//Program on enum :

```java
package com.ravi.enum_demo;

public class Test1
{
    public static void main(String[] args)
    {
        enum Month
        {
            JANUARY, FEBRUARY,MARCH     //public + static + final
        }

        System.out.println(Month.MARCH);
    }
}
```

Note : We can write an enum inside the method

```java
package com.ravi.enum_demo;

enum Month
{
    JANUARY,FEBRUARY,MARCH
}
public class Test2
{
    enum Color { RED,BLUE,BLACK }

    public static void main(String[] args)
    {
        enum Day {SUNDAY, MONDAY, TUESDAY }

        System.out.println(Month.FEBRUARY);
        System.out.println(Color.RED);
        System.out.println(Day.SUNDAY);
    }
}
```

Note : We can define an enum inside the class, outside of the class as well as inside the method also.

```java
//Comparing the constant of an enum

package com.ravi.enum_demo;

public class Test3
{
    enum Color { RED,BLUE }

     public static void main(String args[])
     {
          Color c1 = Color.RED;
          Color c2 = Color.RED;

          if(c1 == c2)
          {
                System.out.println("== Operator");
          }
          if(c1.equals(c2))
          {
                 System.out.println("equals method");
          }
      }
}
```

Note : enum constant, we can compare by using == operator and final equals(Object obj) method.

```java
package com.ravi.enum_demo;

public class Test4
{
    private enum Season   //private, public, protected, static
    {
    SPRING, SUMMER, WINTER, RAINY
    }

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

Note : If we define an enum inside the class then we can write private, private package, protected, public and static modifier.

```java
//Interview Question
package com.ravi.enum_demo;

interface Moveable
{
    void move();
}

class Hello
{
    int x = 100;
}


enum Direction implements Moveable
{
    EAST, WEST, NORTH, SOUTH;

    @Override
    public void move() {
        // TODO Auto-generated method stub

    }
}

class Test5
{
    public static void main(String[] args)
    {
        System.out.println(Direction.SOUTH);
    }
}
```

Note : An enum implicitly extends from java.lang.Enum abstract class so explicitly It **cannot extend** a class.

```java
//All enums are by default final so can't inherit

package com.ravi.enum_demo;

enum Pizza
{
    SMALL, MEDIUM, BIG;
}
class Test6 //extends Pizza
{
    public static void main(String[] args)
    {
        System.out.println(Pizza.BIG);
    }
}
```

Note : Every enum is implicitly final so a class cannot extend an enum

```java
//values() to get all the values of enum

package com.ravi.enum_demo;

public class Test7
{
    enum Season
    {
    SPRING, SUMMER, WINTER, FALL, RAINY
    }

    public static void main(String[] args)
    {
        Season[] seasons = Season.values();

        for(Season season : seasons)
        {
           System.out.println(season.toString());
           System.out.println(season.name());
           System.out.println("================");
        }
    }
}
```

Note : by using values() method we can fetch all the enum constants.

```java
//ordinal() to find out the order position

package com.ravi.enum_demo;

public class Test8
{
    enum Direction
    {
        EAST, WEST, NORTH, SOUTH
    }

    public static void main(String[] args)
    {
         Direction[] directions = Direction.values();

         for(Direction direction : directions)
         {
            System.out.println(direction.name() + " Order Poistion is :"+direction.ordinal());
         }
    }
}
```

Note : By using ordinal() we can find out the order position of an enum constant.

```java
//We can take main () inside an enum

package com.ravi.enum_demo;

enum Test9
{
    TEST1, TEST2, TEST3 ;

    public static void main(String[] args)
    {
        System.out.println("Enum  main method");
    }
}
```

Note : We can write main method inside an enum.

```java
//constant must be in first line of an enum

package com.ravi.enum_demo;


enum Test10
{
    public static void main(String[] args)
    {
       System.out.println("Enum  main method");
    }

    HR, SALESMAN, MANAGER;  //error [enum constant must be in the first line]

}
```

```java
//Writing constructor in enum

package com.ravi.enum_demo;

enum Season
{
    WINTER, SUMMER, SPRING, RAINY;   //public static final Season WINTER = new Season();
                                        //public static final Season SUMMER = new Season();
    private Season()
    {
       System.out.println("Constructor is executed....");
    }
}
class Test11
{
    public static void main(String[] args)
    {
       System.out.println(Season.WINTER);
       System.out.println(Season.SUMMER);

    }
}
```

Note : When enum class will be loaded then all the enum constants will be initialized with user value (enum object) so appropriate constrauctor will be invoked.

```java
//Writing constructor with message
package com.ravi.enum_demo;

   enum MySeason
   {
      SPRING("Pleasant"), SUMMER("UnPleasent"), RAINY("Rain"), WINTER;

        String msg;

        private MySeason(String msg)
        {
          this.msg = msg;
        }

      private MySeason()
      {
          this.msg = "Cold";
      }

      public String getMessage()
      {
          return this.msg;
      }
   }
class Test12
{
    public static void main(String[] args)
    {
       MySeason seasons[] = MySeason.values();

       for(MySeason season : seasons)
       {
          System.out.println(season+"  is :"+season.getMessage());
          System.out.println(season+"  order is :"+season.ordinal());
          System.out.println("Season name is :"+season.name());
          System.out.println("============================");
       }
    }
}
```

How to take enum constant from user (Sacnner class) :
----

public static T valueOf(Enum.class, String constantName) :
----
* It is a predefined method of java.lang.Enum class
* It accept two parameters
  a) Enum.class fiile : It will load enum .class file and search the constantName in this .class file which is provided by in the 2nd parameter. It is case sensitive so provided constantName and available constant name both must be exactly same (toUpperCase()).

  b) String constantName : Here we will pass the constantName which we want to search in the Enum.class file. It internally uses loop to search enum constant, If enum constant is not available then It will generate an Exception i.e java.lang.IllegalArgumentException.

```java
package com.ravi.enum_demo;

import java.util.Scanner;

enum Color
{

    RED, BLUE, PINK
}

public class Test13
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Color Name :");
        String colorName = sc.next().toUpperCase();

        Color color = Color.valueOf(Color.class, colorName);
        System.out.println("Color name is :"+color);
        sc.close();
    }
}
```

** In enum why static block is **always executed after constructor ?**
----
* In an enum, static blocks are always executed after the constructor because the first line of an enum is reserved for enum constant so at the time of loading the enum.class file these constant are always executed in the first line so static block will be always executed after the constructor body.

```java
package com.ravi.enum_demo;

enum MyColor
{
   RED , BLUE, PINK;

   static
   {
        System.out.println("static block ");
    }

   private MyColor()
    {
        System.out.println("Constructor");
    }

}
public class Test14
{
  public static void main(String[] args)
   {
     System.out.println(MyColor.RED);
   }
}
```

Writing switch case with enum :
----
* Recently some new enhancement came in switch case from JDK 13, 14 and JDK 17V

```java
package com.ravi.enum_demo;

import java.util.Scanner;

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

public class Test15
{

public static void main(String args[])
{
   Scanner sc = new Scanner(System.in);
   System.out.println("Enter any Day name from the Week :");
   String dayOfWeek = sc.nextLine().toUpperCase();

   Day day = Day.valueOf(Day.class, dayOfWeek);

   switch(day)
   {
    case SUNDAY ->
    {
                  System.out.println("Today is Sunday");
                  System.out.println("Enjoy Sunday!!!");
    }

    case MONDAY -> System.out.println("Today is Monday");
    case TUESDAY -> System.out.println("Today is Tuesday");
    case WEDNESDAY -> System.out.println("Today is Wednesday");
    case THURSDAY -> System.out.println("Today is Thursday");
    case FRIDAY -> System.out.println("Today is Friday");
    case SATURDAY -> System.out.println("Today is Saturday");
    default -> System.out.println("Not a valid day");

    }


   sc.close();
  }
}
```