

PRACTICA 1 UD3 Héctor Crespo

https://github.com/Hectorsxo/U3_Practica1.git

Ejercicio 1. Realiza un menú para una aplicación de mantenimiento de vehículos. Clase: com.ieschabas.MenuVehiculos Método: public static String seleccionarOpcion(int opcion); Detalles: - Opción 1 → "Mantenimiento de clientes" - Opción 2 → "Introducción de facturas" - Opción 3 → "Listado de facturas" - Opción 4 → "Finalizar" - Otros valores imprimen un mensaje indicando "La opción no existe. Vuelve a intentarlo".

```
package com.ieschabas;

import java.util.Scanner;

/**
 * Programa que muestra un menú para una aplicación de mantenimiento
 * de vehículos.
 *
 * Permite al usuario seleccionar entre varias opciones relacionadas
 * con
 * el mantenimiento de clientes y la gestión de facturas.
 *
 * Opciones disponibles:
 * 1 → Mantenimiento de clientes
 * 2 → Introducción de facturas
 * 3 → Listado de facturas
 * 4 → Finalizar
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class MenuVehiculos {

    /**
     * Constructor por defecto de MenuVehiculos.
     */
    public MenuVehiculos() {

    }

    /**
     * Devuelve un mensaje en función de la opción seleccionada por
     * el usuario.
     *
     * @param opcion Número que representa la opción elegida del
     * menú.
     * @return Mensaje correspondiente a la opción seleccionada.
     */
}
```

```

public static String seleccionarOpcion(int opcion) {
    switch (opcion) {
        case 1:
            return "Opción seleccionada - Mantenimiento de
clientes";
        case 2:
            return "Opción seleccionada - Introducción de
facturas";
        case 3:
            return "Opción seleccionada - Listado de facturas";
        case 4:
            return "Opción seleccionada - Finalizar";
        default:
            return "La opción no existe. Vuelve a intentarlo.";
    }
}

/**
 * Método principal. Muestra el menú por consola, pide una opción
al usuario
 * y muestra el mensaje correspondiente hasta que elige salir.
 *
 * @param args Argumentos de línea de comandos (no se usan).
 */
public static void main(String[] args) {
    java.util.Scanner scanner = new
java.util.Scanner(System.in);
    int opcion;

    do {
        System.out.println("=== MENÚ DE MANTENIMIENTO DE
VEHÍCULOS ===");
        System.out.println("1. Mantenimiento de clientes");
        System.out.println("2. Introducción de facturas");
        System.out.println("3. Listado de facturas");
        System.out.println("4. Finalizar");
        System.out.print("Seleccione una opción: ");

        opcion = scanner.nextInt();
        System.out.println(seleccionarOpcion(opcion));
        System.out.println();

    } while (opcion != 4);

    System.out.println("Gracias por usar la aplicación. ¡Hasta
luego!");
    scanner.close(); // Cierra el Scanner para liberar recursos
}

```

```
}
```

Ejercicio 2. Diseñar un programa que muestre el producto de los 10 primeros números impares. Clase: com.ieschabas.NumerosImpares Método: public long productoDiezPrimerosImpares(); Detalles: - Calcula $1*3*5*...*19 = 654729075$

```
package com.ieschabas;

/**
 * Programa que calcula el producto de los diez primeros números
 * impares.
 *
 * Multiplica los números 1, 3, 5, ..., 19 y muestra el resultado por
 * consola.
 * El resultado final es 654729075.
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class NumerosImpares {

    /**
     * Constructor por defecto de NumerosImpares.
     */
    public NumerosImpares() {

    }

    /**
     * Calcula el producto de los diez primeros números impares.
     *
     * Empieza en 1 y va sumando 2 en cada paso hasta completar los
     * diez primeros impares.
     *
     * @return El producto de los diez primeros números impares.
     */
    public static long productoDiezPrimerosImpares() {
        long producto = 1; // Acumula el producto total
        int contador = 0; // Cuenta cuántos impares se han
multiplicado
        int numero = 1; // Primer número impar

        // Calcula el producto de los 10 primeros números impares
        while (contador < 10) {
            producto *= numero; // Multiplica el número actual
            numero += 2; // Avanza al siguiente número impar
            contador++; // Aumenta el contador
        }
    }
}
```

```

        return producto;
    }

    /**
     * Método principal. Muestra por consola el resultado
     * del producto de los diez primeros números impares.
     *
     * @param args Argumentos de línea de comandos (no se usan).
     */
    public static void main(String[] args) {
        System.out.println(productoDiezPrimerosImpares()); //
Resultado esperado: 654729075
    }
}

```

Ejercicio 3. Realizar un juego para adivinar un número. Clase:

com.ieschabas.JuegoAdivinar Métodos: public void evaluarIntento(int secreto, int intento, int [] intentos); public int jugar(int secreto, int[] intentos); - evaluarIntento: muestra por pantalla el mensaje correspondiente: -Si intento < secreto, el mensaje es "El número es menor. Has realizado n intentos", donde n será el número de intentos hechos por el usuario - Si intento > secreto, el mensaje es "El número es mayor. Has realizado n intentos", donde n será el número de intentos hechos por el usuario - Si intento = secreto, el mensaje es "Has acertado!" - jugar: devuelve el número de intentos hasta acertar o -1 si no se acierta.

```

package com.ieschabas;

/**
 * Programa que simula un juego de adivinanza de números.
 *
 * El jugador intenta adivinar un número secreto. El programa indica
 * si el número
 * buscado es mayor, menor o si se ha acertado.
 * También puede simular una partida con una lista de intentos
 * predefinidos.
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class JuegoAdivinar {

    /**
     * Constructor por defecto de JuegoAdivinar.
     */
    public JuegoAdivinar() {

    }
}

```

```

/**
 * Evalúa un intento del jugador comparando el número introducido
 * con el número secreto. Muestra un mensaje indicando si el
número
 * buscado es mayor, menor o si ha sido acertado.
 *
 * @param secreto Número que el jugador debe adivinar.
 * @param intento Número introducido por el jugador en un
intento.
 */
public static void evaluarIntento(int secreto, int intento) {
    if (intento < secreto) {
        System.out.println("El número es mayor.");
    } else if (intento > secreto) {
        System.out.println("El número es menor.");
    } else {
        System.out.println(";Has acertado!");
    }
}

/**
 * Simula una partida del juego de adivinanza.
 * Recorre una lista de intentos y devuelve el número de intentos
 * realizados hasta adivinar el número secreto.
 * Si no se acierta, devuelve -1.
 *
 * @param secreto Número que el jugador debe adivinar.
 * @param intentos Array con los números introducidos por el
jugador.
 * @return Número de intentos necesarios para acertar, o -1 si no
se acierta.
 */
public static int jugar(int secreto, int[] intentos) {
    for (int i = 0; i < intentos.length; i++) {
        int intento = intentos[i];
        int numIntento = i + 1; // Contador de intentos
(comienza en 1)

        if (intento == secreto) {
            System.out.println(";Has acertado!");
            return numIntento;
        } else if (intento < secreto) {
            System.out.println("El número es mayor. Has
realizado " + numIntento + " intentos.");
        } else {
            System.out.println("El número es menor. Has
realizado " + numIntento + " intentos.");
        }
    }
}

```

```

    }

    return -1; // No se acertó el número
}

/**
 * Método principal que ejecuta una partida de prueba del juego
de adivinanza.
 * Define un número secreto y una lista de intentos, y muestra
los resultados en consola.
 *
 * @param args Argumentos de línea de comandos (no se usan).
 */
public static void main(String[] args) {
    int numeroSecreto = 27; // Número que el jugador debe
adivinar
    int[] misIntentos = {10, 50, 20, 30, 27}; // Intentos de
ejemplo

    int resultado = jugar(numeroSecreto, misIntentos);

    System.out.println("Número de intentos hasta acertar: " +
resultado);
}
}

```

Ejercicio 4. Realiza un programa que obtenga un número par aleatorio menor que el que el usuario introduzca. Clase: com.ieschabas.AleatoriosPar Método: public int parAleatorioMenorQue(int limite, Random rng); Detalles: - Devuelve un par aleatorio $0 \leq x < \text{limite}$, $x \% 2 == 0$. - Si $\text{limite} \leq 0 \rightarrow$ Muestra un mensaje al usuario “El límite no puede ser un número negativo” y vuelve a solicitar el número

```

package com.ieschabas;

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

/**
 * Programa que genera un número par aleatorio menor que un límite
dado.
 *
 * El usuario introduce un número límite y el programa genera un
número par aleatorio
 * comprendido entre 0 (inclusive) y dicho límite (exclusive).
 *
 * Si el límite es menor o igual que 0, se muestra un mensaje de
error y se solicita nuevamente la entrada.

```

```

*
* Autor: Héctor Crespo
* Versión: 1.0
*/
public class AleatoriosPar {

    /**
     * Constructor por defecto de AleatoriosPar.
     */
    public AleatoriosPar(){

    }

    /**
     * Genera un número par aleatorio menor que el límite
    especificado.
     *
     * @param limite Valor máximo (no incluido) hasta el cual se
    puede generar el número par.
     *
     * Debe ser mayor que 0.
     * @param rng Instancia de {@link java.util.Random} utilizada
    para la generación aleatoria.
     * @return Un número par aleatorio menor que el límite.
     * @throws IllegalArgumentException Si el límite es menor o igual
    que 0.
     */

    public static int parAleatorioMenorQue(int limite, Random rng)
    {
        if (limite <= 0) {
            throw new IllegalArgumentException("El límite debe ser
    mayor que 0.");
        }

        int maxPar = limite - 1;
        if (maxPar % 2 != 0) {
            maxPar--; // Ajusta al número par más cercano inferior
        }

        if (maxPar < 0) {
            return 0;
        }

        int numPares = (maxPar / 2) + 1;
        int indice = rng.nextInt(numPares);
        return indice * 2;
    }

    /**

```

```

    * Método principal que solicita un número límite al usuario,
    * valida la entrada y muestra un número par aleatorio menor que
    dicho límite.
    *
    * @param args Argumentos de línea de comandos (no utilizados).
    */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random rng = new Random();
        int limite;

        // Solicita un límite válido
        do {
            System.out.print("Introduce un número límite (debe ser
> 0): ");

            while (!scanner.hasNextInt()) {
                System.out.println("Entrada no válida. Por favor,
introduce un número entero.");
                scanner.next();
            }

            limite = scanner.nextInt();

            if (limite <= 0) {
                System.out.println("El límite no puede ser un
número negativo ni cero.");
            }

        } while (limite <= 0);

        // Generación y salida del resultado
        int parAleatorio = parAleatorioMenorQue(limite, rng);
        System.out.println("Número par aleatorio generado: " +
parAleatorio);

        scanner.close();
    }
}

```

Ejercicio 5. Escribe un algoritmo que devuelva la suma de dos enteros aleatorios (entre 0 y 1000), repetirlo 20 veces. Clase: com.ieschabas.SumasAleatorias Métodos: public int sumaDosAleatorios(Random rng); public int[] generarSumas(int repeticiones, Random rng); Detalles: - Usa rng.nextInt(1001) para cada sumando. - repeticiones <= 0 → Muestra un mensaje al usuario “Las repeticiones no puede ser un número negativo” y vuelve a solicitar el número


```

package com.ieschabas;

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

/**
 * Programa que calcula la suma de dos números enteros aleatorios
 * entre 0 y 1000.
 *
 * Permite generar varios pares de números aleatorios, sumar sus
 * valores y mostrar los resultados.
 * El número de repeticiones lo indica el usuario.
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class SumasAleatorias {

    /**
     * Constructor por defecto de SumasAleatorias.
     */
    public SumasAleatorias() {

    }

    /**
     * Genera dos números aleatorios entre 0 y 1000 y devuelve su
     * suma.
     *
     * @param rng Objeto de tipo Random utilizado para generar los
     * números aleatorios.
     * @return La suma de dos números aleatorios entre 0 y 1000.
     */
    public static int sumaDosAleatorios(Random rng) {
        int a = rng.nextInt(1001); // Primer número aleatorio
        int b = rng.nextInt(1001); // Segundo número aleatorio
        return a + b;              // Devuelve la suma
    }

    /**
     * Genera un array con los resultados de varias sumas aleatorias.
     *
     * @param repeticiones Número de sumas que se deben generar.
     * @param rng Objeto Random para generar los números aleatorios.
     * @return Un array con las sumas generadas.
     */
    public static int[] generarSumas(int repeticiones, Random rng)
{

```

```

        int[] resultados = new int[repeticiones];

        // Genera tantas sumas como indique el usuario
        for (int i = 0; i < repeticiones; i++) {
            resultados[i] = sumaDosAleatorios(rng);
        }

        return resultados;
    }

    /**
     * Método principal. Pide al usuario cuántas sumas desea generar,
     * calcula las sumas aleatorias y muestra los resultados por
     * consola.
     *
     * @param args Argumentos de línea de comandos (no se usan).
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random rng = new Random();
        int repeticiones;

        // Solicita y valida el número de repeticiones
        do {
            System.out.print("Introduce el número de repeticiones
(DEBE SER 20 SEGÚN EL ENUNCIADO): ");

            // Comprueba que el usuario introduzca un número entero
            while (!scanner.hasNextInt()) {
                System.out.println("Entrada no válida. Por favor,
introduce un número entero.");
                scanner.next(); // Descarta la entrada inválida
            }

            repeticiones = scanner.nextInt();

            // Verifica que el número sea positivo
            if (repeticiones <= 0) {
                System.out.println("Las repeticiones no pueden ser
negativas ni cero.");
            }

        } while (repeticiones <= 0);

        // Genera las sumas aleatorias
        int[] sumas = generarSumas(repeticiones, rng);

        // Muestra los resultados
    }

```

```

        System.out.println("\nResultados de las " + repeticiones +
" sumas aleatorias:");
        for (int suma : sumas) {
            System.out.println(suma);
        }

        scanner.close();
    }
}

```

Ejercicio 6. Calcula el área y la longitud de una circunferencia en función del radio. Clase: com.ieschabas.Circunferencia Métodos: public double area(double radio); public double longitud(double radio); Detalles: - Área = $\pi \cdot r^2$ - Longitud = $2 \cdot \pi \cdot r$ - radio < 0 → Muestra un mensaje al usuario “El radio no puede ser un número negativo” y vuelve a solicitar el número

```

package com.ieschabas;

import java.util.Scanner;

/**
 * Programa que calcula el área y la longitud de una circunferencia.
 *
 * Permite al usuario introducir el radio de una circunferencia y
 * calcula:
 *
 * Área =  $\pi \cdot r^2$ 
 * Longitud =  $2 \cdot \pi \cdot r$ 
 *
 * Si el usuario introduce un valor negativo, se muestra un mensaje
 * de error
 * y se solicita nuevamente la entrada.
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class Circunferencia {

    /**
     * Constructor por defecto de Circunferencia.
     */
    public Circunferencia() {

    }

    /**
     * Calcula el área de una circunferencia a partir de su radio.

```

```

    *
    * @param radio Radio de la circunferencia (debe ser mayor o
igual que 0).
    * @return Área de la circunferencia.
    * @throws IllegalArgumentException Si el radio es negativo.
    */
    public static double area(double radio) {
        if (radio < 0) {
            throw new IllegalArgumentException("El radio no puede
ser un número negativo");
        }
        return Math.PI * radio * radio;
    }

    /**
     * Calcula la longitud (perímetro) de una circunferencia a partir
de su radio.
     *
     * @param radio Radio de la circunferencia (debe ser mayor o
igual que 0).
     * @return Longitud (perímetro) de la circunferencia.
     * @throws IllegalArgumentException Si el radio es negativo.
     */
    public static double longitud(double radio) {
        if (radio < 0) {
            throw new IllegalArgumentException("El radio no puede
ser un número negativo");
        }
        return 2 * Math.PI * radio;
    }

    /**
     * Método principal que solicita al usuario el radio de una
circunferencia,
     * valida la entrada y muestra el área y la longitud
correspondientes.
     *
     * @param args Argumentos de línea de comandos (no se utilizan).
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double radio;

        // Solicitud y validación de entrada
        do {
            System.out.print("Introduce el radio de la
circunferencia (debe ser >= 0): ");

```

```

        while (!scanner.hasNextDouble()) {
            System.out.println("Entrada no válida. Por favor,
introduce un número.");
            scanner.next();
        }

        radio = scanner.nextDouble();

        if (radio < 0) {
            System.out.println("El radio no puede ser un número
negativo.");
        }

    } while (radio < 0);

    // Cálculos
    double area = area(radio);
    double longitud = longitud(radio);

    // Salida formateada
    System.out.printf("Área: %.2f%n", area);
    System.out.printf("Longitud: %.2f%n", longitud);

    scanner.close();
}
}

```

Ejercicio 7. Escribe un algoritmo que devuelva el mayor de dos números. Clase: com.ieschabas.Mayores Método: public int max2(int a, int b);

```

package com.ieschabas;

import java.util.Scanner;

/**
 * Programa que determina el mayor de dos números enteros
 * introducidos por el usuario.
 *
 * El usuario ingresa dos números, y el programa muestra cuál de
 * ellos es mayor.
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class Mayores {

    /**

```

```

    * Constructor por defecto de Mayores.
    */
public Mayores() {

}

/**
 * Devuelve el mayor de dos números enteros.
 *
 * @param a Primer número.
 * @param b Segundo número.
 * @return El número mayor entre a y b.
 */
public static int max2(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

/**
 * Método principal. Solicita dos números al usuario,
 * calcula cuál es el mayor y muestra el resultado por consola.
 *
 * @param args Argumentos de línea de comandos (no se usan).
 */
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce el primer número: ");
    int num1 = scanner.nextInt();

    System.out.print("Introduce el segundo número: ");
    int num2 = scanner.nextInt();

    int mayor = max2(num1, num2);

    System.out.println("El mayor de " + num1 + " y " + num2 + "
es: " + mayor);

    scanner.close();
}
}

```

Ejercicio 8. Escribe un algoritmo que devuelva el mayor de tres números aleatorios (entre 0 y 1000), repetirlo 20 veces. Clase: com.ieschabas.MayoresAleatorios3 Métodos: public int

max3(int a, int b, int c); public int[] generarMaximos3(int repeticiones, Random rng);
Detalles: - Usa rng.nextInt(1001) para cada valor. - repeticiones <= 0 → Muestra un mensaje al usuario “Las repeticiones no pueden ser un número negativo” y vuelve a solicitar el número

```
package com.ieschabas;

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

/**
 * Programa que calcula el mayor de tres números aleatorios entre 0 y 1000.
 *
 * Genera grupos de tres números aleatorios y determina el mayor de cada grupo.
 * El proceso se repite tantas veces como indique el usuario.
 *
 * Autor: Héctor Crespo
 * Versión: 1.0
 */
public class MayoresAleatorios3 {

    /**
     * Constructor por defecto de MayoresAleatorios3.
     */
    public MayoresAleatorios3() {

    }

    /**
     * Devuelve el mayor de tres números enteros.
     *
     * @param a Primer número.
     * @param b Segundo número.
     * @param c Tercer número.
     * @return El número mayor entre a, b y c.
     */
    public static int max3(int a, int b, int c) {
        int mayor = a; // Se asume que el primero es el mayor

        if (b > mayor) {
            mayor = b;
        }
        if (c > mayor) {
            mayor = c;
        }
    }
}
```

```

        return mayor;
    }

    /**
     * Genera un array que contiene los valores máximos de varios
    grupos
     * de tres números aleatorios entre 0 y 1000.
     *
     * @param repeticiones Número de grupos (tríos) a generar.
     * @param rng Objeto Random usado para generar los números
    aleatorios.
     * @return Un array con los valores máximos de cada trío.
     */
    public static int[] generarMaximos3(int repeticiones, Random
    rng) {
        int[] resultados = new int[repeticiones];

        // Genera los tríos y calcula el máximo de cada uno
        for (int i = 0; i < repeticiones; i++) {
            int a = rng.nextInt(1001);
            int b = rng.nextInt(1001);
            int c = rng.nextInt(1001);

            resultados[i] = max3(a, b, c);
        }

        return resultados;
    }

    /**
     * Método principal. Pide al usuario el número de repeticiones,
     * genera los máximos de cada trío aleatorio y los muestra por
    consola.
     *
     * @param args Argumentos de línea de comandos (no se usan).
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random rng = new Random();
        int repeticiones;

        // Solicita y valida el número de repeticiones
        do {
            System.out.print("Introduce el número de repeticiones
    (DEBE SER 20 PARA ESTE EJERCICIO): ");

            while (!scanner.hasNextInt()) {

```



```

        System.out.println("Entrada no válida. Por favor,
introduce un número entero.");
        scanner.next(); // Descarta la entrada no válida
    }

    repeticiones = scanner.nextInt();

    if (repeticiones <= 0) {
        System.out.println("Las repeticiones deben ser un
número positivo.");
    }

    } while (repeticiones <= 0);

    // Genera los valores máximos
    int[] maximos = generarMaximos3(repeticiones, rng);

    // Muestra los resultados
    System.out.println("\nMayores de cada uno de los " +
repeticiones +
        " tríos de números aleatorios (0-1000):");
    for (int max : maximos) {
        System.out.println(max);
    }

    scanner.close(); // Cierra el Scanner
}
}

```

Ejercicio 9. Diseña un algoritmo que devuelva el mayor de cuatro números aleatorios (entre 0 y 1000), repetirlo 20 veces. Clase: com.ieschabas.MayoresAleatorios4 Métodos: public int max4(int a, int b, int c, int d); public int[] generarMaximos4(int repeticiones, Random rng); Detalles: - Usa rng.nextInt(1001) para cada valor. - repeticiones <= 0 → Muestra un mensaje al usuario “Las repeticiones no pueden ser un número negativo” y vuelve a solicitar el número

```

package com.ieschabas;

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

/**
 * Programa que calcula el mayor de cuatro números aleatorios entre 0
 * y 1000.
 *
 * Genera grupos de cuatro números aleatorios y determina el mayor de
 * cada grupo.
 */

```

```

* El proceso se repite tantas veces como indique el usuario.
*
* Autor: Héctor Crespo
* Versión: 1.0
*/
public class MayoresAleatorios4 {

    /**
     * Constructor por defecto de MayoresAleatorios4.
     */
    public MayoresAleatorios4(){

    }

    /**
     * Devuelve el mayor de cuatro números enteros.
     *
     * @param a Primer número.
     * @param b Segundo número.
     * @param c Tercer número.
     * @param d Cuarto número.
     * @return El número mayor entre a, b, c y d.
     */
    public static int max4(int a, int b, int c, int d) {
        int mayor = a; // Se asume que el primero es el mayor

        // Compara con los demás números
        if (b > mayor) {
            mayor = b;
        }
        if (c > mayor) {
            mayor = c;
        }
        if (d > mayor) {
            mayor = d;
        }

        return mayor;
    }

    /**
     * Genera un array que contiene los valores máximos de varios
    grupos
     * de cuatro números aleatorios entre 0 y 1000.
     *
     * @param repeticiones Número de grupos (cuartetos) a generar.
     * @param rng Objeto Random usado para generar los números
    aleatorios.

```

```

    * @return Un array con los valores máximos de cada grupo.
    */
    public static int[] generarMaximos4(int repeticiones, Random
rng) {
        int[] resultados = new int[repeticiones];

        // Genera los grupos y calcula el máximo de cada uno
        for (int i = 0; i < repeticiones; i++) {
            int a = rng.nextInt(1001);
            int b = rng.nextInt(1001);
            int c = rng.nextInt(1001);
            int d = rng.nextInt(1001);

            resultados[i] = max4(a, b, c, d);
        }

        return resultados;
    }

    /**
     * Método principal. Pide al usuario el número de repeticiones,
     * genera los máximos de cada grupo aleatorio y los muestra por
     * consola.
     *
     * @param args Argumentos de línea de comandos (no se usan).
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random rng = new Random();
        int repeticiones;

        // Solicita y valida el número de repeticiones
        do {
            System.out.print("Introduce el número de repeticiones
(DEBEN SER 20 COMO INDICA EL ENUNCIADO): ");

            while (!scanner.hasNextInt()) {
                System.out.println("Entrada no válida. Por favor,
introduce un número entero.");
                scanner.next(); // Descarta la entrada no válida
            }

            repeticiones = scanner.nextInt();

            if (repeticiones <= 0) {
                System.out.println("Las repeticiones deben ser un
número positivo.");
            }
        }
    }

```

```
    } while (repeticiones <= 0);

    // Genera los valores máximos
    int[] maximos = generarMaximos4(repeticiones, rng);

    // Muestra los resultados
    System.out.println("\nMayores de cada quarteto de números
aleatorios (0-1000):");
    for (int valor : maximos) {
        System.out.println(valor);
    }

    scanner.close(); // Cierra el Scanner
}
}
```