

Funcionalidad

- El programa realiza los cálculos pedidos, los muestra por pantalla y los guarda en el fichero indicado.
- El programa distingue correctamente si la secuencia de nucleótidos es una proteína o no.
- Se utilizan los arrays y las constantes indicadas.
- El programa podría estar mejor estructurado y tiene alguna redundancia.
- Se utilizan 4 métodos con paso de parámetros y al menos uno de ellos usa la sentencia return. Además a uno de ellos se le pasará un objeto como parámetro.

Por lo menos uno de ellos usa la sentencia return.

```
public static char[] separarNucleotidos(String lineaNucleotidos) {
    int contador = lineaNucleotidos.length(); //medimos para
asignarle el tamaño al array
    char[] contSecuencia = new char[contador];
    lineaNucleotidos = lineaNucleotidos.toUpperCase();
    for(int i = 0; i < contador; i++) { //recorremos el array para
rellenarlo
        char tipoNucleotido = lineaNucleotidos.charAt(i);
        contSecuencia[i] = tipoNucleotido;
    }
    return contSecuencia;
}
```

Por lo menos uno de ellos recibe un objeto como parámetro

```
//Abrir el fichero de entrada.
public static Scanner abrirFicheroEntrada (Scanner teclado)
throws FileNotFoundException {
    System.out.println("Introduce el nombre del fichero: ");
    String nombre = teclado.nextLine();
    //Comprobamos que el fichero se pueda leer si no pedirá otro.
    File ficheroEntrada = new File(nombre);
    while(!ficheroEntrada.canRead()){
        System.out.println("Fichero no encontrado.");
        System.out.print("Nombre del fichero: ");
        nombre = teclado.nextLine();
        ficheroEntrada = new File(nombre);
    }
}
```

```

        //Abre le fichero y lo devuelve
        return new Scanner(ficheroEntrada);
    }

```

- Se ha creado el método escribirFichero pero no se le ha llamado 2 veces como dice el enunciado, solo se le llama para escribir en el fichero no por consola.
- Se utilizará la sentencia try-catch para evitar que el programa falle por una excepción mientras se lee o escribe en los ficheros

Nota: 4,5

Comentarios

- Están bien escritos, en los lugares apropiados y ayudan a entender el funcionamiento del programa.

Nota: 1.5

Legibilidad

- El código está bien organizado, sigue la guía de estilo y es fácil de leer.

Nota: 1

Nota de la autoevaluación 7

Programa

```

import java.util.*;
import java.io.*;
public class ADN {

    //Número mínimo de codones de una proteína
    public static final int CODONESMIN = 5;
    //Porcentaje de masa de Citosina y Guanina juntas
    public static final int PORCENTAJECG = 30;
    //Número de tipos de nucleótidos
    public static final int TIPOSNUCLEOTIDOS = 4;
    //Número de nucleótidos por codon
    public static final int NUCLEOTIDOSXCODON = 3;
    //Valores de las masas unitarias
    // [0] A = 135.128; // ADENINA
    // [1] C = 111.103; // CITOSINA
    // [2] G = 151.128; // GUNANINA
    // [3] T = 125.107; // TIMINA
    // [4] B = 100.0; //BASURA -
    public static final double[] valoresNucleotidos = {135.128,
111.103, 151.128, 125.107, 100.0};

```

```

    public static void main(String[] args) throws
FileNotFoundException{
        Scanner teclado = new Scanner(System.in);
        introduccion();

        trabajarFicheros(teclado);
    }
    /***** MÉTODOS *****/
    public static void introduccion(){
        System.out.println("Este programa genera información
sobre\nsecuencias de nucleótidos de ADN contenidas en un fichero\n");
        System.out.println("También indicará si pueden codificar
proteínas o no\nTodos los resultados se guardarán en un fichero");
        System.out.println();
    }

    //Abrir el fichero de entrada.
    public static Scanner abrirFicheroEntrada (Scanner teclado) throws
FileNotFoundException {
        System.out.println("Introduce el nombre del fichero: ");
        String nombre = teclado.nextLine();
        //Comprobamos que el fichero se pueda leer si no pedirá otro.
        File ficheroEntrada = new File(nombre);
        while(!ficheroEntrada.canRead()){
            System.out.println("Fichero no encontrado.");
            System.out.print("Nombre del fichero: ");
            nombre = teclado.nextLine();
            ficheroEntrada = new File(nombre);
        }
        //Abre el fichero y lo devuelve
        return new Scanner(ficheroEntrada);
    }

    //Crear el fichero de salida si no lo está creado ya.
    public static PrintStream escribirFichero(Scanner teclado) throws
FileNotFoundException {
        //Leer el nombre del fichero
        System.out.println("Introduce el nombre del fichero: ");
        String nombre = teclado.nextLine();
        //Si no existe lo crea nuevo
        File ficheroSalida = new File(nombre);
        return new PrintStream(ficheroSalida);
    }

    //Lee la descripción y la secuencia y muestra por pantalla.
    //El fichero tiene dos líneas por secuencia
    //Muestra los demás resultados
    public static void trabajarFicheros(Scanner teclado) throws
FileNotFoundException {
        //Gestiona cualquier excepción que ocurra mientras se lee el
        fichero.
        Scanner leerFichero = null;
        // Abre el fichero para escribir

        char[] tipoNucleotidos = new char[0];
        int[] indiceNucl = new int[TIPOSNUCLEOTIDOS];
        int[] contadores = new int[TIPOSNUCLEOTIDOS];
        double masaTotal = 0.0;
        double[] listadoMasasNucleotidos = new double[0];
        double[] valoresDecimalesNucleotidos = new double[0];
        String[] listaCodones = new String [0];
        boolean resultProteina = false;

```

```

String confirmacionProteina = "";
try{ //Si el fichero no se abre correctamente se producirá una
excepción.
    leerFichero = abrirFicheroEntrada(teclado);
    // Abre el fichero para escribir
    PrintStream escribirFich = escribirFichero (teclado);
    int cont = 0;
    while(leerFichero.hasNextLine()){
        cont++;
        String linea = leerFichero.nextLine();
        Scanner leerLinea = new Scanner(linea);

        //comprobamos que es la línea par que es la última de cada
        grupo en el archivo.
        if(cont % 2 == 0){
            System.out.print("Nucleótidos: ");
            tipoNucleotidos = separarNucleotidos(linea);
            //Nucleótidos separados cahr[]
            indiceNucl = cambiarAInt(tipoNucleotidos); //Índice de
            nucleotidos de 0 a 3 y 4 representa la basura int[]
            contadores = contadorNucleotidosReducido(
            indiceNucl); //Número de elementos
            masaTotal = resultadosMasaTotal(valoresNucleotidos
            ,indiceNucl); //masa total double
            listadoMasasNucleotidos =
            resultadosMasasNucleotidos(indiceNucl, valoresNucleotidos, masaTotal);
            //masas de los nucleótidos double[]
            listaCodones = listarCodones(linea); // listado de
            codones String[]
            resultProteina = comprobarProteina(listaCodones,
            listadoMasasNucleotidos, masaTotal); // resultado de la proteina
            boolean
            //Si el resultado de la proteina es true mostrará Sí,
            si no mostrará No
            if(resultProteina){
                confirmacionProteina = "SI";
            }else{
                confirmacionProteina = "NO";
            }
        }else{//es la primera línea de cada grupo de datos.
            linea = linea.toLowerCase();
            System.out.print("Descripción: ");
            escribirFich.print("Descripción: ");
        }
        //mientras que tenga para leer palabras las leerá
        while(leerLinea.hasNext()) {
            String palabra = leerLinea.next();
            if(cont % 2 == 0){
                escribirFich.print(palabra.toUpperCase() + " ");
                System.out.print(palabra.toUpperCase() + " ");

            }else{
                escribirFich.print(palabra.toUpperCase() + " ");
                System.out.print(palabra+ " ");
            }
        }
        System.out.println();
        escribirFich.println();
        //comprobamos que es la segunda línea de cada grupo de
        datos.

```

```

        if(cont % 2 == 0) {
            System.out.println("Contadores: " +
java.util.Arrays.toString(contadores));
            System.out.println("Masa (%): " +
java.util.Arrays.toString(listadoMasasNucleotidos) + " de " +
redondearDecimales(masaTotal));
            System.out.println("Lista Codones: " +
java.util.Arrays.toString(listaCodones));
            System.out.println("Es proteina: " +
confirmacionProteina);
            System.out.println();

            escribirFich.println("Contadores: " +
java.util.Arrays.toString(indiceNucl));
            escribirFich.println("Masa (%): " +
java.util.Arrays.toString(listadoMasasNucleotidos) + " de " +
masaTotal);
            escribirFich.println("Lista de codones: " +
java.util.Arrays.toString(listaCodones));
            escribirFich.println("Es proteina: " +
confirmacionProteina);
            escribirFich.println();
        }
    }
    escribirFich.close();

} catch (Exception e) {
    System.out.println(e.getMessage()); //Muestra que excepción ha
ocurrido
}
//Cierra el fichero si la conexión se ha realizado
if(leerFichero != null) {
    leerFichero.close();
}
}

//Método para separar la secuencia de nucleotidos
//Recibe un String y devuelve un array char[]
public static char[] separarNucleotidos(String lineaNucleotidos) {
    int contador = lineaNucleotidos.length(); //medimos para
asignarle el tamaño al array
    char[] contSecuencia = new char[contador];
    lineaNucleotidos = lineaNucleotidos.toUpperCase();
    for(int i = 0; i < contador; i++){ //recorremos el array para
rellenarlo
        char tipoNucleotido = lineaNucleotidos.charAt(i);
        contSecuencia[i] = tipoNucleotido;
    }
    return contSecuencia;
}

//Método que convierte los tipos de nucleotidos en un índice del 0
al 3, el 4 índice representa la basura.
//Recibe un array char[] y devuelve un array int[] con el índice
public static int[] cambiarAInt(char[] tiposNucleotidos) {
    int[] indiceNucleotidos = new int[TIPOSNUCLEOTIDOS + 1]; // + 1
basura
    for(int i = 0; i < tiposNucleotidos.length; i++){
        switch(tiposNucleotidos[i]) {
            case 'A' :
                indiceNucleotidos[0]++;

```

```

        break;
    case 'C' :
        indiceNucleotidos[1]++;
        break;
    case 'G' :
        indiceNucleotidos[2]++;
        break;
    case 'T' :
        indiceNucleotidos[3]++;
        break;
    case '-' :
        indiceNucleotidos[4]++;
        break;
    }
}
return indiceNucleotidos;
}

public static int[] contadorNucleotidosReducido(int[]
indiceNucleotidos) {
    int[] contadorReducido = new int[TIPOSNUCLEOTIDOS];
    for(int i = 0; i < TIPOSNUCLEOTIDOS; i++){
        contadorReducido[i] = indiceNucleotidos[i];
    }
    return contadorReducido;
}

//Método para calcular la masa total recibe un array double[] con
los valores de los nucleótidos y un int[] con el contador de
nucleótidos.
//devuelve un double con la masa total.
public static double resultadosMasaTotal(double[]
valoresNucleotidos, int[] cantidadNucleotidos) {
    double masaTotal = 0.0;
    for(int i = 0; i < valoresNucleotidos.length; i++) {
        //Calculamos la masa total
        masaTotal += valoresNucleotidos[i] * cantidadNucleotidos[i];
    }
    return masaTotal;
}

//Método que calcula las masas de los nucleótidos recibe un int[]
con el contador de nucleótidos,
//un double[] con los valores de cada nucleótido y la masa.
Devuelve el cálculo de nucleótidos
public static double[] resultadosMasasNucleotidos(int[]
contadorNucleotidos, double[] valoresNucleotidos, double masa) {
    double masaTotalNucleotido = 0.0;
    masaTotalNucleotido = redondearDecimales(masaTotalNucleotido);
    //Calculamos el % cada nucleótido
    double[] sumaNucleotidos = new double[TIPOSNUCLEOTIDOS];
    for(int j = 0; j < TIPOSNUCLEOTIDOS; j++ ) {
        //(Valor Nucleotidos * Número Nucleotidos * 100) / MasaTotal
        sumaNucleotidos[j] = ((valoresNucleotidos[j] *
contadorNucleotidos[j]) * 100) / masa;
        sumaNucleotidos[j] = redondearDecimales(sumaNucleotidos[j]);
    }
    return sumaNucleotidos;
}

//Métod que redondea un número a 1 decimal, recibe un número double
y devuelve un número double.

```

```

public static double redondearDecimales (double num) {
    num = (double) Math.round(num * 10) / 10;
    return num;
}

//Método que lista los codones y quita los guiones, recibe un
String y devuelve un Array String[] con las lista de codones.
public static String[] listarCodones(String lista) {
    int tamanhoLista = lista.length();
    if(lista.contains("-")){
        lista = lista.replace("-", "");
        int tamanhoListaNueva = lista.length();
        int NuevaTamanhoLista = tamanhoLista - tamanhoListaNueva;
        String[] listaCodones = new String[NuevaTamanhoLista /
NUCLEOTIDOSXCODON];
    }
    String[] listaCodones = new String[lista.length() /
NUCLEOTIDOSXCODON];
    int cont = 0;
    for(int i = 0; i < lista.length() / NUCLEOTIDOSXCODON; i++ ) {
        String cadenaNucleotidos = lista.substring(cont, cont +
NUCLEOTIDOSXCODON).toUpperCase();

        listaCodones[i] = cadenaNucleotidos;
        cont +=3;
    }
    return listaCodones;
}

//Método para comprobar si es proteína o no, recibe un array
String[] con los codones y devuelve un boolean.
public static boolean comprobarProteina(String[] codones, double[]
listadoMasasNucleotidos, double masaTotal) {
    String[] codonValido = {"ATG", "TAA", "TAG", "TGA"}; //Array que
contiene los codones validos;
    //Comienza con un codón valido, es decir el primer codón es ATG
    if(!codones[0].equals(codonValido[0])) {
        return false;
    }
    //Finaliza con un codón válido, es decir el último codón será
TAA, TAG o TGA
    int invalido = 0;
    //creamos un marcador, si no es ninguno se incrementara, si no
es ninguno de los 3 devolvera false
    for(int i = 1; i < 4; i++){
        if(!codones[codones.length - 1].equals(codonValido[i])) {
            invalido++;
        }
        if(invalido == 3){
            return false;
        }
    }
    //Contiene al menos 5 codones incluidos el de inicio y el de fin

    int cuentaCodones = 0;
    for(int i = 0; i < codones.length; i++) {
        cuentaCodones++;
        if(cuentaCodones == 5){
            break;
        }
    }
    if(cuentaCodones < CODONESMIN) {

```

```
        return false;
    }
    //Citosina (C) y Guanina (G) juntos suponen al menos el 30% del
    total de la masa.
    //Sumamos la citosina y la guanina
    double totalCG = listadoMasasNucleotidos[1] +
listadoMasasNucleotidos[2];
    if(totalCG < PORCENTAJECG){
        return false;
    }
    return true;
}
} //class
```