Ejercicio 1: búsqueda del buey

Número de respuestas: 5
Búsqueda del buey (OX en inglés).

Crea un programa que rellene **un array de 20 caracteres** (no cadenas), rellenado con caracteres aleatorios. Los caracteres a elegir son 'O', 'X', 'L' y 'B'. Con estos cuatro caracteres se deberá rellenar el array, **conformando un array distinto para cada ejecución**. Un ejemplo de array generado podría ser:

[B, B, B, X, L, X, L, L, O, X, O, L, O, O, B, O, B, X, O, L]

Después se deberá buscar si dentro de dicho array de 20 caracteres aparecen, en posiciones consecutivas los símbolos O y X. Solo hay que buscar la primera aparición (si la hubiese). En el caso del ejemplo anterior estarían en la posición 8:

[B, B, B, X, L, X, L, L, **O, X**, O, L, O, O, B, O, B, X, O, L]

Para la búsqueda no se puede invocar a ningún método de ninguna clase (ni convertir el array a un String). Como resultado de la ejecución, el programa debería mostrar algo así:

[B, B, B, X, L, X, L, L, O, X, O, L, O, O, B, O, B, X, O, L]

OX encontrado en posición 8

O bien, si no aparece la secuencia OX:

[L, L, B, X, B, B, X, L, L, X, B, X, O, L, X, L, X, X, X, L]

OX no encontrado

SOLUCIÓN INMA:

```
*/
package ejercicios_entrenamiento_programacion_febrero;

import java.util.Arrays;
import java.util.Random;

public class Ejercicio1 {

    public static void main(String[] args) {
        //creamos un array con los caracteres a coger de forma aleatoria
        char[] arrayLetras = {'O', 'X', 'L', 'B'};
        //creamos un array de salida
        char[] arraySalida = new char[20];

        //creamos un objeto Random para luego utilizarlo en la eleccion de caracteres aleatorios creandolos por su
posición en nuestro array de entrada
        Random letra = new Random();

        //creamos un recorrido de 20 creaciones de letras aleatorias que irán a nuestro array de salida
        for (int i = 0; i < 20; i++) {

            int letraAleatoria = letra.nextInt(4);
            arraySalida[i] = arrayLetras[letraAleatoria];

        }
```

```
        System.out.printf(Arrays.toString(arraySalida));
        System.out.println("\n");

        //variable para la posición que ocupa O si una X es encontrada justo después de ella
        int posicion = 0;

        //variable para el mensaje de salida por pantalla
        String mensajeSalida = "";

        //variable para saber si se ha encontrado OX
        boolean encontrado = false;

        //recorremos el array de salida
        for (int i = 0; i < arraySalida.length; i++) {
           //si encuentra OX en su recorrido extraeremos en qué posición
           if ((arraySalida[i] == 'O') && ((i + 1 <= arraySalida.length-1) && (arraySalida[i + 1] == 'X'))) {
              posicion = i;
              encontrado = true;
           }
        }

        if (encontrado) {
           mensajeSalida = "OX encontrado en posción" + posicion;
                 System.out.printf(mensajeSalida);

        } else {
           mensajeSalida = "OX no encontrado";
                 System.out.printf(mensajeSalida);
        }
     }
}
```

Ejercicio 2: conversor de nombres de números

Número de respuestas: 2

Dado el siguiente programa que ya tiene implementada la entrada de datos y la salida de resultados, escribir el código necesario en Java para que se procese el array de entrada con números del 0 al 99 y se rellene un array de salida con los nombres en español de esos números.

```
public class NumeroAPalabra99 {

    /**
     * Programa principal
     *
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        //------------------------------------------------
        //          Declaración de variables
        //------------------------------------------------
        // Variables de entrada
        int[] arrayEntrada = {0, 99, 10, 20, 15, 25, 66, 11, 7, 90, 72};

        // Variables de salida
```

```
        String[] arrayResultado;

        // Variables auxiliares
        String[][] arrayNombresNumeros = {
            {"cero", "uno", "dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho",
"nueve"},
            {"diez", "once", "doce", "trece", "catorce", "quince", "dieciséis", "diecisiete",
"dieciocho", "diecinueve"},
            {"veinte", "veintiuno", "veintidós", "veintitrés", "veinticuatro", "veinticinco",
"veintiséis", "veintisiete", "veintiocho", "veintinueve"},
            {"treinta"},
            {"cuarenta"},
            {"cincuenta"},
            {"sesenta"},
            {"setenta"},
            {"ochenta"},
            {"noventa"}
        };
        int contador;

        //-----------------------------------------------
        //                Entrada de datos
        //-----------------------------------------------
        System.out.println("CONVERSOR DE NOMBRES DE NÚMEROS");
        System.out.println("------------------------------");
        System.out.println("Lista de números de prueba:");
        System.out.println(Arrays.toString(arrayEntrada));

        //-----------------------------------------------
        //  Procesamiento (a implementar por el alumnado)
        //-----------------------------------------------


        //-----------------------------------------------
        //              Salida de resultados
        //-----------------------------------------------
        System.out.println();
        System.out.println("RESULTADO");
        System.out.println("---------");
        System.out.printf("Los nombres de esos números son:\n%s\n",
                Arrays.toString(arrayResultado));

    }

}
```

El resultado que debe proporcionar el programa es el siguiente:

CONVERSOR DE NOMBRES DE NÚMEROS

------------------------------ -

Lista de números de prueba:

[0, 99, 10, 20, 15, 25, 66, 11, 7, 90, 72]

RESULTADO

Los nombres de esos números son:

[cero, noventa y nueve, diez, veinte, quince, veinticinco, sesenta y seis, once, siete, noventa, setenta y dos]

Tan solo hay que implementar la parte del procesamiento así como declarar y utilizar todas las variables auxiliares que consideres oportunas. La entrada de datos y la salida de resultados están ya hechas y no hay que tocar nada.

En el procesamiento que implementes podrás hacer uso del array bidimensional arrayNombresNumeros que contiene los textos de los números necesarios para componer cada uno de los nombres de los números en castellano entre 0 y 99.

Debes aprovechar ese array para componer los cien posibles números:

1. Si el número es menor que 30, dispondrás directamente del nombre del número en el array arrayNombresNumeros.
2. partir de 30, tendrás que componer el nombre usando decena y unidad. Recuerda que puedes obtener la decena de un número mediante la división entera y su unidad mediante el módulo o resto (el texto "y xxx", donde "xxx" sería la unidad, por ejemplo *"setenta y dos"* para 72). Y no olvides que si el número es una decena "pura" (por ejemplo 90), entonces no debes añadir la unidad (tan solo sería "*noventa*").

SOLUCIÓN INMA:

```
package ejercicios_entrenamiento_programacion_febrero;

import java.util.Arrays;

public class Ejercicio2 {
   public static void main(String[] args) {

      //---------------------------------------------
      //         Declaración de variables
      //---------------------------------------------
      // Variables de entrada
      int[] arrayEntrada = {0, 99, 10, 20, 15, 25, 66, 11, 7, 90, 72};

      // Variables de salida
      String[] arrayResultado;

      // Variables auxiliares
      String[][] arrayNombresNumeros = {
         {"cero", "uno", "dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho", "nueve"},
         {"diez", "once", "doce", "trece", "catorce", "quince", "dieciséis", "diecisiete", "dieciocho", "diecinueve"},
         {"veinte", "veintiuno", "veintidós", "veintitrés", "veinticuatro", "veinticinco", "veintiséis", "veintisiete",
"veintiocho", "veintinueve"},
         {"treinta"},
         {"cuarenta"},
         {"cincuenta"},
         {"sesenta"},
         {"setenta"},
         {"ochenta"},
         {"noventa"}
      };
      int contador;

      //---------------------------------------------
      //              Entrada de datos
      //---------------------------------------------
```

```
        System.out.println("CONVERSOR DE NOMBRES DE NÚMEROS");
        System.out.println("-----------------------------");
        System.out.println("Lista de números de prueba:");
        System.out.println(Arrays.toString(arrayEntrada));

        //---------------------------------------------
        //  Procesamiento (a implementar por el alumnado)
        //---------------------------------------------



        //---------------------------------------------
        //              Salida de resultados
        //---------------------------------------------
        System.out.println();
        System.out.println("RESULTADO");
        System.out.println("---------");
        System.out.printf("Los nombres de esos números son:\n%s\n",
                Arrays.toString(arrayResultado));

}
*/

package ejercicios_entrenamiento_programacion_febrero;

import java.util.Arrays;

        public class Ejercicio2{

    public static void main(String[] args) {

        //---------------------------------------------
        //          Declaración de variables
        //---------------------------------------------
        // Variables de entrada
        int[] arrayEntrada = {0, 99, 10, 20, 15, 25, 66, 11, 7, 90, 72};

        // Variables de salida
        String[] arrayResultado = new String [arrayEntrada.length];

        // Variables auxiliares
        String[][] arrayNombresNumeros = {
            {"cero", "uno", "dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho", "nueve"},
            {"diez", "once", "doce", "trece", "catorce", "quince", "dieciséis", "diecisiete", "dieciocho", "diecinueve"},
            {"veinte", "veintiuno", "veintidós", "veintitrés", "veinticuatro", "veinticinco", "veintiséis", "veintisiete",
"veintiocho", "veintinueve"},
            {"treinta"},
            {"cuarenta"},
            {"cincuenta"},
            {"sesenta"},
            {"setenta"},
            {"ochenta"},
            {"noventa"}
        };



        //---------------------------------------------
        //               Entrada de datos
        //---------------------------------------------
```

```
        System.out.println("CONVERSOR DE NOMBRES DE NÚMEROS");
        System.out.println("-----------------------------");
        System.out.println("Lista de números de prueba:");
        System.out.println(Arrays.toString(arrayEntrada));

        //-------------------------------------------
        //  Procesamiento (a implementar por el alumnado)
        //-------------------------------------------
        for(int i=0; i<arrayEntrada.length; i++){
           //si el numero tiene una unidad (del 0 al 9)
           if(arrayEntrada[i]<10){
              arrayResultado[i]=arrayNombresNumeros[0][arrayEntrada[i]];
           }else if(arrayEntrada[i]>=10 && arrayEntrada[i] < 20){
              arrayResultado[i]=arrayNombresNumeros[1][arrayEntrada[i]%10];
           }else if(arrayEntrada[i]>=20 && arrayEntrada[i] < 30){
              arrayResultado[i]=arrayNombresNumeros[2][arrayEntrada[i]%10];
           }else {
              if(arrayEntrada[i]%10==0){
                 arrayResultado[i]=arrayNombresNumeros[arrayEntrada[i]/10][0];
              }else {
                 arrayResultado[i]=arrayNombresNumeros[arrayEntrada[i]/10][0] + " y " +
arrayNombresNumeros[0][arrayEntrada[i]%10];
              }
           }
        }

        //-------------------------------------------
        //              Salida de resultados
        //-------------------------------------------
        System.out.println();
        System.out.println("RESULTADO");
        System.out.println("---------");
        System.out.printf("Los nombres de esos números son:\n%s\n",
              Arrays.toString(arrayResultado));

    }
        }
```

Ejercicio 3: Implementación de la clase Baraja

Número de respuestas: 0

Implementa una clase `Baraja` que represente a una baraja de naipes con las siguientes propiedades y comportamiento:

1. Un objeto de la clase `Baraja` podrá tener 40 o 48 naipes de la baraja española. Los únicos atributos que necesitas para representar un objeto de **tipo** de baraja serán el **número de naipes** que tiene (40 ó 48) y el **número de extracciones de naipes** que se han realizado desde que se creó.
2. Se dispondrá de **dos constructores**:
    2.1 Un **constructor con un parámetro**, donde se indicará el **número de naipes** de la baraja (un número entero). Si el número de naipes es inadecuado, se lanzará una excepción de tipo `IllegalArgumentException`.

2.2 Un **constructor sin parámetros**, donde se tomará **40** como el **número de naipes por omisión**.

3 Los métodos `getNumNaipes` y `getNumExtracciones`, que devolverán, respectivamente, el número de naipes de los que dispone la baraja (número entero) y el número de naipes que se han sacado de la baraja desde que se creó.

4 Un método `extraerNaipeRandom`, que devolverá un naipe aleatorio de la baraja. La forma de representar un naipe será un String donde se indique el valor del naipe con el formato "<NUMERO> de <PALO>", donde <NUMERO> será una cadena que podrá valer "UNO", "DOS", "TRES", etc. o bien "SOTA", "CABALLO" o "REY". Y <PALO> será una cadena que podrá valer "OROS", "ESPADAS", "COPAS", "BASTOS". Algunos ejemplos de lo que devolvería este método podrían ser: "CINCO de ESPADAS", "OCHO de OROS" o "SOTA de BASTOS". Cuando se extrae un naipe de la baraja, se vuelve a meter dentro. Por tanto un mismo naipe puede salir varias veces repetido en sucesivas extracciones.

5 Un método `toString`, que devuelva una cadena del tipo "Baraja de xxx naipes. Extracciones: zzz", donde xxx será el número de naipes de la baraja y zzz la cantidad de extracciones que se han realizado hasta el momento.

6 En la misma clase `Baraja`, crea un método `main()` donde:

6.1 Se instancie una baraja con el **constructor sin parámetros** y se muestren sus datos usando `toString()`.

6.2 Se solicite por teclado un número de naipes y a continuación se intente instanciar una baraja con ese valor de estado inicial:

2.1 Si el valor es válido, se mostrarán los atributos de la baraja usando los métodos `getNumNaipes` y `toString`. A continuación se extraerán diez naipes consecutivamente (bucle `for`) mostrando el valor de cada naipe extraído. Finalmente se volverá a mostrar el estado de la baraja (llamada al método `toString`). Una vez hecho esto se volverá a solicitar un nuevo número de naipes.

2.2 Si el valor no es válido (un número que el constructor no acepte o un valor no entero), entonces se capturará la excepción correspondiente (`IllegalArgumentException` o `InputMismatchException`) y se finalizará el programa.

No es obligatorio incluir comentarios javadoc para documentar la clase.

Aquí tienes un ejemplo de la salida por pantalla tras una ejecución de prueba:

```
BARAJAS

-------

Prueba del constructor sin parámetros:

Objeto creado: Baraja de 40 naipes. Extracciones: 0

Prueba del constructor con parámetros:

Introduzca el número de naipes (40 o 48): 40

Prueba del getNumNaipes: 40

Prueba del toString: Baraja de 40 naipes. Extracciones: 0

Pruebas de extracción:

REY de BASTOS

SIETE de OROS

SOTA de COPAS

REY de ESPADAS
```

CINCO de OROS

CINCO de ESPADAS

SOTA de BASTOS

CINCO de BASTOS

DOS de COPAS

CINCO de ESPADAS

Prueba del toString: Baraja de 40 naipes. Extracciones: 10

Introduzca el número de naipes (40 o 48): 41

Error. Número de naipes inválido: 0. Finalizamos

SOLUCIÓN INMA:

```
 */
package ejercicios_entrenamiento_programacion_febrero;

import java.util.Random;
import java.util.Scanner;

public class Baraja {

   final int naipesPorDefecto = 40;
   private int numNaipes;
   private int numExtracciones=0;
   private Random numRandom = new Random();

   public Baraja(int numCartas) throws IllegalArgumentException {

      if (numCartas != 40 && numCartas != 48) {
         throw new IllegalArgumentException(String.format("Numero de naipes inválido: %s", numCartas));
      } else {
         this.numNaipes = numCartas;
      }
   }

   public Baraja() {
      this.numNaipes = this.naipesPorDefecto;
   }

   public int getNumNaipes() {
      return numNaipes;
   }

   public int getNumExtracciones() {
      return numExtracciones;
   }
```

```
   String[] numerosCartas = {"UNO", "DOS", "TRES", "CUATRO", "CINCO", "SEIS", "SIETE", "SOTA", "CABALLO", "REY"};

   String[] palo = {"OROS", "COPAS", "ESPADAS", "BASTOS"};

   public String extraerNaipeRandom() {

      int numeroAleatorio = numRandom.nextInt(numerosCartas.length);//(int) Math.random() *
numerosCartas.length + 1;
      int paloAleatorio = numRandom.nextInt(palo.length);//(int) Math.random() * palo.length + 1;
      numExtracciones++;
      return numerosCartas[numeroAleatorio] + " de " + palo[paloAleatorio];

   }

   public String toString() {
      return "Baraja de " + numNaipes + " naipes. "
            + "Extracciones: " + numExtracciones;
   }

   public static void main(String[] args) {
      System.out.print("Prueba del constructor sin parámetros:");
      Baraja baraja = new Baraja();
      System.out.print("Objeto creado: ");
      System.out.println(baraja.toString());

      Scanner teclado = new Scanner(System.in);
      System.out.print("Introduzca el número de naipes (40 o 48):");
      int numNaipes = teclado.nextInt();
      System.out.println("Prueba del constructor con parámetros:");
      try {
         Baraja baraja2 = new Baraja(numNaipes);
         System.out.print("Prueba del getNumNaipes:");
         System.out.println(baraja2.getNumNaipes());
         System.out.print("Prueba del toString:");
         System.out.println(baraja2.toString());
         System.out.println("Pruebas de extracción:");
         for (int i = 0; i < 10; i++) {
            System.out.println(baraja2.extraerNaipeRandom());
         }
         System.out.print("Prueba del toString:");
         System.out.println(baraja2.toString());
         System.out.print("Introduzca el número de naipes (40 o 48):");
         numNaipes = teclado.nextInt();
         Baraja baraj3 = new Baraja(numNaipes);
      } catch (IllegalArgumentException e) {
         System.out.print("Error: "+e.getMessage());
      }

   }

}
```

Ejercicio 4: Implementación de la clase CajaCambios

Número de respuestas: 0

Implementa una clase `CajaCambios` que represente a una caja de cambios de las marchas de un vehículo con las siguientes propiedades y comportamiento:

1. Un objeto de la clase `CajaCambios` podrá tener entre tres y seis marchas. Los únicos atributos que necesitas para representar un objeto de tipo caja de cambios serán el número de marchas que tiene (entre tres y seis) y la **marcha actual** que hay metida.
2. Se dispondrá de **dos constructores** :
   - 2.1 Un **constructor con un parámetro** , donde se indicará el **número de marchas** de la caja (un número entero). Si el número de marchas es inadecuado, se lanzará una excepción de tipo `IllegalArgumentException` . Toda caja de cambios recién creada tendrá metida la primera marcha.
   - 2.2 Un **constructor sin parámetros** , donde se tomará **5** como el **número de marchas por omisión** .
3. Los métodos `getNumMarchas` y `getEstado` , que devolverán, respectivamente, el número de marchas de los que disponga la caja de cambios (número entero) y la marcha que hay metida en ese momento (una cadena de caracteres con el valor “PRIMERA”, “SEGUNDA”, “TERCERA”, etc.). Un método `subirMarcha` , que hará que se incremente una marcha en la caja de cambios. Si se está en la marcha más alta y por tanto no se puede subir más, se lanzará una excepción del tipo `IllegalStateException` con un mensaje de error apropiado. Si se ha podido cambiar de marcha con éxito, se devolverá un `String` con la marcha actual.Un método `bajarMarcha`, que hará que se decremente una marcha en la caja de cambios. Si se está en la marcha más baja y por tanto no se puede bajar más, se lanzará una excepción del tipo `IllegalStateException` con un mensaje de error apropiado. Si se ha podido cambiar de marcha con éxito, se devolverá un `String` con la marcha actual.
4. Un método `toString` , que devuelva una cadena del tipo “Caja de cambios de xxx marchas. Actualmente en zzz ”, donde xxx será el número de marchas de la caja y zzz la marcha metida actualmente.
5. En la misma clase `CajaCambios` , crea un `métodomain()` donde:
   - 5.1 Se instancie una caja de cambios con el **constructor sin parámetros** y se muestren sus datos usando `toString()`.
   - 5.2 Se solicite por teclado un número de marchas y a continuación se intente instanciar una caja de cambios con esa cantidad:
     - 2.1 Si el valor es válido, se mostrarán los atributos de la caja de cambios usando los métodos `getNumMarchas`, `getEstado` y `toString`. A continuación se irá subiendo de marcha hasta que no se pueda más y salte la excepción. Para cada cambio de marcha se irá indicando el estado de la caja mostrando el resultado de la llamada al método `toString`. Una vez superado el tope máximo se irá bajando de marcha de la misma manera hasta que no se pueda más y también salte una excepción. Nuevamente para cada cambio de marcha también se irá mostrando el resultado de la ejecución del método `toString`. Una vez hecho esto se volverá a solicitar un nuevo número de marchas.
     - 2.2 Si el valor no es válido (un número que el constructor no acepte o un valor no entero), entonces se capturará la excepción correspondiente (`IllegalArgumentException` o `InputMismatchException`) y se finalizará el programa.

No es obligatorio incluir comentarios javadoc para documentar la clase.

Aquí tienes un ejemplo de la salida por pantalla tras una ejecución de prueba:

CAJA DE CAMBIOS

Prueba del constructor sin parámetros:

Objeto creado: Caja de cambios de 5 marchas. Actualmente en PRIMERA

Prueba del constructor con parámetros:

Introduzca el número de marchas (4-6): 4

Prueba del getEstado: PRIMERA

Prueba del toString: Caja de cambios de 4 marchas. Actualmente en PRIMERA

Pruebas de cambio (subir):

Subiendo: SEGUNDA

Subiendo: TERCERA

Subiendo: CUARTA

Subiendo: Error. Marcha más alta alcanzada. No se puede subir más

Pruebas de cambio (bajar):

Bajando: TERCERA

Bajando: SEGUNDA

Bajando: PRIMERA

Bajando: Error. Marcha más baja alcanzada. No se puede bajar más

Introduzca el número de marchas (4-6): 3

Error. Número de marchas inválido: 3. Finalizamos

SOLUCIÓN INMA:

```
*/
package ejercicios_entrenamiento_programacion_febrero;

import java.util.Arrays;
import java.util.Scanner;

/**
 *
 * @author inmar
 */
public class CajaCambios {

   private int numMarchas;
   String[] marchasPosibles = {"PRIMERA", "SEGUNDA", "TERCERA", "CUARTA", "QUINTA", "SEXTA"};
```

```
   String marchaActual;

   public CajaCambios(int numMarchas) throws IllegalArgumentException {
      if (numMarchas >= 3 && numMarchas <= 6) {
         this.numMarchas = numMarchas;
         this.marchaActual = this.marchasPosibles[0];

      } else {
         throw new IllegalArgumentException(String.format("Error: número de marchas inválida: %s",
this.numMarchas));
      }

   }

   public CajaCambios() {
      this.numMarchas = 5;
   }

   public int getNumMarchas() {
      return this.numMarchas;
   }

   public String getEstado() {
      return this.marchaActual;
   }

   public String subirMarcha()  {
      int index = Arrays.asList(marchasPosibles).indexOf(this.marchaActual);

      if (this.marchaActual.equals(this.marchasPosibles[numMarchas-1])) {
         return "Error: marcha máxima alcanzada: "+ this.marchaActual;
      } else {
         this.marchaActual=marchasPosibles[index + 1];
         return marchasPosibles[index + 1];
      }

   }

   public String bajarMarcha() {
      int index = Arrays.asList(marchasPosibles).indexOf(this.marchaActual);
      if (this.marchaActual.equals(this.marchasPosibles[0])) {
         return "Error: marcha minima alcanzada: "+ this.marchaActual;
      } else {
         this.marchaActual=marchasPosibles[index - 1];
         return marchasPosibles[index - 1];
      }
   }

   public String toString() {
      return "Objeto creado: Caja de cambios de " + getNumMarchas() + " marchas.  Actualmente en " +
this.marchaActual;

   }
```

```
   public static void main(String[] args) {
      int numMarchas = 0;
      Scanner teclado = new Scanner(System.in);

      System.out.println("CAJA DE CAMBIOS");
      System.out.println("---------------");
      System.out.println("Prueba del constructor sin parámetros:");
      CajaCambios cajadeCambio = new CajaCambios();
      System.out.println(cajadeCambio.toString());

      System.out.println("Prueba del constructor con parámetros:");
      do {
         System.out.print("Introduzca el número de marchas (4-6):");
         try {
            numMarchas = teclado.nextInt();
            CajaCambios cajadeCambio2 = new CajaCambios(numMarchas);

            System.out.println("Prueba del getEstado: " + cajadeCambio2.getEstado());
            System.out.println("Prueba del toString: " + cajadeCambio2.toString());
            System.out.println("Pruebas de cambio (subir):");
            System.out.println("Subiendo " + cajadeCambio2.subirMarcha());
            System.out.println("Subiendo " + cajadeCambio2.subirMarcha());
            System.out.println("Subiendo " + cajadeCambio2.subirMarcha());
            System.out.println("Subiendo " + cajadeCambio2.subirMarcha());

            System.out.println("Pruebas de cambio (bajar):");
            System.out.println("Bajando " + cajadeCambio2.bajarMarcha());
            System.out.println("Bajando " + cajadeCambio2.bajarMarcha());
            System.out.println("Bajando " + cajadeCambio2.bajarMarcha());
            System.out.println("Bajando " + cajadeCambio2.bajarMarcha());

         } catch (IllegalStateException e) {
            System.out.println(e.getMessage());
         }
      } while (numMarchas >= 3 && numMarchas <= 6);
   }
}
```

SOLUCIÓN Dark-Dream:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package practicas;

import java.util.InputMismatchException;
import java.util.Scanner;
```

```
/**
 *
 * @author Jorge
 */
//Ejercicio 4
public class CajaCambios {

   private final static String[] marchas = {"PRIMERA", "SEGUNDA", "TERCERA", "QUARTA", "QUINTA", "SEXTA"};

   public final static int CANTIDAD_MARCHAS_MIN = 3;
   public final static int CANTIDAD_MARCHAS_MAX = 6;
   public final static int DEFAULT_MARCHAS = 5;

   private final int cantidadMarchas;
   private int marchaActual;

   public CajaCambios(int cantidadMarchas) throws IllegalArgumentException {
      if (cantidadMarchas < CajaCambios.CANTIDAD_MARCHAS_MIN || cantidadMarchas >
CajaCambios.CANTIDAD_MARCHAS_MAX) {
         throw new IllegalArgumentException("Cantidad de marchas invalidas");
      }

      this.cantidadMarchas = cantidadMarchas;
      this.marchaActual = 1;

   }

   public CajaCambios() {
      this(CajaCambios.DEFAULT_MARCHAS);
   }

   public int getCantidadMarchas() {
      return cantidadMarchas;
   }

   public String getMarchaActual() {
      return marchas[this.marchaActual - 1];
   }

   public String subirMarcha() throws IllegalStateException {
      if (this.marchaActual >= this.cantidadMarchas) {
         throw new IllegalStateException("Ya estas en la ultima marcha");
      }

      this.marchaActual++;

      return String.format("Se subio a %s correctamente", CajaCambios.marchas[this.marchaActual - 1]);

   }

   public String bajarMarcha() throws IllegalStateException {
      if (this.marchaActual <= 1) {
         throw new IllegalStateException("Ya estas en la primera marcha");
```

```
        }

        this.marchaActual--;

        return String.format("Se bajo a %s correctamente", CajaCambios.marchas[this.marchaActual - 1]);

    }

    @Override
    public String toString() {
        return String.format("Caja de cambios de %d marchas. Actualmente en %s", this.cantidadMarchas,
CajaCambios.marchas[this.marchaActual - 1]);
    }

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        int num;
        CajaCambios caja = new CajaCambios();
        CajaCambios caja2 = null;
        System.out.println(caja);

        System.out.print("Numero de marchas: ");
        try {
            num = teclado.nextInt();
            caja2 = new CajaCambios(num);
            System.out.println(caja2.getCantidadMarchas());;
            System.out.println(caja2.getMarchaActual());
            System.out.println(caja2);
            try {
                while (true) {
                    System.out.println(caja2.subirMarcha());
                }
            } catch (IllegalStateException e) {
                System.out.println(e.getMessage());
            }
            try {
                while (true) {
                    System.out.println(caja2.bajarMarcha());
                }
            } catch (IllegalStateException e) {
                System.out.println(e.getMessage());
            }

        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        } catch (InputMismatchException e) {
            System.out.println("No es un numero entero");
        }

    }

}
```

Ejercicio 5: Exploración numérica

Número de respuestas: 0
Dados dos arrays de 10 elementos que se os proporcionan, escribir un programa en Java que cree un tercer array de tamaño el doble de los anteriores y con el siguiente contenido para su primera mitad:

- En cada posición impar se almacenará la suma de todos los elementos del primer array hasta esa posición.
- En cada posición par se almacenará la suma de todos los elementos del segundo array hasta esa posición.

Para la segunda mitad:

- En cada posición impar se almacenará el doble de la posición impar anterior de ese mismo array.
- En cada posición par se almacenará la suma de las dos posiciones pares anteriores de ese mismo array

NOTA: consideramos la posición 0 como par .

Por último, debe proporcionarse la suma de todos los números impares del array resultado.

Dado que los arrays se proporcionan y, por tanto, no hay entrada de datos, el resultado debería ser similar al siguiente:

```
EJERCICIO DE ARRAYS
-------------------
Array 1:         [1, 8, 3, 1, 3, 7, 5, 2, 4, 6]
Array 2:         [3, 4, 5, 6, 2, 5, 1, 7, 5, 5]

RESULTADO
---------
Array resultado: [3, 9, 12, 13, 20, 23, 26, 30, 38, 40, 64, 80, 102, 160, 166, 320, 268, 640, 434,
1280]
Suma de todos los números impares del array resultado: 48
```

SOLUCIÓN INMA:

```
*/
package ejercicios_entrenamiento_programacion_febrero;

import java.util.Arrays;

/**
 *
```

```
 * @author inmar
 */
public class Ejercicio5 {

    public static void main(String[] args) {
        int[] array1 = {1, 8, 3, 1, 3, 7, 5, 2, 4, 6};
        int[] array2 = {3, 4, 5, 6, 2, 5, 1, 7, 5, 5};
        int[] arrayResultado = new int[20];

        for (int i = 0; i < arrayResultado.length / 2; i++) {
            int suma = 0;
            if (i % 2 == 0) {
                for (int x = 0; x <= i; x++) {
                    suma = suma + array2[x];
                }
                arrayResultado[i] = suma;
            } else {
                for (int x = 0; x <= i; x++) {
                    suma = suma + array1[x];
                }
                arrayResultado[i] = suma;
            }

        }

        for (int i = arrayResultado.length / 2; i < arrayResultado.length; i++) {

            if (i % 2 == 0) {
                arrayResultado[i] = arrayResultado[i - 2] + arrayResultado[i - 4];
            } else {
                arrayResultado[i] = 2 * arrayResultado[i - 2];
            }
        }

        System.out.print(Arrays.toString(arrayResultado));
    }
}
```

Ejercicio 6: validación de códigos

Número de respuestas: 0

En el siguiente ejercicio tienes que verificar si el código de un producto es correcto. El código tiene el siguiente formato:

- Un número de 10 a 99 (dos dígitos numéricos), llamado N1.
- Un guión (“-”).
- Un número de 100 a 999 (tres dígitos numéricos), llamado N2.

- Un guión (“-”).
- Una letra que podrá ser “V” o “X”. En el caso anterior sería “X”.

En este ejercicio deberás:

- Leer el código por teclado como una única cadena de texto.
- Comprobar con una expresión regular si el formato del código es adecuado. Si no lo es, el programa mostrará “formato de código no adecuado” y terminará.
- Si el formato es correcto, comprobar si el código es válido.

Para saber si el código es válido deberás multiplicar N1 por N2, y hacer el módulo 2 del resultado:

- Si el módulo es 0 la letra debería ser “V”.
- Si el módulo es 1 la letra debería ser “X”.

Por ejemplo:

15-525-X → 15 * 525 es 7875, y 7875%2 es 1 (la letra es, por tanto, “X”)

A continuación tienes una serie de códigos válidos que puedes usar para probar tu programa:

76-427-V
94-435-V
34-864-V
95-324-V
90-263-V
34-635-V
51-619-X
42-901-V
68-524-V

SOLUCIÓN INMA:

```
 */
package ejercicios_entrenamiento_programacion_febrero;

import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author inmar
 */
public class Ejercicio6 {

    public static void main(String[] args) {

        Scanner teclado = new Scanner(System.in);

        Pattern patron = Pattern.compile("([1-9][0-9]{1})-([1-9][0-9]{2})-([V,X])");

        String codigo;

        System.out.print("Introduzca código de producto: ");
        codigo = teclado.nextLine();
```

```
        Matcher encaja = patron.matcher(codigo);

        if (!encaja.matches()) {
            System.out.print("formato de código no adecuado");
        } else {
            int n1 = Integer.valueOf(encaja.group(1));
            int n2 = Integer.valueOf(encaja.group(2));
            int operacion = n1 * n2 % 2;

            if (operacion == 0 && encaja.group(3).equals("V")) {
                System.out.printf("Código válido %s", codigo);
            } else if (operacion == 1 && encaja.group(3).equals("X")) {
                System.out.printf("Código válido %s", codigo);
            } else {
                System.out.printf("Código no válido %s", codigo);
            }

        }

    }

}
```

Ejercicio 7: Validación de códigos (II)

Escribir un programa en Java para que se procese un array de entrada con números de serie y se rellene un array de salida indicando para cada número de serie si es válido o no (cadena “válido” o “inválido”).

Los requisitos que debe cumplir un número de serie para ser válido son:

1. Debe comenzar por dos letras mayúsculas. La primera obligatoriamente tiene que ser X o Y. La segunda puede ser cualquiera (sin contar la eñe).
2. A continuación habrá un guión (‘-’).
3. Ha de terminar por un año que debe encontrarse entre 1970 y el año actual.
4. Si el año es impar, la primera letra del número de serie debe ser ‘Y’. Si el año es par, la primera letra debe ser ‘X’.

El resultado que debe proporcionar el programa es el siguiente:

```
VALIDADOR DE NÚMEROS DE SERIE
-----------------------------
Lista de números de serie de prueba:
[ZA-2000, XAZ2000, XA2000, XA-1969, YH-1969, XQ-1970, XJ-2002, YV-2021, XB-2022, YV-2022, YA-1970, YH-2002, XB-2021]

RESULTADO
---------
El resultado de la comprobación de la validez de los números de serie es:
[inválido, inválido, inválido, inválido, inválido, inválido, inválido, inválido, inválido, inválido, válido, válido,
válido]
```

SOLUCIÓN INMA:

```
 */
package ejercicios_entrenamiento_programacion_febrero;

import java.time.LocalDate;
import java.util.Arrays;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author inmar
 */
public class Ejercicio7 {

    public static void main(String[] args) {

        Pattern patron = Pattern.compile("([XY])([A-Z])-([0-9]{4})");

        String[] arrayNumSerie = {"ZA-2000", "XAZ2000", "XA2000", "XA-1969", "YH-1969", "XQ-1970", "XJ-
2002", "YV-2021", "XB-2022", "YV-2042", "YA-1970", "YH-2002",
            "XB-2021"};

        String[] arraySalida = new String[arrayNumSerie.length];
        Matcher encaja;
        for (int i = 0; i < arrayNumSerie.length; i++) {
            encaja = patron.matcher(arrayNumSerie[i]);

            if (encaja.matches()) {
                int anio = Integer.valueOf(encaja.group(3));
                if (anio >= 1970 && (anio <= LocalDate.now().getYear())) {
                    if (anio % 2 != 0 && encaja.group(1).equals("Y")) {
                        arraySalida[i] = "válido";
                    } else if (anio % 2 == 0 && encaja.group(1).equals("X")) {
                        arraySalida[i] = "válido";
                    } else {
                        arraySalida[i] = "no válido";
                    }
                } else {
                    arraySalida[i] = "no válido";
                }

            }else {
                arraySalida[i] = "no válido";
            }



        }
        System.out.println(Arrays.toString(arraySalida));
    }
}
```

Ejercicio 8: Códigos de guardia

Número de respuestas: 0
En la sucursal 1 de una empresa, los empleados han de hacer guardia un día de la semana, asignada según su código de identificación personal. Este código está compuesto por 4 posiciones alfanuméricas, donde la primera indica el código de sucursal, seguidas de dos dígitos numéricos, que indican la edad del trabajador. En total 6 posiciones.

Para la sucursal 1 se ha decidido asignar las guardias de la siguiente forma:

Solo harán guardias los empleados que cumplan las siguientes condiciones:

- La primera posición de su código de empleado ha de ser 1 (código de sucursal)
- La segunda posición ha de ser una vocal mayúscula, o una S o una D (también mayúsculas)
- El resto de posiciones alfanuméricas han de ser mayúsculas.
- El empleado ha de tener 20 años o más.

El día de guardia está determinado por la segunda posición del código (vocal), correspondiendo:

A : Lunes, E: Martes, I:Miércoles, O:Jueves, U:Viernes,
S ó D : Fin de Semana completo

Si no se cumplen estas condiciones el empleado no hará guardias.

Escribe un programa en Java que determine el día de guardia / fin de semana que le toca o, en caso de no hacer guardia, que indique que no debe hacerla, para la siguiente relación de códigos de empleado:

"3ABC36", "1EDE27", "1UWX19", "1DEF32", "1AB45", "1STU45", "1aBC28", "1ABC31"

El programa debe empezar preguntando al usuario si ha de ejecutarse o no.
Solo se admitirán como respuestas correctas 0 ó 1. Si la pulsación ha sido 1 se ejecutará, y si ha sido 0 el programa terminará. En otro caso se volverá a preguntar, y en este input se ha de controlar el posible error provocado al introducir un carácter no numérico como respuesta.

Se debe usar expresiones regulares para determinar si un trabajador hace guardia o no.

Para asignar el día de la semana (salvo fin de semana), que tendrá la guardia un trabajador se debe usar una variable tipo Enum.

Salida para la lista de códigos de empleado indicada:

```
Ejecutar programa ? (0 = No, 1 = Sí):  1
ASIGNACIÓN DE GUARDIAS
======================
El empleado  3ABC36    No trabaja
```

El empleado 1EDE27   Trabaja el MARTES
El empleado 1UWX19   No trabaja
El empleado 1DEF32   Fin de Semana
El empleado 1AB45   No trabaja
El empleado 1STU45   Fin de Semana
El empleado 1aBC28   No trabaja
El empleado 1ABC31   Trabaja el LUNES

SOLUCIÓN DARK-DREAM:

```
public class Ejercicio8 {

   private enum Dia {
      Lunes, Martes, Miércoles, Jueves, Viernes;

      private static final Dia[] dias = Dia.values();
      private static final char[] letras = {'A', 'E', 'I', 'O', 'U'};

      public static Dia diaTrabajo(char letra) throws IllegalArgumentException {
         int index = -1;
         for (int i = 0; i < letras.length && index == -1; i++) {
            if (letra == letras[i]) {
               index = i;
            }
         }
         if (index == -1) {
            throw new IllegalArgumentException("Caracter no valido");
         }
         return dias[index];
      }
   }

   public static void main(String[] args) {
      final Pattern REGEX_GUARDIA = Pattern.compile("1([AEIOUSD])[A-Z]{2}[2-9][0-9]");

      String[] codigosEmpleados = {"3ABC36", "1EDE27", "1UWX19", "1DEF32", "1AB45", "1STU45", "1aBC28",
"1ABC31"};

      String[] resultado = new String[codigosEmpleados.length];

      Scanner teclado = new Scanner(System.in);
      char caracter;
      Matcher comprobacion;
      int num = 2;

      do {
         System.out.print("Ejecutar programa?(0 = No, 1 = Sí): ");
         try {
            num = teclado.nextInt();
         } catch (Exception e) {
```

```
            System.out.println("Introduce un numero entero");
            teclado.nextLine();
          }
      } while (num != 1 && num != 0);

      if (num == 0) {
         System.exit(0);
      }

      for (int i = 0; i < codigosEmpleados.length; i++) {

         comprobacion = REGEX_GUARDIA.matcher(codigosEmpleados[i]);
         if (comprobacion.matches()) {
            caracter = comprobacion.group(1).charAt(0);
            resultado[i] = (caracter == 'S' || caracter == 'D') ? String.format("El empleado %s Fin de Semana",
codigosEmpleados[i]) : String.format("El empleado %s Trabaja el %s", codigosEmpleados[i],
Dia.diaTrabajo(caracter));
         } else {
            resultado[i] = String.format("El empleado %s No trabaja", codigosEmpleados[i]);
         }

      }

      System.out.println("\nASIGNACIÓN DE GUARDIAS");
      System.out.println("======================");
      for (int i = 0; i < resultado.length; i++) {
         System.out.println(resultado[i]);
      }

   }

}
```

Ejercicio 9: Tabla de días de la semana por mes

Número de respuestas: 0

Implementa un programa en Java que solicite por teclado un año y a continuación rellene un **array bidimensional de enteros de tamaño 12x7** que contenga, para cada mes (fila), cuántos días de la semana contiene de cada tipo (cuántos lunes, cuántos martes, cuántos miércoles, etc.).

Una vez relleno ese array, mostrarlo por pantalla en forma de tabla.

Por ejemplo, si se introdujera por teclado el año 2022, el resultado que debería mostrarse por pantalla sería el siguiente:

```
AÑO 2022
--------
        L M X J V S
D
Mes  1  5 4 4 4 4 5
5
Mes  2  4 4 4 4 4 4
4
```

```
Mes  3  4 5 5 5 4 4
4
Mes  4  4 4 4 4 5 5
4
Mes  5  5 5 4 4 4 4
5
Mes  6  4 4 5 5 4 4
4
Mes  7  4 4 4 4 5 5
5
Mes  8  5 5 5 4 4 4
4
Mes  9  4 4 4 5 5 4
4
Mes 10  5 4 4 4 4 5
5
Mes 11  4 5 5 4 4 4
4
Mes 12  4 4 4 5 5 5
4
```

Eso significa que en enero hay cinco lunes, cuatro martes, cuatro miércoles, etc.

SOLUCIÓN Dark-Dream:

```
public class Ejercicio9 {

   public static void main(String[] args) {
      int[][] calendario = new int[12][7];
      char[] dias = {'L', 'M', 'X', 'J', 'V', 'S', 'D'};
      Scanner teclado = new Scanner(System.in);
      LocalDate fecha;
      int ano = 0;

      do {
         try {
            System.out.print("Año entre 1970 y 2100: ");
            ano = teclado.nextInt();
         } catch (Exception e) {
            System.out.println("Numero invalido");
            teclado.nextLine();
         }
      } while (ano < 1970 || ano > 2100);

      for (int i = 1; i <= calendario.length; i++) {
         for (int j = 1; j <= LocalDate.of(ano, i, 1).lengthOfMonth(); j++) {
            fecha = LocalDate.of(ano, i, j);
            calendario[i - 1][fecha.getDayOfWeek().getValue() - 1]++;

         }

      }

      System.out.printf("\nAÑO %s\n", ano);
      System.out.println("--------");

      for (int i = 0; i <= calendario.length; i++) {
         for (int j = 0; j <= calendario[0].length; j++) {
```

```
            if (i == 0) {
               System.out.print((j == 0) ? "      " : dias[j - 1]);
            } else {
               System.out.print((j == 0) ? String.format("Mes %2d", i) : String.valueOf(calendario[i - 1][j - 1]));

            }
            System.out.print((j < calendario[0].length) ? " " : "\n");

         }

      }

   }
}
```