**ENUM – Part\_01**

* **Enum (Enumeration):**

If we want to represent a group of named constants then we should go for ENUM.

Example:

enum Month{

JAN, FEB, MAR, …, DEC

}

enum Beer{

KF, KO, RC, FO;

}

In the enum declaration, ; (semi-colon is optional).

The main objective of enum is to define our own data types (Enumarated Data Types).

Enum concept introduced in 1.5 version. When compared with old languages ENUM, Java ENUM is more powerful.

* **Internal implementation of Enum:**

1. Every ENUM is internally implemented by using class concept.
2. Every ENUM constant is always “public static final”.
3. Every ENUM constant represents an object of the type ENUM

Example:

enum Beer{

KF, RC;

}

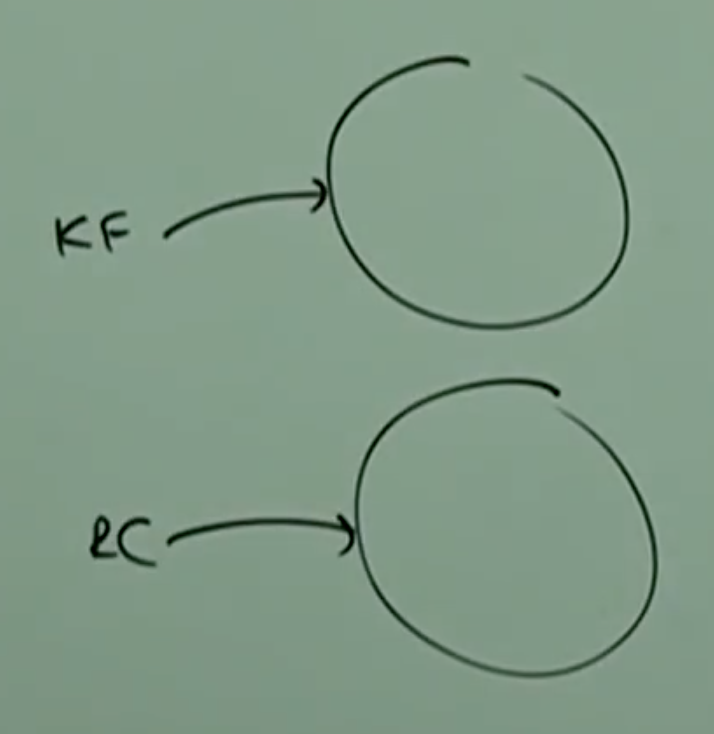
Is translated to

class Beer{

public static final Beer KF = new Beer();

public static final Beer RC = new Beer();

}



* **ENUM declaration and usage:**

Every ENUM constant is always “public static final” and hence we can access ENUM constant by ENUM name.

Example:

enum Beer{

KF, KO, RC, FO;

}

class Test{

public static void main(String[] args){

Beer b = Beer.RC;

System.out.println(b);

}

}

Output: RC

Note:

Inside ENUM toString() method is internally implemented to return name of the constant.

We can declare ENUM either within the class or outside the of the class but not inside a method.

If we are trying to declare inside a method, then we will get compile time error saying:

“Enum types must not be local”

Example\_01:

enum X{

}

class X{

}

// Valid

Example\_02:

class X{

enum Y{

}

}

// Valid

Example\_03:

class X{

public void m1(){

enum Y{

}

}

}

CE: enum types must not be local.

If we declare enum outside of the class, the applicable modifiers are

public

default

strictfp

If we declare enum inside a class, the applicable modifiers are:

public

default

strictfp

private

protected

static

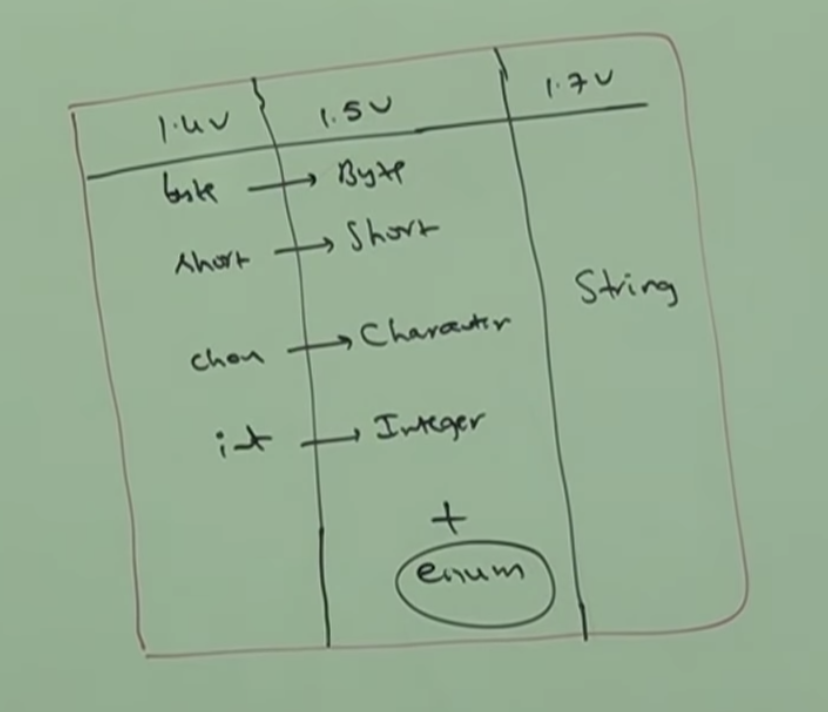
* **Enum VS switch:**

Until 1.4 version the allowed argument types for the swtich statement are

byte,shor, char, int.

But from 1.5 version onwards corresponding wrapper classes and enum types are allowed.

From 1.7, version onwards String type also allowed.



Hence, from 1.5 version onwards we can pass enum type as argument to switch statement.

* **Using enum with switch:**

enum Beer{

KF, KO, RC, FO;

}

class Test{

public static void main(String[] args){

Beer b = Beer.RC;

switch(b){

case KF:

System.out.println(“It is childrens brand”);

break;

case KO:

System.out.println(“It is too light”);

break;

case RC:

System.out.println(“It is not that much kick”);

break;

case FO:

System.out.println(“Buy one get one”);

break;

default:

System.out.println(“Other brands are not recommended”);

}

}

}

Output: It is not that kick

Note:

If we pass enum type as argument to switch statement, then every case label should be valid enum constant, otherwise we will get compile time error:

Unqualified enumeration constant name required.