1.13 **ENUMS**

Table of Contents

1.13.1 **Purpose:**

Enums are lists of constants. When you need a predefined list of values which do not represent some kind of numeric or textual data, you should use an enum. For instance, in a chess game you could represent the different types of pieces as an enum:

```
enum ChessPiece {
PAWN,
ROOK,
KNIGHT,
BISHOP,
QUEEN,
KING;
}
```

You should always use enums when a variable (especially a method parameter) can only take one out of a small set of possible values. Examples would be things like type constants (contract status: "permanent", "temp", "apprentice"), or flags ("execute now", "defer execution").

1.13.2 Introduction

Enumerations already existed in other languages like C, C++, SmallTalk etc. Hence Java Enums are not something very new to programmers. Enums are used primarily to handle a collection of logically grouped constants. In Java, for ages we have been handling constants (number constants here) through separate Java files, sometimes by encapsulating those in relevant POJOs, and even using a wrong design style of implementing an interface containing all constants.

We used these constants in conditions implemented using 'if' statement or switch cases also. First thing you can notice about all these approaches is that this made the code look clumsy. Usage of interfaces made the code unnecessarily complex. Next, though the constants had something common, the language hardly provided anything to treat them as an object. List of drawbacks doesn't end here. Let us take an example below and see what other drawbacks ar

```
public class StatusWithoutEnums {

   public static final int STATUS_OPEN = 0;
```

```
    public static final int STATUS_STARTED = 1;
    public static final int STATUS_INPROGRESS = 2;
    public static final int STATUS_ONHOLD = 3;
    public static final int STATUS_COMPLETED = 4;
    public static final int STATUS_CLOSED = 5;
}
```

Drawbacks of This Approach:

➢**Type Safety:** **All statuses above may carry some business meaning, but in Java language context, these are just int values. This means any int value is a status for this Java program. So the program using these statuses can break with any int value not defined in this group.**

➢**Compile Time Constants:** **All these constants are compiled and used in the program. If you want to add any new constant then you will have to add it to the list and recompile everything.**

➢**Uninformative:** **When printed, these are just numbers for a reader. E.g. if a program prints status = 3, the reader will have to go and find out what does it actually mean. Amount of information available with the print is minimal.**

➢**Restricted Behavior:** **If you want to use these values in a remote call, or compare them, then you will have to handle it explicitly. Serializable, Comparable interfaces offered by Java for same purpose cannot be used. Also there is no room to add any additional behavior to the statuses, e.g. attaching basic object behavior of hashCode(), toString() and equals() methods.**

➢**Meaningless Switch-Case Use:** **When you use statuses above in switch case, you cannot use the variable names; instead you will have to use the meaningless/uninformative numbers to compare. Readability of program goes down considerably in this case.**

➢**Non-Iterative List:** **This is a list of values, but you cannot iterate over it the way you can on any collection.**

The solution to above problem is Enum data type in Java 5. Concept of Enum is obtained from the counterpart technologies like C, C++, C# etc. ***~~Introduction:~~***

An enum type, also called enumeration type, is a type whose fields consist of a fixed set of constants.

1.13.3 Detailed Explanation:

The Java programming language contains the enum keyword, which represents a special data type that enables for a variable to belong to a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.

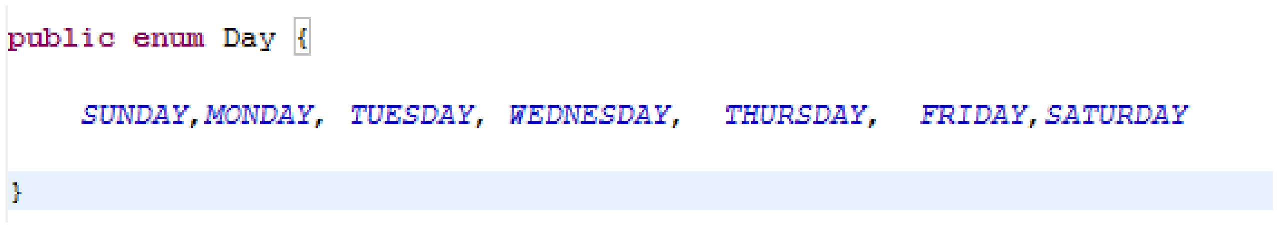
The values defined inside an enum are constants and shall be typed in uppercase letters. Also, these values are implicitly static and final and cannot be changed, after their declaration. If an enum is a member of a class, then it is implicitly defined as static. Finally, you should use an enum, when you need to define and represent a fixed set of constants.

A very basic enum declaration

Syntax:

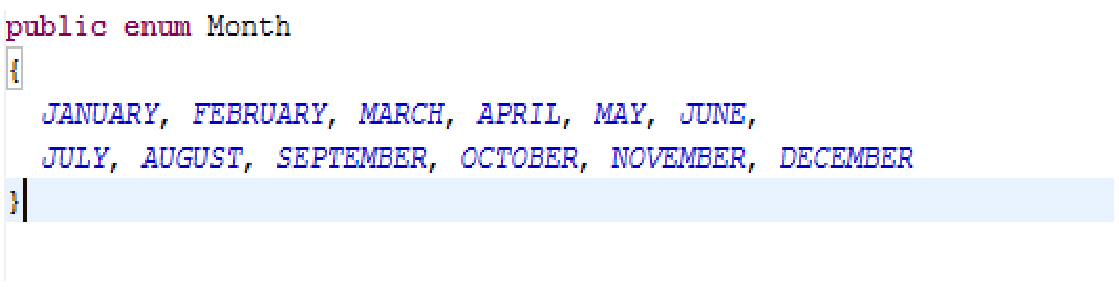
```
access-modifier enum enum-name
 {
          enum_type_1,
            enum_type_2      //(optional semi colon)
 }
```

The most common example of a simple enum includes the days of the week, as shown below



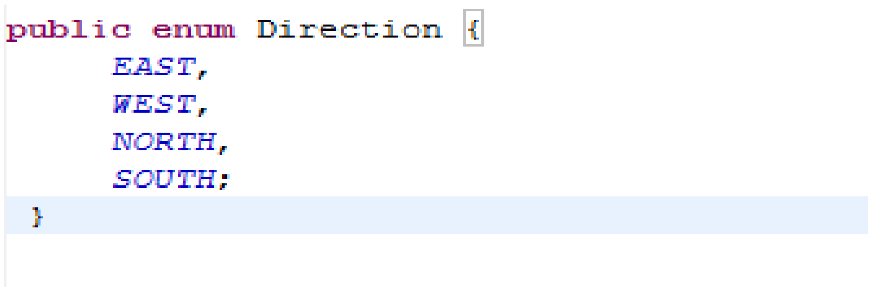
```
public enum Day {

    SUNDAY,MONDAY, TUESDAY, WEDNESDAY,  THURSDAY,  FRIDAY,SATURDAY

}
```

Some Examples:



```
public enum Month
{
  JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE,
  JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER
}
```

2.



```
public enum Direction {
      EAST,
      WEST,
      NORTH,
      SOUTH;
 }
```

Note: ***Whenever an enum is defined, a class that extends java.lang.Enum is created. Hence, enum cannot extend another class or enum. The compiler also create an instance of the class for each constants defined inside the enum.***

The java.lang.Enum has these methods:

public final String name(); // Returns the name of this enum constant, exactly as declared in its enum declaration.
// You could also override the toString() to provide a more user-friendly description.

public String toString(); // Returns the name of this enum constant, as contained in the declaration.
// This method may be overridden.

public final int ordinal(); // Returns the ordinal of this enumeration constant.

Properties of Enum:

- **All constants defined in an enum are public static final. Since they are static, they can be accessed via EnumName.instanceName.**
- **You do not instantiate an enum, but rely the constants defined.**
- **Enums can be used in a switch-case statement, just like an int.**
- **To define Enum type you have to use enum keyword.**
- **All enums extend java.lang.Enum.**
- **You can define Enum inside class as any class member.**
- **Enum can't extends (inherit) any other type.**
- **Enum type constructor must be package-private.**
- **Enum declaration defines a class (called an enum type).**
- **Enum class can include methods and other variables.**
- **Compiler automatically adds some methods when it compile an Enum such as values() and valueOf() functions.**

Defining Enums: ***The enum can be defined within or outside the class because it is similar to a class.***

Example of enum that is defined outside the class:

```
enum Season { WINTER, SPRING, SUMMER, FALL }

class EnumExample2{
public static void main(String[] args) {

Season s=Season.WINTER;
System.out.println(s);

}}
```

Output:WINTER

Example of enum that is defined within the class:

```
class EnumExample2{
enum Season { WINTER, SPRING, SUMMER, FALL; }//semicolon(;) is optional here

public static void main(String[] args) {
Season s=Season.WINTER;//enum type is required to access WINTER
System.out.println(s);

}}
```

Output:WINTER

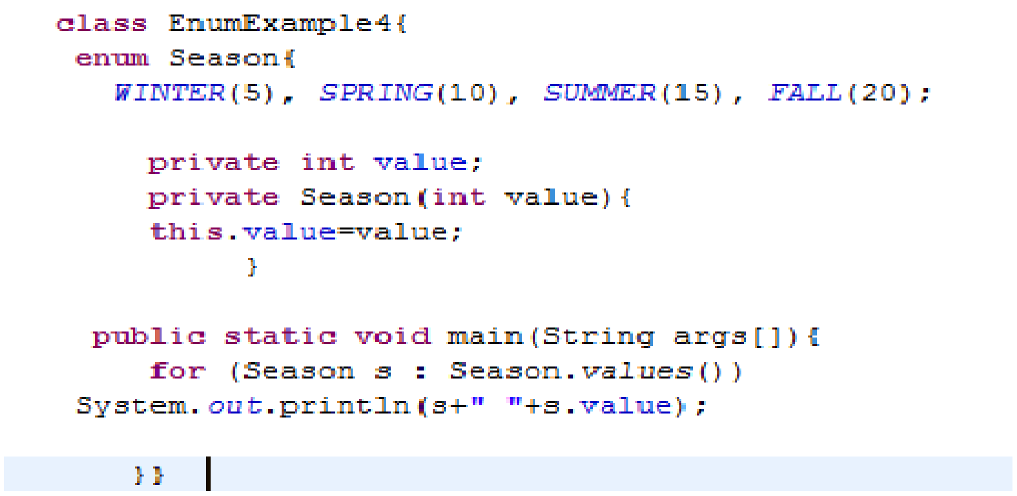
Note:**Every time you create enum in java you are creating a fully-fledged enum type or class and by default they provide basic implementation of all Object class methods. In addition to getting a complete enum type in your code, they are also Comparable and Serializable.**

After your enum is created you can declare variables of enum type to store one of the enum predefined variable.

Initializing specific value to the enum constants:

The enum constants have initial value that starts from 0, 1, 2, 3 and so on. But we can initialize the specific value to the enum constants by defining fields and constructors. As specified earlier, Enum can have fields, constructors and methods.

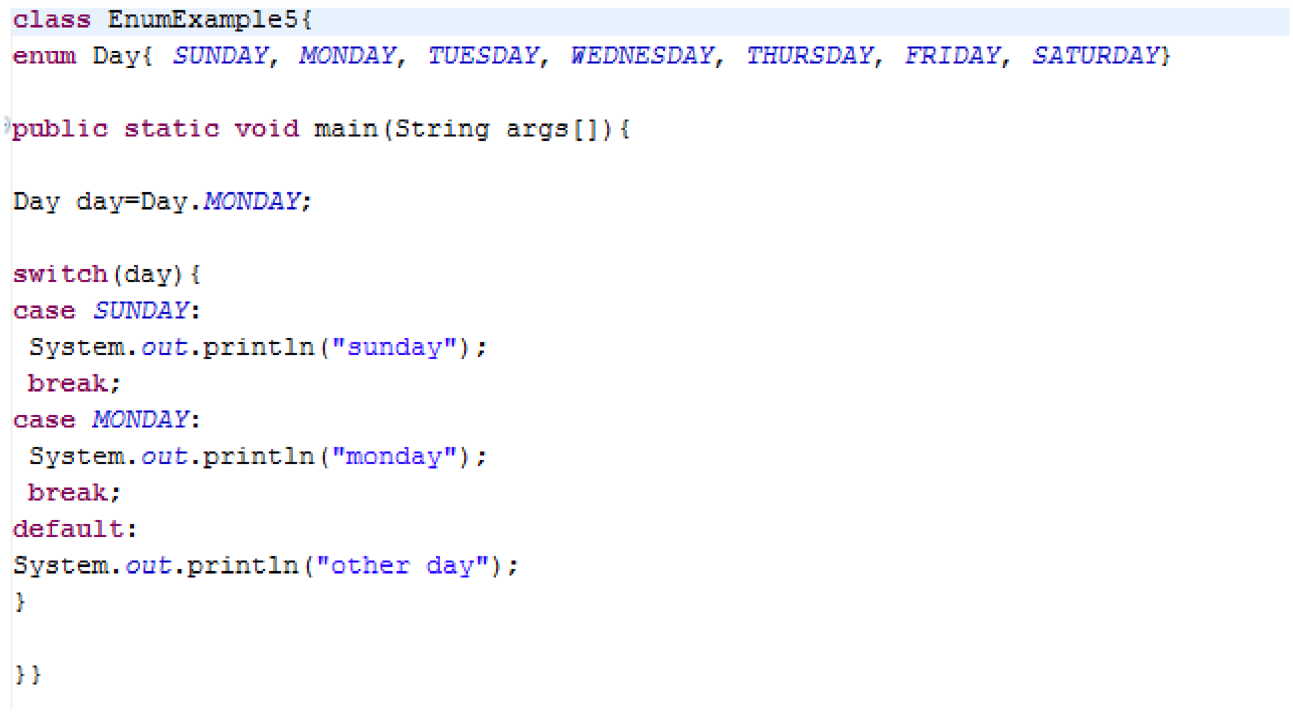
Example of specifying initial value to the enum constants



```
class EnumExample4{
 enum Season{
   WINTER(5), SPRING(10), SUMMER(15), FALL(20);

     private int value;
     private Season(int value){
     this.value=value;
          }

  public static void main(String args[]){
     for (Season s : Season.values())
 System.out.println(s+" "+s.value);

     }}
```

Output:WINTER 5
SPRING 10
SUMMER 15
FALL 20

Example of applying enum on switch statement



```
class EnumExample5{
enum Day{ SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY}

public static void main(String args[]){

Day day=Day.MONDAY;

switch(day){
case SUNDAY:
 System.out.println("sunday");
 break;
case MONDAY:
 System.out.println("monday");
 break;
default:
System.out.println("other day");
}

}}
```

```
Output:WINTER
       SPRING
```

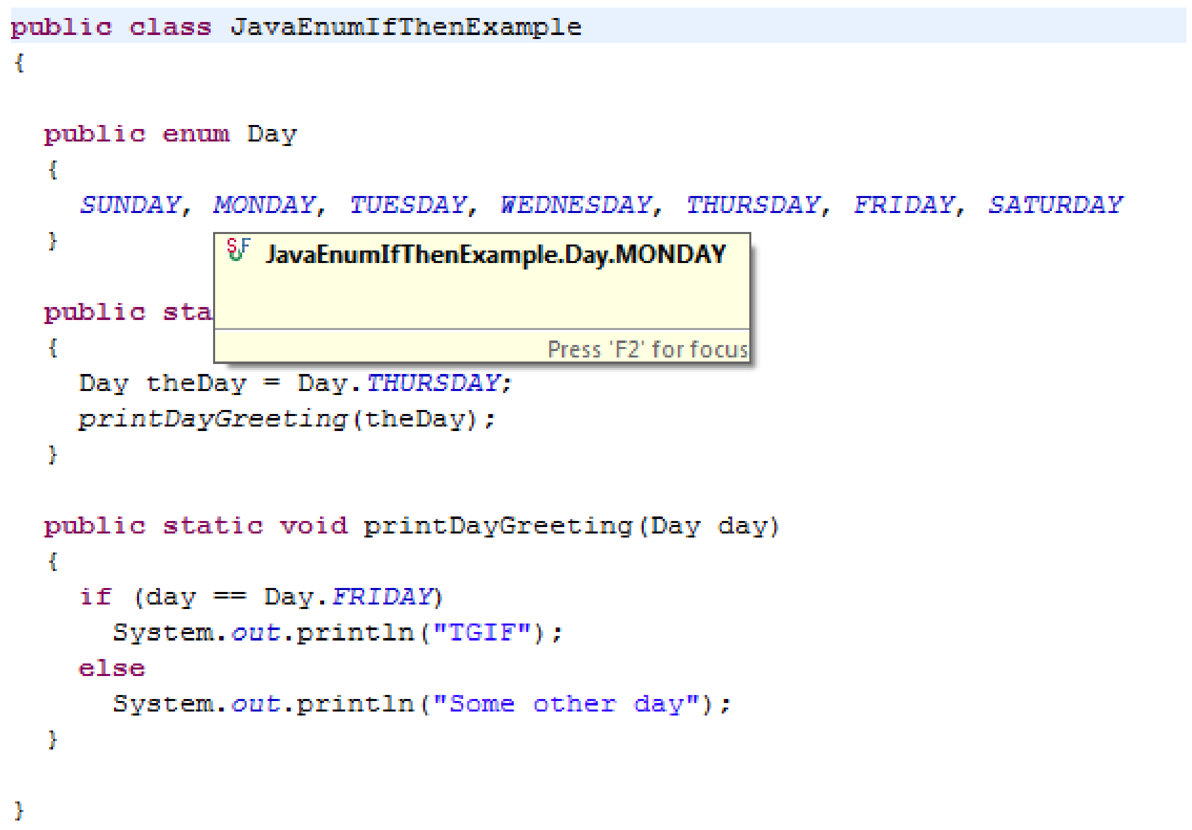
```
SUMMER
FALL
```

Example 2:enum using for loop

If you need to iterate through all the constant fields in a Java enum, you can do this pretty easily in a Java 5 for loop. You just need to use the values method of the enum type in the for loop, as shown in this Java enum for loop example:

Example 3:enum using if/then example

You can also use a Java enum type in an if/then statement. While the example below might be better implemented as a switch/case statement, it does provide an enum if/then example:



```
public class JavaEnumIfThenExample
{

  public enum Day
  {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
  }

  public sta
  {
    Day theDay = Day.THURSDAY;
    printDayGreeting(theDay);
  }

  public static void printDayGreeting(Day day)
  {
    if (day == Day.FRIDAY)
      System.out.println("TGIF");
    else
      System.out.println("Some other day");
  }

}
```

1.13.4 **Benefits with enum:**

1) Enums in Java are type-safe, It means your enum will have a type for example "Currency" in below example and you can not assign any value other than specified in Enum Constants.

public enum Currency {PENNY, NICKLE, DIME, QUARTER};

Currency coin = Currency.PENNY;

coin = 1; //compilation error

2) Enum has its own name-space,we can use enum field only with class name(refer to the below Currency example)

3) Best feature of Enum is you can use Enum in Java inside Switch statement like int or char primitive data type.We will also see example of using java enum in switch statement in this java enum tutorial.

4) Adding new constants on Enum in Java is easy and you can add new constants without breaking existing code.

1.13.5 Important points about Enum in Java

1/**Enum in Java are reference type like class or interface and you can define constructor, methods and variables inside java Enum which makes it more powerful than Enum in C and C++** .

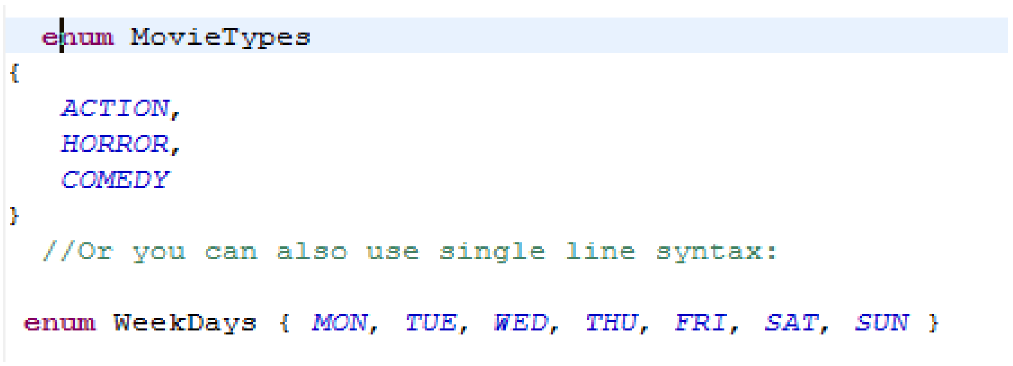
2.You can specify values of enum constants at the creation time as shown in below example:

But for this to work you need to define a member variable and a constructor because PENNY (1) is actually calling a constructor which accepts int value , see below example.

```
 public enum Currency {
     PENNY(1), NICKLE(5), DIME(10), QUARTER(25);
     private int value;

     private Currency(int value) {
          this.value = value;
     }
   public static void main(String[] args){
   int i = 1;
   for (Currency curr: Currency.values())
{
System.out.printf("Currency %s = %s\n", i++, curr);   }
```

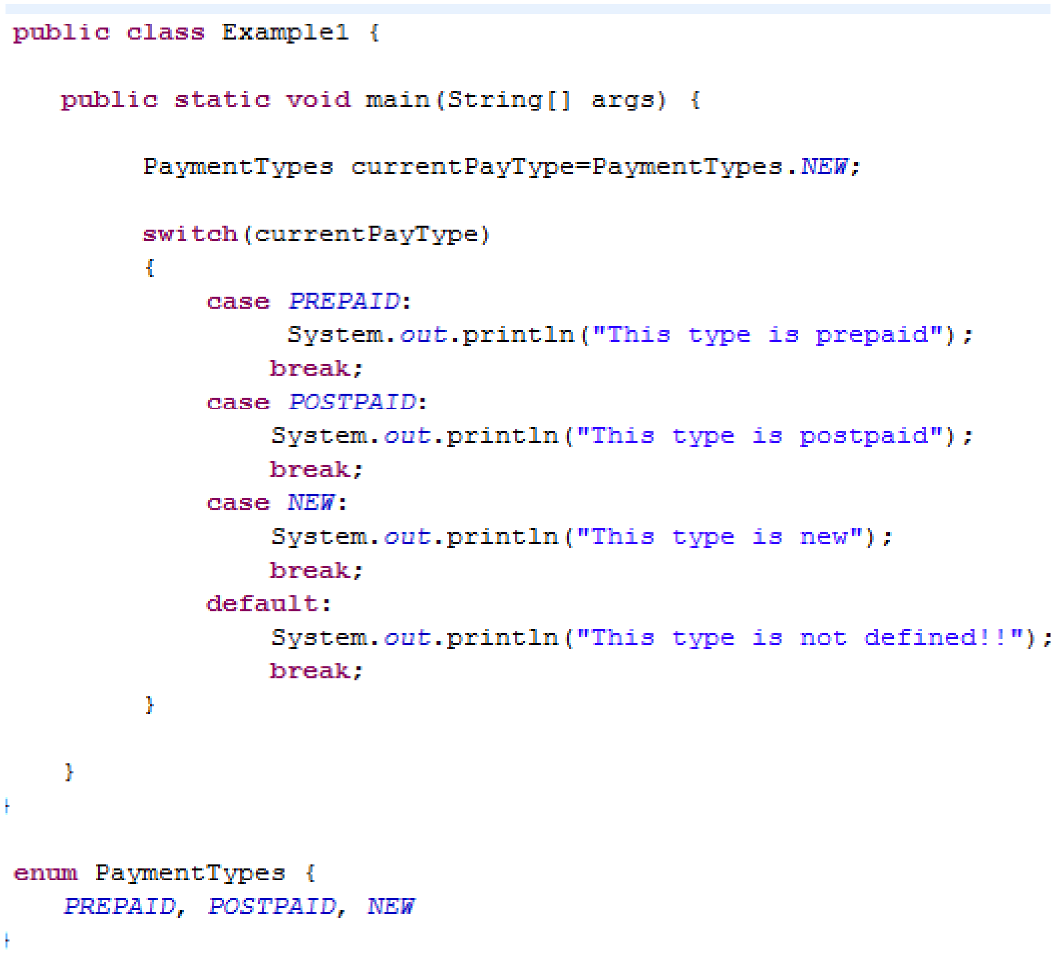
1.13.6 Code Snippets:

1.



```java
  enum MovieTypes
{
   ACTION,
   HORROR,
   COMEDY
}
  //Or you can also use single line syntax:

 enum WeekDays { MON, TUE, WED, THU, FRI, SAT, SUN }
```

2.The following example defines a string Enums values and use switch expression to control decision based on Enum values



```java
public class Example1 {

   public static void main(String[] args) {

        PaymentTypes currentPayType=PaymentTypes.NEW;

        switch(currentPayType)
        {
            case PREPAID:
                 System.out.println("This type is prepaid");
                break;
            case POSTPAID:
                System.out.println("This type is postpaid");
                break;
            case NEW:
                System.out.println("This type is new");
                break;
            default:
                System.out.println("This type is not defined!!");
                break;
        }

   }
}

enum PaymentTypes {
   PREPAID, POSTPAID, NEW
}
```

The output of the above ENUM example

TCS Internal

java code
?
1
This type is new

1.13.7 Check your Self:

1.enum can be used to define a set of enum constants.

A.TRUE   B.FALSE

2.Select all the correct statements regarding Enums (Select any 2 options)
a) All enums are subclasses of interface java.lang.Enum.
b) Enums is simply a data structure and hence it is not compiled to a .class file.
c) Enums enable you to define a new data type. For example, If you create an Enum 'Days' you can declare a variable of type 'Days'
d)All instances of Enums are serializable by default.

3.Select all the incorrect options:
a) Since enums are comparable to traditional classes, they may not be arguments in switch statements.
b) Enums may not extend another class/ enum.
c) Enums inherit the java.lang.Object class.
d)Enums may be extended by another Enum.

4.What is the output of the following code:

```
public enum IceCream {
        VANILLA ("white"),
        STRAWBERRY ("pink"),
        WALNUT ("brown"),
        CHOCOLATE ("dark brown");

        String color;

        IceCream (String color) {
                this.color = color;
        }
```

```
                    public static void main (String[] args) {
                              System.out.println (VANILLA);
                              System.out.println (CHOCOLATE);
                    }
          }
```

Options:

a) Compilation error : Cannot run an enum as a standalone application.
b) Compilation error at line no 14 & 15 : Cannot access VANILLA and CHOLOCLATE in 'static' main method.
c) No errors. Output:

```
          VANILLA
          CHOCOLATE
```

d) No errors. Output:

```
          white
          dark brown
```

***5.*Enumerated values are used to represent a set of named values.**

A.YES   A.NO

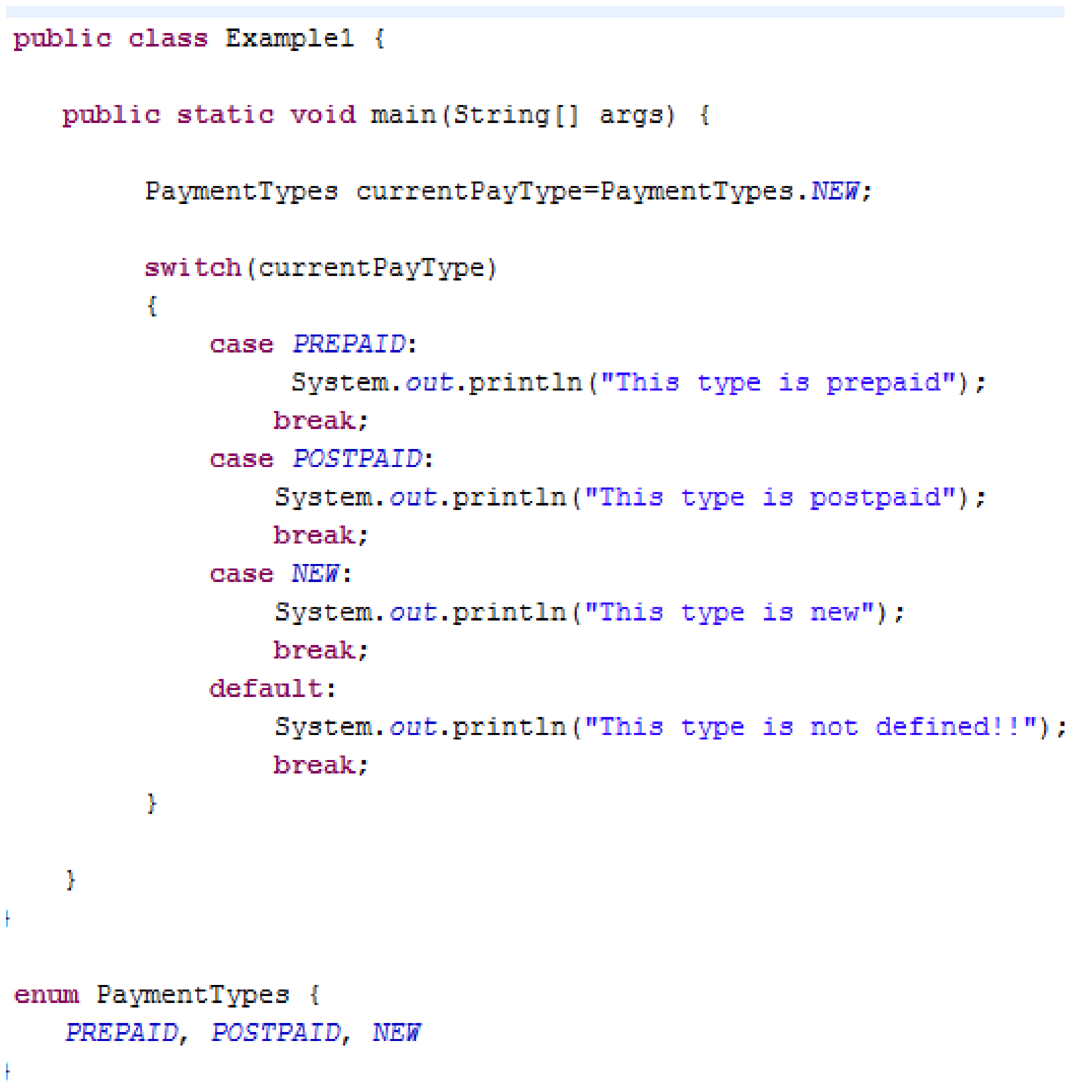
1.13.8 **Additional:**

Using CompareTo to compare two Enum Values:

The java.lang.Enum.compareTo() method compares this enum with the specified object for order.Enum constants are only comparable to other enum constants of the same enum type.

    public final int compareTo(E object1)

This method returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

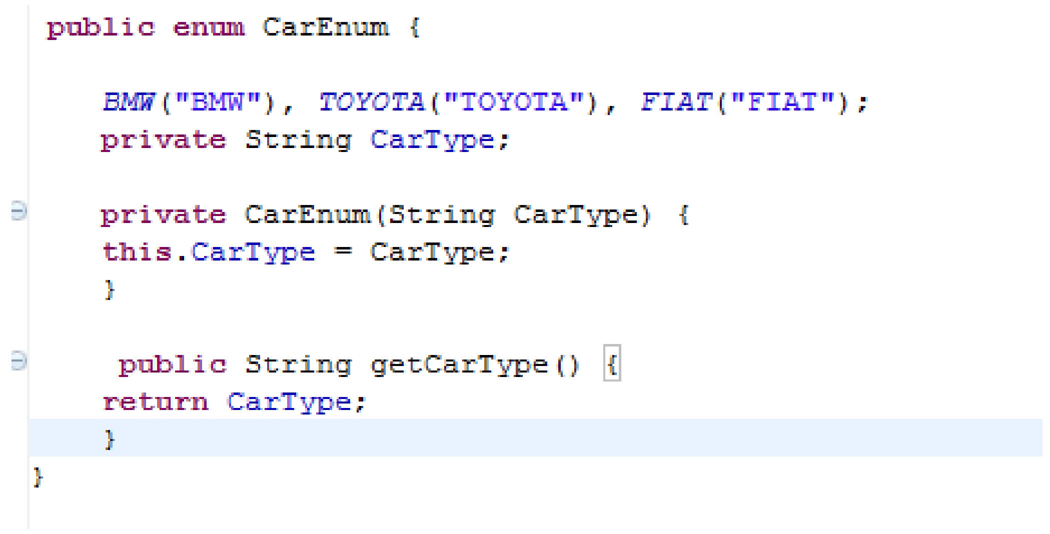


```
public class Example1 {

   public static void main(String[] args) {

        PaymentTypes currentPayType=PaymentTypes.NEW;

        switch(currentPayType)
        {
            case PREPAID:
                 System.out.println("This type is prepaid");
                break;
            case POSTPAID:
                System.out.println("This type is postpaid");
                break;
            case NEW:
                System.out.println("This type is new");
                break;
            default:
                System.out.println("This type is not defined!!");
                break;
        }

   }
}

enum PaymentTypes {
   PREPAID, POSTPAID, NEW
}
```

The output is : The two enum values are different and parameter is less ordered

Enum Type Constructor:

Note that the Enum's constructor has to be private , and you can't access it , this private constructor defines the constants listed in the beginning of the Enum body:



```
public enum CarEnum {

    BMW("BMW"), TOYOTA("TOYOTA"), FIAT("FIAT");
    private String CarType;

    private CarEnum(String CarType) {
    this.CarType = CarType;
    }

     public String getCarType() {
    return CarType;
    }
}
```

You may need to know how to access a value of Enum.For example the following code snippet shows you how to access car Enum in the main class:

```
public class EnumTest {
static CarEnum mycar;
   public static void main(String args[])
   {
      System.out.println(mycar.BMW.getCarType());
   }
}
```

To print all values of a specific Enum Type:

For each ENUM class type a values functions is auto generated which can be used to print all the ENUMs available.

```
public class Main {

    public static void main(String[] args) {
        // To print all the  values of enum type :
        for (CarEnum carType : CarEnum.values()) {
            System.out.println("Car type:- " + carType);
        }
    }
}
enum CarEnum {
    BMW("BMW"), TOYOTA("TOYOTA"), FIAT("FIAT");
    private String CarType;
    private CarEnum(String CarType) {
        this.CarType = CarType;
    }
    public String getCarType() {
        return CarType;
    }
}
```

Using valueOf function to return Enum:

Return enum constant type if a specific declared Enum constant identifier matches the string passed to valueOf() function.

```
public class Main {
    public static void main(String[] args) {

        System.out.print(CarEnum.valueOf("BMW"));
    }
}
 enum CarEnum {
    BMW("Bavarian Motor Works"),
    TOYOTA("Too Often Yankees Overprice This Auto"),
    FIAT("Fabbrica Italiana Automobili Torino");
    private String CarType;
    private CarEnum(String CarType) {
        this.CarType = CarType;
    }
    public String getCarType() {
        return CarType;
    }
}
```

The output of this program is :BMW

1.13.9 **Best Practices:**

- **You should use enum types any time you need to represent a fixed set of constants. That includes natural enum types such as the planets in our solar system and data sets where you know all possible values at compile time—for example, the choices on a menu, command line flags, and so on.**

1.13.10 **Take Away Points :**

- **Java 5 added an enum type to the language.**
- **enum can be traversed**
- **enum can have fields, constructors and methods**
- **enum may implement many interfaces but cannot extend any class because it internally extends Enum class**
- **Enumerated values are used to represent a set of named values.**
- **Historically in Java (and other languages), these were often stored as constants.**
- **In its simplest form, it contains a comma separated list of names representing each of the possible options.**

1.13.11 **Related Topics:**

- **Define Enum Type As Inner Variable (Inside a Class)**
- **Passing Enum as Type Parameter method**
- **Extending enum in java.**
- **java.util.EnumSet & java.util.EnumMap**

1.13.12 **References:**

http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html
http://howtodoinjava.com/2012/12/07/guide-for-understanding-enum-in-java/

Books for your Reference:

- ✓ **Thinking in Java by Bruce Eckel**
- ✓ **Head First Java by Bert Bates, Kathy Sierra**
- ✓ **Java 7 Programming Black Book by Dreamtech Pres**