Solución: Cuaderno de ejercicios: Optimización de código

Ejercicio 1: Varios métodos a optimizar:

Optimiza los siguientes métodos.
Dificultad: ★★★

Solución:

```
public class EjercicioOptimizacion {
   void ejemplo1Optimizado() {
       int b = 0, c = 0, d = 0;
       int a = b + c;
       d = a - d;
       int e = a * d;
   }
    void ejemplo2Optimizado() {
       int valor = 0, item;
       item = 10;
       do {
           valor += valor + item;
       } while (valor < 100);
   }
    void ejemplo3Optimizado() {
       String total = "";
       String m = "Mensaje hola";
       int i;
       for (i = 0; i < 10; i++) {
           // otras operaciones....
       }
       total = m + (i - 1);
       System.out.println(total);
   }
```

```java
 void ejemplo4Optimizado() {
    int i = 5;
    int j = 4;
    float f = 6.5f;
}
 void ejemplo5Optimizado() {
    int i = 10, c = 10, m = 10;
    int a = 3 + i;
    // int f=a;
    int b = a + c;
    int d = a + m;
    m = a + d;
}
 void ejemplo7Optimizado() {
    int i = 1;
    float array[] = new float[5];
    float temporal = array[8 + i];
    // tofix (i+1)*5*8 se puede a'adir a temporal
    // tofix potimizar calculos constantes 5+1 y 6+1
    array[i] = temporal + (i + 1) * 5 * 8 + (5 + 1);
    array[i - 1] = temporal + (i + 1) * 5 * 8 + (6 + 1);
}
 void ejemplo8Optimizado() {
    int pos = 3;
    int contador = 0;
    int array[] = new int[5];
    int temporal = array[pos];
    for (int i = 0; i < 2000; i++) {
        // cosas
        contador = temporal + i;
        // cosas
    }
    System.out.println(contador);
    // cosas
}

void ejemplo9Optimizado() {
    int a = 10;
    float b = a * 0.5f;
}

 int ejemplo10Optimizado() {
    int x = 10, y = 20;
    return x / y;
}

void ejemplo11Optimizado() {
    int i = 0;
    while (i < 10) {
        System.out.println("hola:" + i);
        i++;
    }
    i = 0;
    while (i < 10) {
        System.out.println("adios:" + i);
        i++;
```

```
        }
    }
}
```

Ejercicio 2: Trabajo con array de enteros:

Algunos de los métodos de este ejercicio tienen mejoras de optimización. Tu tarea será encontrarlas y solucionarlas.
Además añade un comentario (encima del método) indicando, mediante una breve reseña, que has hecho para solucionar el problema.
Dificultad: ★★★

Solución:

```
import java.util.Arrays;

public class Optimiza {
    int numeros[] = { -5, 3, 6, 66, 55, 2, -7, 6, 1 };

    /* Este método busca si un numero está en una lista de números */
    boolean busca(int numeroBuscado) {
        for (int n : numeros) {
            if (numeroBuscado == n) {
                return true;
            }
        }
        return false;
    }

    /* Este funcion cuantos números positivos hay en la lista de numeros */
    int cuentaPositivos() {
        int contador = 0;
        for (int n : numeros) {
            if (n >= 0) {
                contador++;
            }
        }
        return contador;
    }

    /*
     * Este método calcula la media de todos los números guardado en la lista de
     * números
     */
    float calculaMedia() {
        float cont = 0;
        for (int num : numeros) {
```

```
            cont += num;
        }
        return cont / numeros.length;
    }

    /*
     * Este método divide cada número de la lista entre la media de todos los
     * numeros
     */
    float[] dividelosPorLaMedia() {
        float nuevosNumeros[] = new float[numeros.length];
        float media = calculaMedia();
        for (int i = 0; i < numeros.length; i++) {
            nuevosNumeros[i] = numeros[i] / media;
        }
        return nuevosNumeros;
    }

    /*
     * Este método calcula la mediana de la lista de numeros.
     * Recueda que la mediana representa el valor de la variable de posición central
     * en un conjunto de datos
     */
    double calculaMediana() {
        int[] copiedArray = numeros.clone();
        Arrays.sort(copiedArray);
        int mediana;
        int mitad = copiedArray.length >> 1;
        if (copiedArray.length % 2 == 0) { // Si la longitud es par, se deben promediar
los del centro
            mediana = (copiedArray[mitad - 1] + copiedArray[mitad]) >> 1;
        } else {
            mediana = copiedArray[mitad];
        }
        return mediana;
    }

    /*
     * Este método calcula el valor que más se repite en la lista
     */
    int moda() {
        int maximaVecesQueSeRepite = 0;
        int moda = 0;
        for (int i = 0; i < numeros.length; i++) {
            int vecesQueSeRepite = 0;
            for (int j = 0; j < numeros.length; j++) {
                if (numeros[i] == numeros[j])
                    vecesQueSeRepite++;
            }
            if (vecesQueSeRepite > maximaVecesQueSeRepite) {
                moda = numeros[i];
                maximaVecesQueSeRepite = vecesQueSeRepite;
            }
        }
        return moda;
    }
```

```
    public static void main(String[] args) {
        new Optimiza();
    }

    public Optimiza() {
        System.out.println("Numeros: " + Arrays.toString(numeros));
        System.out.println("Tiene el 5:" + busca(5));
        System.out.println("Tiene el 2:" + busca(2));
        System.out.println("Hay " + cuentaPositivos() + " números positivos");
        System.out.println("La media vale: " + calculaMedia());
        System.out.println("Cada número dividido por la media de todos:" +
Arrays.toString(dividelosPorLaMedia()));

        System.out.println("La mediana vale:" + calculaMediana());
        System.out.println("La moda vale:" + moda());
    }
}
```

Ejercicio 3: Algoritmo:

Optimiza el siguiente método.

Solución:

```
public class Optimizacion {
    public int[] algoritmo(int[] datos) {
            //float array[] = new float[10];
            //int i=1;
            //int sumador=2;
            int limite = datos.length-2;
            for(int i=1;i<limite;i++) {
                //int temp = datos[2+i] + (i+1)*16 +2;
                //int temp = datos[2+i] + 16*i + 18 + 7;
                datos[i] = datos[2+i] + 16*i + 18 + 7 ;
                datos[i-1] =  datos[i]  + 1;
                //int z = datos[2+i] / sumador;
            }
            return datos;
    }
    public static void main(String[] arg) {
        Optimizacion ejemplo = new Optimizacion();
        int[] datos = new int[] {1,2,3,4,5,6,100,90};
        ejemplo.algoritmo(datos);

        for (int d : datos) {
            System.out.println(d);
        }
    }
}
```

Ejercicio 4: Trasladar punto

Realiza primero los Test unitarios JUnit y luego optimiza el código.

Dificultad: ★★

Solución: Test

```
import static org.junit.jupiter.api.Assertions.*;

import java.awt.geom.Point2D;

import org.junit.jupiter.api.Test;

class OptimizacionTest {

   @Test
   void testTrasladarPunto() {
       Optimizacion o = new Optimizacion();
       Point2D.Double p = new Point2D.Double(2,5);
       Point2D.Double nuevoPunto = o.trasladarPunto(p, 90, 5, 1);
       assertEquals(2.0, nuevoPunto.x, 0.1);
       assertEquals(10.0, nuevoPunto.y, 0.1);
   }

}
```

Solución: Optimización

```
import java.awt.geom.Point2D;

public class Optimizacion {
   /**
   * Mueve un punto en 2d en funci'n de su velocidad, y su angulo
   * @param puntoInicial
   * @param anguloGrados
   * @param espacio
   * @param tiempo
   * @return
   */
   public Point2D.Double trasladarPunto(Point2D.Double puntoInicial, float anguloGrados,
float espacio, float tiempo) {
       float velocidad = espacio / tiempo;
       float anguloRadianes = (float)Math.toRadians(anguloGrados);
       //double x = puntoInicial.x + (velocidad) * Math.cos(anguloRadianes);
       //double y = puntoInicial.y + (velocidad) * Math.sin(anguloRadianes);
       return new Point2D.Double(
               puntoInicial.x + velocidad * Math.cos(anguloRadianes),
               puntoInicial.y + velocidad * Math.sin(anguloRadianes)
           );
   }
   public static void main(String[] args) {
```

```
        Optimizacion ejemplo = new Optimizacion();
        Point2D.Double p = new Point2D.Double(2.0,5.0);
        Point2D.Double nuevoPunto = ejemplo.trasladarPunto(p, 90, 5, 1);
        System.out.println(nuevoPunto);
    }
}
```

Ejercicio 5: Varios métodos a optimizar:

Optimiza los siguientes métodos.
Dificultad: ★★★

Solución:

```
import java.util.Arrays;

public class Principal {

    String getNombre() {
        StringBuilder sb = new StringBuilder();  //StringBuffer
        sb.append("Antonio");
        sb.append("Jose");
        sb.append("Maria");
        sb.append("Ruben");
        return sb.toString();
    }
    float calculo(int c) {
        float resultado = 0;
        float cto = 5;
        do {
            resultado += c + cto;
            //TOFIX corregir el problema del bucle infinito  c++
        }while(c<20);
        return resultado;
    }
    int[] calculo2() {
        int[] lista = {3,5,7};
        //int[] l1 = lista.clone();
        Arrays.sort(lista);
        if(lista.length > 10) {
            //Arrays.sort(l1);
            return lista;
        }else
            return null;
    }

    float contar(int[] listaNumeros) {
        return listaNumeros.length << 2; // Multiplicar por 4
    }
```

```
   float calculo3(int[] listaNumeros) {
       //int valor = 0;
       float total = contar(listaNumeros);
       //total=4

       //return (total-1) * (total-1+1) / 2;
       return (total-1) * (total) / 2;
   }
}
```

Ejercicio 6: Pintar en pantalla números

Optimiza el siguiente método.

Solución:

```
   void ejercioSinOptimizar() {
          int i=1;
          StringBuilder sb = new StringBuilder();
       while(i<1000000) {
          sb.append(i);
          sb.append("\n");
           i++;
       }
       System.out.println(sb.toString());
   }
```

Ejercicio 7: Nombres profesores

La siguiente clase no está totalmente optimizada,
Optimízala tú. Yo al menos veo 5 optimizaciones que se pueden aplicar.
NOTA: (que no te importe si está construida con código limpio, tu solo optimízala)

Solución:

```
public class Optimiza {
   String nombres[] = { "Angel", "Bea", "Pepe" };
    /**
```

```
    * Este método comprueba si un nombre está en la lista
    *
    * @param nombre valor a comprobar
    * @return true si el nombre está en la lista
    */
   boolean compruebaSiEsta(String nombreBuscado) {
       for (String nombre : nombres) {
           if (nombreBuscado.equals(nombre)) {
               return true;
           }
       }
       return false;
   }
    /**
    * Este método hace un calculo sobre la lista
    *
    * @return un valor cualquiera
    */
   float calculo() {
       int suma = 0;
       for (String nombre : nombres) {
           suma += nombre.length();
       }
       return suma / 8 + 5;
   }
    /**
    * Este método devuelve todos los nombres de la lista concatenados uno a
    * continuación del otro
    *
    * @return los nombres concatenados
    */
   String obtenerNombresConcatenados() {
       StringBuilder sb = new StringBuilder();
       for (String nombre : nombres) {
           sb.append(nombre);
       }
       return sb.toString();
   }
}
```

Ejercicio 8: Operaciones con arrays de nombres

La siguiente clase no está totalmente optimizada,
Optimiza el siguiente código.
También refactorízalo en la medida de lo posible.

Solución optimiza

```
import java.io.Reader;
```

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Scanner;

public class Nombres {
	ArrayList<String> lista = new ArrayList<String>(Arrays.asList("Angel",
"Pepe", "Bea"));

	void Insert(String nuevoNombre) {
		lista.add(nuevoNombre);
	}

	void InsertMultiple(String... nombres) {
		for (String nombre : nombres) {
			Insert(nombre);
		}
	}

	void removeLastElement() throws Exception {
		if (lista.isEmpty())
			throw new Exception("No hay elementos para borrar");
		lista.remove(lista.size() - 1);
	}

	void clear() {
		lista.clear();
	}

	void sortNames() {
		Collections.sort(lista);
	}

	String getAllNames() {
		String resultado = "";
		for (String nombre : lista) {
			resultado += nombre + "\n";
		}
		return resultado;
	}

	void printAllNames() {
```

```
            System.out.println("---- Lista de nombres -----");
            System.out.println(getAllNames());
      }

      void rellenarNombres() {
            Scanner in = new Scanner(System.in);
            System.out.println("Introduce la cantidad de nombres a leer: ");
            int cantidad = in.nextInt();
            for (int i = 0; i < cantidad; i++) {
                  System.out.println("Introduce el nombre: ");
                  String nombre = in.next();
                  Insert(nombre);
            }
            in.close();
      }

      public static void main(String[] args) {
            try {
                  Nombres n = new Nombres();
                  n.rellenarNombres();
                  n.printAllNames();
                  n.Insert("Miguel");
                  n.printAllNames();
                  n.removeLastElement();
                  n.printAllNames();
                  n.clear();
                  n.printAllNames();
                  n.removeLastElement();
                  n.printAllNames();
                  n.Insert("Paula");
                  n.Insert("Carmen");
                  n.Insert("Ana");
                  n.sortNames();
                  n.printAllNames();
            } catch (Exception ex) {
                  System.out.println("Fallo:"+ex.getMessage());
            }

      }
}
```

Solución test

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.fail;

import org.junit.jupiter.api.Test;

public class NombresTest {
    @Test
    void testInsert() {
        Nombres n = new Nombres();
        n.insert("Miguel");
        assertEquals(4, n.lista.size());
        int ultimaPosicon = n.lista.size() - 1;
        assertEquals("Miguel", n.lista.get(ultimaPosicon));
    }

    @Test
    void insertarMuchos() {
        Nombres n = new Nombres();
        n.insertMultiple("jose", "pedro", "ana", "roi", "pio", "felipe");
        assertEquals(9, n.lista.size());
    }

    @Test
    void eliminarUltimo() {
        Nombres n = new Nombres();
        try {
            n.removeLastElement();
            assertEquals(2, n.lista.size());
        } catch (Exception e) {
            e.printStackTrace();
            fail();
        }
    }

    @Test
```

```
    void borrarLista() {
        Nombres n = new Nombres();
        n.clear();
        assertEquals(0, n.lista.size());
    }

    @Test
    void ordenar() {
        Nombres n = new Nombres();
        n.sortNames();
        assertEquals("Angel", n.lista.get(0));
        assertEquals("Bea", n.lista.get(1));
        assertEquals("Pepe", n.lista.get(2));
    }
}
```

Ejercicio 9: Cálculos de frases

La siguiente clase no está totalmente optimizada,
Optimiza el siguiente código.
También refactorízalo en la medida de lo posible.

Se parte del siguiente programa que almacena un conjunto de frases y realiza una serie de cálculos sobre ellas.
Las frases no se van a modificar (ni se añaden o quitan frases), con lo cual **no será necesario** usar un ArrayList, con un array tradicional (como el que se usa en el programa) es suficiente.
Puede que no todos los métodos requieren ser optimizados.
Pon encima de cada método, una breve explicación de lo que has hecho para optimizar.
Dificultad: ★★★

```
import java.util.Arrays;

public class OptimizaSolucion {
   String frases[] = { "Hola", "Angel", "que tal estas", "Angel", "Pepe", "que tal
estoy"};

   /* Este método busca si una frase está en la lista de frases */
   /* sobra una variable y retornamos directamente */
   boolean busca(String fraseBuscar) {
       for (String frase : frases) {
           if (frase.equals(fraseBuscar)) {
```

```
                return true;
            }
        }
        return false;
    }

    /* Devuelve la frase mas larga (si hay varias de igual tamaño devuelve la
primera que se encuentra) */
    /* Comenzamos en 1 */
    String masLarga() {
        String masLarga = frases[0];
        for(int i=1;i<frases.length;i++){
            if (frases[i].length() >= masLarga.length()) {
                masLarga = frases[i];
            }
        }
        return masLarga;
    }
    /* Este cuenta cuantos frases de como minimo X letras hay */
    // Sobra variable num
    int cuentaFrasesMinimoLetras(int minimoLetras) {
        int contador = 0;
        for (String frase : frases) {
            if (frase.length() >= minimoLetras) {
                contador++;
            }
        }
        return contador;
    }

    /* Calcula la media de caracteres de todas las frases*/
    // variable suma será un entero no un float
    float mediaCaracteres(){
        int suma = 0;
        for(String frase: frases){
            suma+=frase.length();
        }
        return (float)suma / frases.length;
    }

    /* Por cada frase, devuelve el numero de caracteres de cada frase dividido por
la media de caracteres totales  */
    // Contamos numero de caracteres de cada frase = [4, 5, 13, 5, 4, 13]
    // Numero total de frases = 6
    // Media total de caracteres  = 4+5+13+5+4+13  /  6  = 7.33
```

```
   // Dividimos el numero de caracteres de cada frase por la media total de
caracteres
   // [0.5714286, 0.71428573, 1.8571428, 0.71428573, 0.5714286, 1.8571428]

   // extraemos la media fuera del for
   float[] numerosCaracteresEntreMediaTotal(){
       float media = mediaCaracteres();
       float[] resultado = new float[frases.length];
       for(int i=0;i<frases.length;i++){
           resultado[i] = frases[i].length() / media ;
       }
       return resultado;
   }

   /* Devuelve la frase que mas se repite (moda) */
   /* no es necesario hacer nada */
   String moda() {
       int maximaVecesQueSeRepite = 0;
       String moda = "";
       for (int i = 0; i < frases.length; i++) {
           int vecesQueSeRepite = 0;
           for (int j = 0; j < frases.length; j++) {
               if (frases[i] == frases[j])
                   vecesQueSeRepite++;
           }
           if (vecesQueSeRepite > maximaVecesQueSeRepite) {
               moda = frases[i];
               maximaVecesQueSeRepite = vecesQueSeRepite;
           }
       }
       return moda;
   }
/*
   no es necesairo usar dos modas
   el abs de la moda no es necesario
   mediaEntera fuera del for
   la moda la multiplicado por 2
   uso >>
   */
   int calcula(){
       int l = 0;
       int moda = moda().length();
       //moda=Math.abs(moda);
       int mediaEntera = (int)mediaCaracteres() + moda << 1;
       for(int i=0;i<frases.length;i++){
           l += frases[i].length() + mediaEntera;
       }
       return l>>3;
```

```
    }


   public static void main(String[] args) {
       new OptimizaSolucion();
   }

   public OptimizaSolucion() {
       System.out.println("Frases:: " + Arrays.toString(frases));
       System.out.println("Contiene la frase Angel:" + busca("Angel"));
       System.out.println("Contiene la frase Bea:" + busca("Bea"));
       System.out.println("Hay " + cuentaFrasesMinimoLetras(5) + " frases de mínimo
5 letras");
       System.out.println("La frase mas larga es: " + masLarga());
       System.out.println("Media caracteres: " + mediaCaracteres());

       System.out.println("La moda vale:" + moda());
       System.out.println("Media de cada frase:" +
Arrays.toString(numerosCaracteresEntreMediaTotal()));
   }
}
```