Ejercicio ejercicios calificable unidad 3

Este examen está compuesto por **4 ejercicios prácticos de Java** que evalúan:

- Clases y objetos.
- Encapsulamiento y métodos.
- Estructuras de control (condicionales y bucles).
- Manejo de excepciones.
- Uso de tipos básicos y colecciones.

La **nota máxima del examen es 10 puntos**.
Cada ejercicio aporta una parte de esa nota según la ponderación indicada.

📊 Distribución de puntos del examen (sobre 10)

- **Ejercicio 1 – Alumno → 2 puntos**
- **Ejercicio 2 – CalculadoraEstadistica → 2,5 puntos**
- **Ejercicio 3 – JuegoNotas → 2,5 puntos**
- **Ejercicio 4 – AnalizadorNumerico → 2,5 puntos**

> **Total: 10 puntos**

🧩 Ejercicio 1 – Clase `Alumno` (2 puntos)

Paquete: `org.docencia.unidad3.examen.ejercicio01`
Clase principal: `Alumno`

Qué se pide

Implementar una clase `Alumno` que modele a un estudiante con:

- Atributos:
  - `String nombre`
  - `int nota` (0–10)
- Constructores:
  - Vacío: `nombre = null`, `nota = 0`.
  - Completo: recibe `nombre` y `nota` (validando 0–10).
- Getters y setters:
  - Validación en `setNota(int nota)`:
    - Si `nota < 0 ó nota > 10` → `IllegalArgumentException`.
- Lógica adicional:
  - `String getCalificacionTexto()`:
    - 0–4 → `"Insuficiente"`
    - 5 → `"Suficiente"`
    - 6 → `"Bien"`
    - 7–8 → `"Notable"`

    - 9–10 → `"Sobresaliente"`
- Métodos sobrescritos:
    - `toString()` mostrando nombre, nota y calificación texto.
    - `equals()` y `hashCode()` basados en `nombre` y `nota`.

🧱 Ejercicio 2 – `CalculadoraEstadistica` (2,0 puntos)

Paquete: `org.example.unidad3.examen.ejercicio02`
Clase principal: `CalculadoraEstadistica`
Clase auxiliar interna: `Estadisticas`

Qué se pide

1. **Calculadora básica con menú lógico:**

   ```
   double operar(double a, double b, int opcion)
   ```

    - `1` → suma (`a + b`)
    - `2` → resta (`a - b`)
    - `3` → multiplicación (`a * b`)
    - `4` → división (`a / b`)

   Validaciones:

    - `opcion` fuera de `[1,4]` → `IllegalArgumentException`.
    - `opcion == 4` y `b == 0` → `ArithmeticException` (división entre cero).

2. **Cálculo de estadísticas:**

   ```
   Estadisticas calcularEstadisticas(double[] numeros)
   ```

    - Entrada: array de `double`.
    - Salida: objeto `Estadisticas` con:
        - `int cantidad`
        - `double media`
        - `double maximo`
        - `double minimo`
    - Si `numeros` es `null` o está vacío → `IllegalArgumentException`.

3. **Clase interna `Estadisticas`:**

    - Atributos inmutables (`final` vía constructor).
    - Getters para todos los campos.
    - `toString()`, `equals()` y `hashCode()` implementados.

🎯 Ejercicio 3 – `JuegoNotas` (1,5 puntos)

Paquete: `org.example.unidad3.examen.ejercicio03`
Clase principal: `JuegoNotas`

Qué se pide

1. **Cálculo de la media de un conjunto de notas:**

```
double calcularMedia(int[] notas)
```

   - Devuelve la media aritmética de las notas.
   - Si el array es `null` o está vacío → `IllegalArgumentException`.

2. **Evaluar un intento de adivinar la media:**

```
String evaluarIntento(double mediaReal, double intento)
```

   - Devuelve:
     - `"Demasiado alta"` si `intento > mediaReal`.
     - `"Demasiado baja"` si `intento < mediaReal`.
     - `"¡Correcta!"` si `intento == mediaReal` (comparación con `Double.compare` o similar).

🧱 Ejercicio 4 – `AnalizadorNumerico` (2,0 puntos)

Paquete: `org.example.unidad3.examen.ejercicio04`
Clase principal: `AnalizadorNumerico`
Clases auxiliares internas: `ContadorSignos`, `ParesImpares`

Qué se pide

1. **Contar positivos, negativos y ceros:**

```
ContadorSignos contarSignos(int[] numeros)
```

   - Devuelve un objeto con:
     - `positivos`
     - `negativos`
     - `ceros`
   - Si el array es `null` → `IllegalArgumentException`.
   - Si está vacío → todos los contadores a 0.

2. **Separar pares e impares en un rango:**

```
ParesImpares calcularParesImpares(int a, int b)
```

- Considera el intervalo `[a, b]` (incluidos).
- Si `a > b`, se intercambian.
- Rellena:
  - Lista de pares.
  - Lista de impares.
- Los getters devuelven listas inmodificables.

3. **Contar números primos en un rango:**

```
int contarPrimos(int a, int b)
```

- Cuenta cuántos primos hay en `[a, b]`.
- Si `a > b`, se intercambian.
- Solo números > 1 se consideran candidatos.
- Usa un método auxiliar `esPrimo(int n)` hasta `sqrt(n)`.

4. **Clases auxiliares:**

- `ContadorSignos`: enteros inmutables; getters, `toString`, `equals`, `hashCode`.
- `ParesImpares`: listas copiadas en el constructor; getters inmodificables; `toString`, `equals`, `hashCode`.