[15/12/2025]

Trabajo de programacion

Introduccion/Objectivo

Este trabajo explica cómo funciona un programa en Java que calcula los llamados *números afortunados*. La idea es bastante simple: empezamos con una lista de números del 1 al N y vamos quitando elementos siguiendo unas reglas, primero cada 2, luego cada 3, y así sucesivamente, hasta que ya no se pueden eliminar más. Al final, los que quedan en la lista son los afortunados.

El código está dividido en funciones pequeñas para que sea más fácil de entender y mantener, asi siendo bastante modular. Además, se han añadido comentarios y pruebas para comprobar que todo funciona bien. Con este documento se busca mostrar cómo se ha construido el programa, cómo se organiza, y qué resultados da con ejemplos concretos de pruebas realizadas y complementar este documento junto al javadocs creado y la presentación realizada.

Imágenes del código

```java
package org.example;
import java.util.Scanner;

Rename usages
/**
 * @Author Kevin Muñoz Corcoles
 * @deprecated  implementa la generación y filtrado de números afortunados.
 * Contiene métodos para crear vectores, eliminar elementos según reglas,
 * compactar resultados y aplicar la salida de números afortunados (mayor a menor).
 *
 */

public class NumerosAfortunados {  new *

    /**
     * @param numero - Pasamos el numero para crear un vector con el tamaño de ese numero
     * @return vector - devolvemos el vector creado
     */
    public static int[] crearVector(int numero) {  1 usage  new *
        int[] vector = new int[numero]; //Crea la anchura del vector
        for (int i = 0; i < numero; i++) { // mete los numeros en orden
            vector[i] = i + 1;
        }
        return vector;
    }
```

```java
/**
 * @param vector - pasamos el vector
 * @param paso - pasamos el paso que es cada posicion que queremos quitar ( cada 2, cada 3...)
 * @return compactar(nuevo, contador) - devolvemos el vector despues de ser compactado
 */
public static int[] eliminarCadaPaso(int[] vector, int paso) {  1 usage  new *
    int longitud = vector.length; //tomamos la longitud
    int[] nuevo = new int[longitud]; //creamos el nuevo vector que utilizaremos
    int contador = 0; //contamos cuantos espacios tendra que tener el nuevo vector

    for (int i = 0; i < longitud; i++) {
        if (i % paso != 0) { // eliminamos el primero y luego cada 'paso' (índices múltiplos de
            nuevo[contador] = vector[i]; //colocamos solo los numeros que queramos en posiciones
            contador++; //contamos el espacio
        }
    }
    return compactar(nuevo, contador); //antes de entregarlo lo compactamos llamando a la funcio
}
/**
 * @param vector - pasamos el vector con los espacio con 0
 * @param contador - pasamos el contador con los espacios rellenos con numeros
 * @return resultado - devolvemos el vector sin espacio con 0
 */
public static int[] compactar(int[] vector, int contador) {  1 usage  new *
    int[] resultado = new int[contador]; //creamos el vector final
    for (int i = 0; i < contador; i++) {
        resultado[i] = vector[i]; //metemos los numeros hasta llenar el resultado, controlado gr
    }
    return resultado;
}
```

```java
/**
 * @param vector - Pasamos el vector con todos los numeros
 * @return vector - devuelve el vector  con los numeros afortunados despues de haber sido elimin
 */
public static int[] quitarNumeros(int[] vector) {  1 usage  new *
    int longitud = vector.length;
    for (int paso = 2; paso <= longitud; paso++) {
        vector = eliminarCadaPaso(vector, paso); // usamos la funcion eliminarCadaPaso para elim
        longitud = vector.length; //guardamos la nueva longitud del vector
    }
    return vector;
}
/**
 * @param numero - pasamos -1 àra asegurarnos que el bucle se hara
 * @return numero- el numero ya verificado que hayamos elegido
 */
public static int Filtro(int numero) {  1 usage  new *
    Scanner entrada = new Scanner(System.in);
    do {
        System.out.print("Introduce un número entre 1 y 10000 (0 para salir): ");
        String linea = entrada.nextLine();

        try {
            numero = Integer.parseInt(linea);
            if (numero == 0) { //si es cero nos da igual porque el 0 es cerrar el programa

                break;
            } else if (numero < 1 || numero > 10000) { //queremos que sea el numero de este rang
                System.out.println("Número fuera de rango. Debe estar entre 1 y 10000.");
                numero = -1;
            }
        } catch (NumberFormatException e) { //si el que mete el numero de casualidad o saber por
            System.out.println("Entrada inválida. Solo se permiten números enteros.");
            numero = -1;
        }

    } while (numero == -1); //hacemos que el bucle siempre se repita haciendo que en caso de que

    return numero;
```

```
        return numero;
    }

    public static void main(String[] args) {  new *
        while (true) {
            int numero = Filtro( numero: -1); //Filtramos el numero usando la funcion filtro
            if (numero == 0) {
                System.out.println("Programa terminado.");
                break;
            }
            int[] vector = crearVector(numero); //usamo la funcion crear vector para crar el vector

            // enseñar el vector que hemos creado :3
            System.out.print("Vector inicial: ");
            for (int num : vector) {
                System.out.print(num + " ");
            }
            System.out.println();

            // eliminar los numeros que tengamos que eliminar  :^/ (modo termineitor)
            int[] resultado = quitarNumeros(vector); //Usamos la funcion quitar numeros

            // Mostrar resultado final °W°
            System.out.print("Números afortunados: ");
            for (int i = resultado.length - 1; i >= 0; i--) {
                System.out.print(resultado[i] + " ");
            }
            System.out.println();
        }
    }
}
```

Pruebas unitarias

```java
package org.example;
import org.junit.jupiter.api.Test;

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

class bonitoTest {  new *

    @Test  new *
    void testCrearVector() {
        int[] vector = bonito.crearVector( numero: 5); // crear vector de 5
        assertArrayEquals(new int[]{1, 2, 3, 4, 5}, vector); // debe salir [1,2,3,4,5]
    }

    @Test  new *
    void testEliminarCadaPaso() {
        int[] vector = {1, 2, 3, 4, 5, 6}; // vector inicial
        int[] resultado = bonito.eliminarCadaPaso(vector,  paso: 2); // eliminamos cada 2 empezando por el primero
        assertArrayEquals(new int[]{2, 4, 6}, resultado); // deben quedar los pares
    }

    @Test  new *
    void testCompactar() {
        int[] vector = {1, 2, 3, 0, 0}; // vector con ceros
        int[] resultado = bonito.compactar(vector,  contador: 3); // compactamos hasta contador=3
        assertArrayEquals(new int[]{1, 2, 3}, resultado); // deben quedar solo los 3 primeros
    }

    @Test  new *
    void testQuitarNumeros() {
        int[] vector = bonito.crearVector( numero: 10);
        int[] resultado = bonito.quitarNumeros(vector);
        assertArrayEquals(new int[]{4, 6, 10}, resultado);
    }
}
```

```
bonitoTest (org.example)  58 ms
  testCompactar()  46 ms
  testCrearVector()  12 ms
  testQuitarNumeros()
  testEliminarCadaPaso()

4 tests passed  4 tests total, 58 ms
"C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe" ...

Process finished with exit code 0
```

Pruebas manuales:

Colocar un numero y lo haga bien:

```
"C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.e
Introduce un número entre 1 y 10000 (0 para salir): 10
Vector inicial: 1 2 3 4 5 6 7 8 9 10
Números afortunados: 10 6 4
Introduce un número entre 1 y 10000 (0 para salir):
```

Colocar un 0 para que termine:

```
"C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe"
Introduce un número entre 1 y 10000 (0 para salir): 10
Vector inicial: 1 2 3 4 5 6 7 8 9 10
Números afortunados: 10 6 4
Introduce un número entre 1 y 10000 (0 para salir): 0
Programa terminado.

Process finished with exit code 0
```

Colocar un numero fuera de lo permitido y te avise de ello yte vuelva a preguntar:

```
"C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe"
Introduce un número entre 1 y 10000 (0 para salir): 10000000
Número fuera de rango. Debe estar entre 1 y 10000.
Introduce un número entre 1 y 10000 (0 para salir):
```

Colocar una letra y te avise de ello para luego volver a preguntarte:

```
"C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe" "-javaagent:C:\
Introduce un número entre 1 y 10000 (0 para salir): 10000000
Número fuera de rango. Debe estar entre 1 y 10000.
Introduce un número entre 1 y 10000 (0 para salir): a
Entrada inválida. Solo se permiten números enteros.
Introduce un número entre 1 y 10000 (0 para salir):
```

Mejoras/Juez

Después de pasar el código el código al juez me daba run time error, Después de mirar porque e intentar comprimir y quitar todo lo posible, seguía dando el mismo error, despues lo intente con ayuda de la ia como se me aconsejo. Los resultados fueron los mismos, después de investigarlo me di cuenta de que el problema realmente era que el ejercicio se tendría que hacer con arrayslist, para que no diese el “run time error”,

⭐ CÓDIGO FINAL QUE EL JUEZ ACEPTA

java Copiar código

```java
import java.util.*;

public class Main {

    // Precomputar valores hasta MAXN = 10000
    static final int MAXN = 10000;
    static int[] respuesta = new int[MAXN + 1];

    public static void precomputar() {
        // Generamos lista inicial de 1..MAXN
        ArrayList<Integer> lista = new ArrayList<>();
        for (int i = 1; i <= MAXN; i++) lista.add(i);

        int pasoIndex = 1; // Empezamos en paso=2 (índice 1)

        // Criba de números afortunados
        while (pasoIndex < lista.size()) {
            int paso = lista.get(pasoIndex);
            if (paso > lista.size()) break;

            // Eliminar cada 'paso'
            for (int i = lista.size() - 1; i >= 0; i--) {
                if ((i + 1) % paso == 0) lista.remove(i);
            }

            pasoIndex++;
        }

        // Para cada N, buscamos el mayor afortunado <= N
        int idx = 0;
        for (int n = 1; n <= MAXN; n++) {
            while (idx + 1 < lista.size() && lista.get(idx + 1) <= n) {
                idx++;
            }
            // El mayor afortunado menor o igual a n
            if (lista.get(idx) <= n) respuesta[n] = lista.get(idx);
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        precomputar(); // ← Se ejecuta UNA SOLA VEZ
```

Fecha	12/12/2025, 17:48:02 (UTC)
Lenguaje del envío	Java
Veredicto	Run-time error (RTE)
Comentario (0/1000)	Comentario de este envío para diferenciarlo de otros. Al igual que el código, sólo será visible por ti.

Guardar