ENUM

1. We can use **enum** to define a group of named constants.
2. By using enum we can define our own data types which are also come enumerated data types.

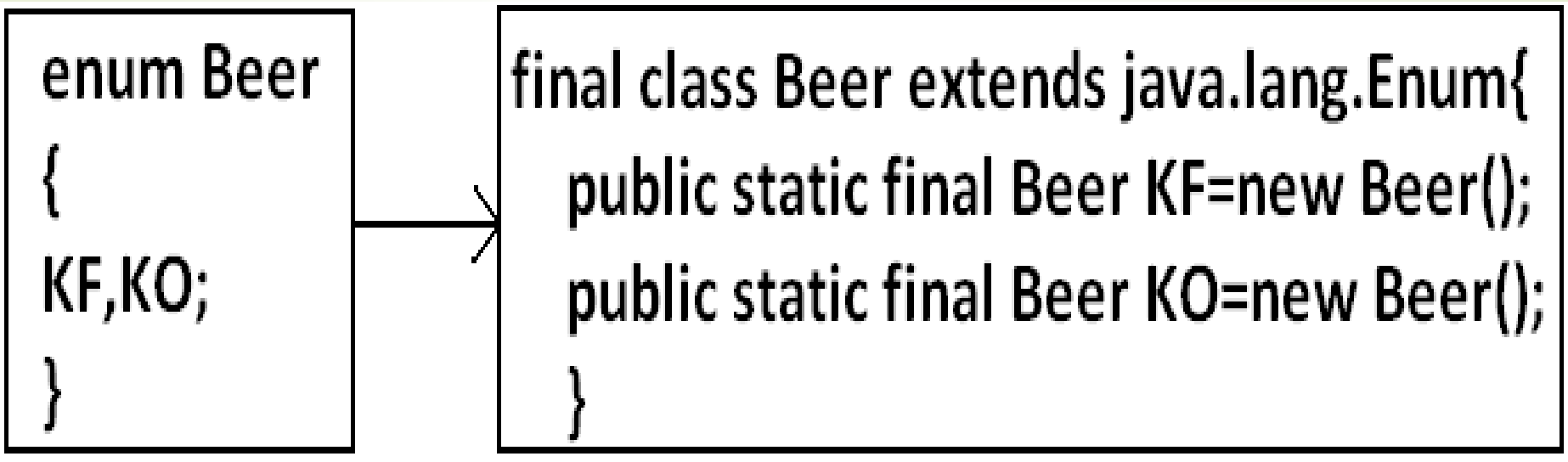
Example 1:

```
enum  Month
{
    JAN, FEB, MAR,  ... DEC;      //; -->optional
}
```

Internal implementation of enum:

- ✓ Internally **enum's** are implemented by using class concept.
- ✓ Every **enum** constant is a reference variable to that **enum** type object.
- ✓ Every **enum** constant is implicitly **public static final** always.
- ✓ Every **enum** constant internally static hence we can access by using "**enum name**".
- ✓ Internally inside every enum **toString()** method is implemented to return name of the constant.

Example 3:



enum Vs Enum Vs Enumeration :

enum : enum is a keyword which can be used to define a group of named constants.

Enum :

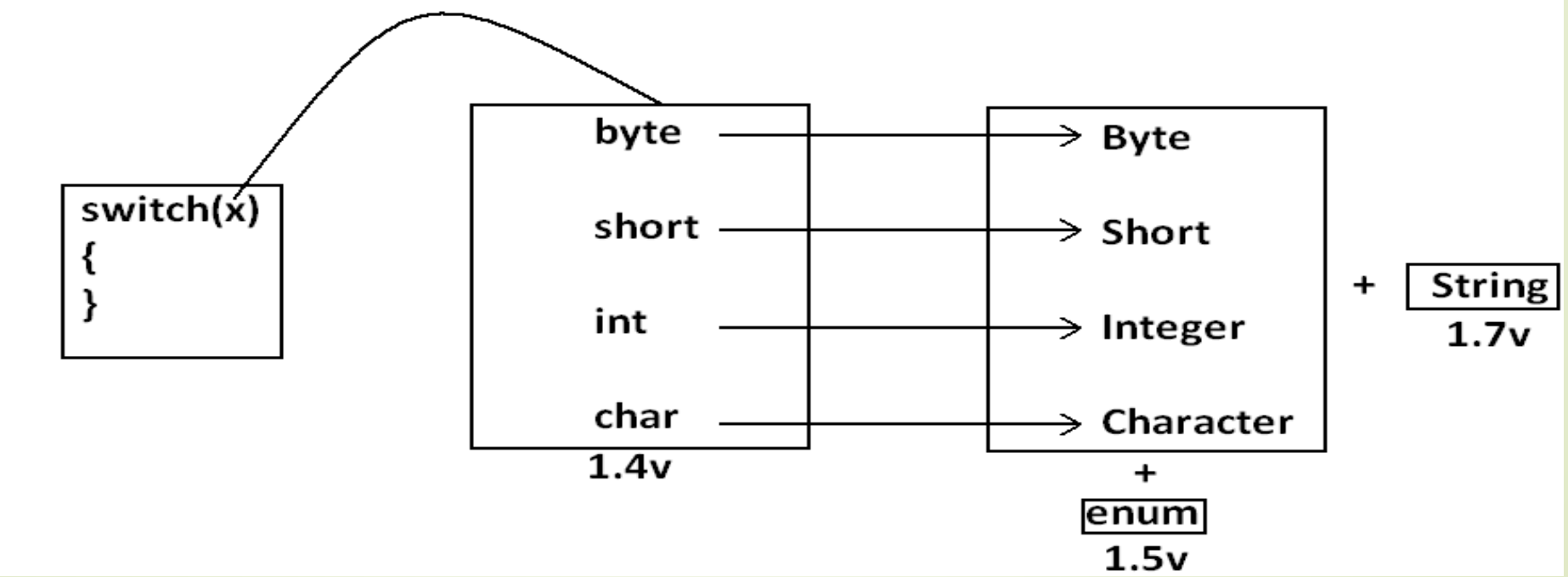
1. It is a class present in **java.lang** package .
2. Every enum in java is the direct child class of this class. Hence this **Enum** class acts as base class for all java enum's .

Enumeration :

1. It is a interface present in **java.util** package .
2. We can use Enumeration to get the objects one by one from the Collections.

Enum vs switch statement:

- Until 1.4 versions the allowed argument types for the switch statement are byte, short, char int.
- But from 1.5 version onwards in addition to this the corresponding wrapper classes and enum type also allowed.
- That is from 1.5 version onwards we can use enum type as argument to switch statement.



Points to remember:

- If we are passing **enum** type as argument to switch statement then every case label should be a valid **enum** constant otherwise we will get compile time error.
- We can declare **enum** either outside the class or within the class but not inside a method.
- If we declare **enum** outside the class the allowed modifiers are :
  1. public
  2. default
  3. strictfp
- If we declare **enum** inside a class then the allowed modifiers are :

  public private

  default + protected

  strictfp static

Example:

```
enum X
{ }
class Y
{ }

(valid)
```

```
class X
{
enum Y
{}
}

 (valid)
```

```
class X
{
public void methodOne(){
enum X
{}
}
}
```

output:

compile time error.

D:\Enum>javac X.java

X.java:4: enum types must not be local

enum X

Enum vs inheritance:

- ✓ Every **enum** in java is the direct child class of **java.lang.Enum** class hence it is not possible to extends any other **enum**.
- ✓ Every **enum** is implicitly final hence we can't create child **enum**.
- ✓ Because of above reasons we can conclude inheritance concept is not applicable for **enum's** explicitly. Hence we can't apply extends keyword for **enum's** .
- ✓ But **enum** can implement any no. Of **interfaces** simultaneously.

Enum vs inheritance:

```
enum X
{}
enum Y extends X
{}
```

(invalid)

```
enum X extends Enum
{}
```

(invalid)

```
class X
{}
enum Y extends X
{}
```

(invalid)

Enum vs inheritance:

```
enum X
{}
class Y extends X
{}
```

output:

```
compile time error.
D:\Enum>javac Y.java
Y.java:3: cannot inherit from final X
class Y extends X
 Y.java:3: enum types are not extensible
class Y extends X
```

(invalid)

```
interface X
{}
enum Y implements X
{}
```

(valid)

Java.lang.Enum class:

- Every enum in java is the direct child class of **java.lang.Enum** class. Hence this class acts as base class for all java **enums**.
- It is abstract class and it is direct child class of "**Object class**"
- It implements **Serializable** and **Comparable**.

values() method and ordinal() method:

1. values() method:

Every enum implicitly contains a static values() method to list all constants of enum.

Example:

```
Months[] b = Months.values();
```

2. ordinal() method:

Within enum the order of constants is important we can specify by its ordinal value.
We can find ordinal value(index value) of enum constant by using ordinal() method.

Example:

```
public final int ordinal();
```

Speciality of java enum:

- ✓ When compared with old languages **enum**, java's enum is more powerful because in addition to constants we can take normal variables, constructors, methods etc which may not possible in old languages.
- ✓ Inside **enum** we can declare main method and even we can invoke **enum** directly from the command prompt.

Enum vs constructor:

- enum can contain constructor.
- Every enum constant represents an object of that enum class which is static hence all enum constants will be created at the time of class loading automatically and hence constructor will be executed at the time of enum class loading for every enum constants.