;
}
Beer()
{
this.price=65;
}
Publicintgetprice()
{
return price;
}
}
class Test
{
p.s.v.main(String[] args)
{
Beer() b=Beer.values();
For(Beer b1=b)
{
S.o.pln(b1+"……"+b1.getprice());
}
}
}
```

o/p:-
KF…..75

KO….90
RC….70
FO….65
->Within the enum we can take instance and 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 methods also.

Ex.
Q)Which of the fallowing expressions are valid
(1) Beer.KF.equals(Beer.RC) //false
(2)Beer.KF,hashcode() // write
(3)Beer==Beer.RC //false
(4)Beer.KF >Beer.Rc //wrong
(5) Beer.KF.ordinal >Beer.RC.ordinal //write

Case(1):

```
Package pack1;          package pack2
Public enum fish              class Test1
{                       {
STAR,GUPPY,APOLLO;      p.s.v.m(String[] args)
}                       {
                        s.o.pln(STAR);          //import static pack1.fish.STAR
                                                  //import static pack1.fish.*;
                        }
                        }
```

```
Package pack3;              package pack4;
Class Test2                       class Test3
{                           {
p.s.v.m(String args[])      p.s.v.main(String args[])
{                           {
Fish f=fish.STAR();               fish f=fish.STAR;  //import pack1.fish(m)
                                        //import pack1.*;
s.o.pln(f);                       s.o.pln(GUPPY);//import static pack1.fish.GUPPY
    or
                                        // import static pack1.fish.*;
}                           }
```

```
}                                   }

Import pack1.fish;
Or
Import pack1.*;
```

Case(2):

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.main(String[] args)

{

```
Color[] c=color.value();
For(color c1=c)
{
C1.info();
}
}
}
```

o/p:-

universal color

Dangerous color

Universal color

enumVsEnum VS Enumeration:-

enum:-

->It is a keyword which can be used to define a group of named constants.

Enum:-

->It is a class parent in java.lang package which acts as a base class for all java enums.

Enumeration:-

->It is an interface present in java.utilpackage,which can be used for retrieving objects from collection one by one.