EJERCICIOS JAVA ENUMERADOS

¿Qué es un ENUM en Java? Los enum en Java se usan cuando conocemos todos los valores posibles en tiempo de compilación.

La declaración de Enum puede hacerse fuera de una clase, o dentro de una clase (class), pero NO dentro de un método.

```
// Un simple ejemplo donde se declara enum
// fuera de cualquier clase (Nota la palabra enum en lugar // de la palabra class)
enum Color
{
        ROJO, VERDE, AZUL;
}
public class Test
{
// El método
  public static void main(String[] args)
  {
        Color c1 = Color.ROJO;
        System.out.println(c1);
        c1 = Color.VERDE;
        System.out.println(c1);
 }
}
```

El tipo enum se puede pasar como un argumento para switch.

```
enum Dia
{
 LUNES, MARTES, MIERCOLES, JUEVES,
 VIERNES, SABADO, DOMINGO;
}


public class Test
{
 // Metodo
 public static void main(String[] args)
 {
         Dia dia = Dia.LUNES;

    switch (dia)
    {
    case LUNES:
       System.out.println("Los lunes son feos.");
       break;
    case VIERNES:
       System.out.println("Los viernes son mejores.");
       break;
    case SABADO:
    case DOMINGO:
       System.out.println("Los fines de semana son mejores.");
       break;
    default:
       System.out.println("Los días entre semana son regulares.");
       break;
    }
 }
}
```

Todas las enumeraciones tienen automáticamente dos métodos predefinidos: values() y valueOf(). Sus formas generales se muestran aquí:

public static tipo-enum[ ] values( )
public static tipo-enum valueOf(String str)

Estos métodos están presentes dentro de java.lang.Enum.
El método values() se puede usar para devolver todos los valores presentes dentro de enum.
El orden es importante en las enumeraciones. Al usar el método ordinal(), se puede encontrar cada índice de la constante enum, al igual que el índice de matriz.
El método valueOf() devuelve la constante enum del valor de cadena especificado, si existe.

```
enum Color
{
   ROJO, VERDE, AZUL;
}

public class Test
{
   public static void main(String[] args)
   {
      // Llamando a values()
      Color arr[] = Color.values();

      // enum con bucle
      for (Color col : arr)
      {
         // Llamando a ordinal() para encontrar el índice  de color.
         System.out.println(col + " en el índice "  + col.ordinal());
      }

      // Usando valueOf(). Devuelve un objeto de  Color con la constante dada.
      // La segunda línea comentada causa la excepción         // IllegalArgumentException
      System.out.println(Color.valueOf("ROJO"));
      // System.out.println(Color.valueOf("BLANCO"));
   }
}
```

```
package pruebas;

import java.util.Scanner;

enum Transporte{
   COCHE(60), CAMION(50), AVION(600), TREN(70), BARCO(20);
         private int velocidad; //velocidad típica de cada transporte
         //Añadir un Constructor
         Transporte(int s){velocidad=s;}
         //Añadir un método
         public  int getVelocidad(){return velocidad;}
}
```

Es importante comprender que cada constante de enumeración es un objeto de su tipo de enumeración. Por lo tanto, una enumeración puede definir constructores, agregar métodos y tener variables de instancia.

```
public class Test
```

```
{

   public static void main(String[] args)
   {
        Scanner teclado = new Scanner(System.in);
        // Mostrar la velocidad de un transporte
        String medio;
        System.out.println("Introduzca un medio de transporte");
        medio = teclado.nextLine();
        try {
             System.out.println("La velocidad del " + medio + " es:" + Transporte.valueOf(medio).getVelocidad());

        }catch(Exception e) {
             System.out.println("No existe el " + medio);
        }
   }
}
```