**ENUM – Part\_03**

* **Enum VS constructor:**

An enum can contain constructor.

Enum constructor will be executed separately for every enum constant at the time of enum class loading automatically.

Example:

enum Beer{

KF,KO,RC,FO

Beer(){

Sytem.out.println(“constructor”); // Line-1

}

}

class Test{

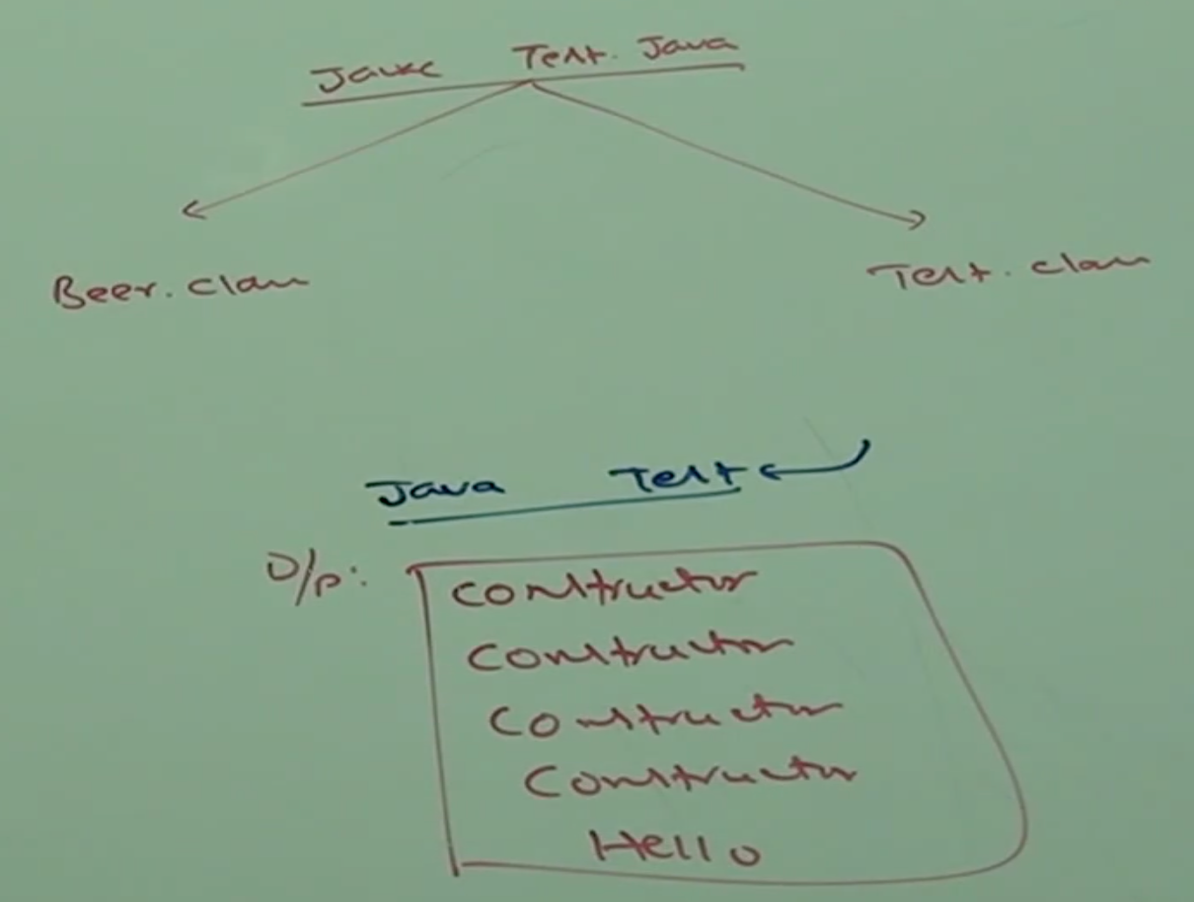
public static void main(String[] args){

Beer b = Beer.RC;

System.out.println(“Hello”);

}

}



If we comment Line-1 then the output is: Hello

We can’t create enum object directly and hence we can’t invoke enum constructor directly.

Beer b = new Beer();

CE: enum type may be not instantiated

* **Full fledged example:**

enum Beer{

KF(70),KO(80),RC(90),FO;

int price;

Beer(int price){

this.price = price;

}

Beer(){

this.price = 65;

}

public int getPrice(){

return price;

}

}

class Test{

public static void main(String[] args){

Beer[] b = Beer.values();

for(Beer b1: b){

System.out.println(b1+”….”+b1.getPrice());

}

}

}

Output:

KF….70

KO….80

RC….90

FO….65

Note:

KF ===> public static final Beer KF = new Beer();

KF(70) ===> public static final Beer KF = new Beer(70);

Inside we can declare methods but should be concrete methods only and we can’t declare abstract methods.

* **Case\_01:**

Every enum constant represents an object of the type enum. Hence, whatever methods we can apply on norma Java objects, can be applicable on enum constants also.

Example:

Beer.KF.equals(Beer.RC); // Valid

Beer.KF.hashCode() > Beer.RC.hashCode(); // Valid

Beer.KF < Beer.RC; // Invalid, we can use relational operators with object.

Beer.KF.ordinal() > Beer.RC.ordinal(); // Valid

* **Case\_02:**

If we want to any class or interface directly from outside package, then the required import is normal import.

If we want to access static members without class name then the required import is static import.

import static java.lang.Math.sqrt;

import java.util.ArrayList;

class Test{

public static void main(String[] args){

ArrayList l = new ArrayList();

System.out.println(sqrt(4));

}

}

Example\_01:

package pack1;

public enum Fish{

START,GUPPY;

}

package pack2;

public class Test{

public static void main(String[] args){

Fish f = Fish.START;

System.out.println(f);

}

}

The required import is:

import pack1.FISH;

or

import pack1.\*;

package pack3;

public class Test{

public static void main(String[] args){

System.out.println(STAR);

}

}

The required import is:

import static pack2.Fish.START;

or

import static pack2.Fish.\*

package pack4;

public class Test{

public static void main(String[] args){

Fish f= Fish.STAR;

System.out.println(GUPPY);

}

}

The required imports are:

import pack1.Fish;

or

import pack1.\*

import static pack1.Fish.GUPPY;

or

import static pack1.Fish.\*

* **Case\_03:**

enum Color{

BLUE,RED,GREEN;

public void info(){

System.out.println(“universal color”);

}

}

class Test{

public static void main(String[] args){

Color[] c = Color.values();

for(Color c1 : c){

c1.info();

}

}

}

Output:

universal color

universal color

universal color

enum Color{

BLUE,RED{

public void info(){

System.out.println(“Dangerous color”);

}

}, GREEN;

public void info(){

System.out.println(“universal color”);

}

}

class Test{

public static void main(String[] args){

Color[] c = Color.values();

for(Color c1: c){

c1.info();

}

}

}

Output:

universal color.

dangerous color.

universal color.

* **Case\_04 (enum vs Enum vs Enumeraton):**

enum:

Is a keyword in Java which can be used to define a group of named constants.

Enum:

Enum is a class in Java present in java.lang package. Every enum in Java should be direct chid class of Enum class. Hence, this class acts as based class for all Java enums.

Enumeration:

Enumeration is an interface present in java.util package.

We can use Enumeration object to get objects one by one from the collection.