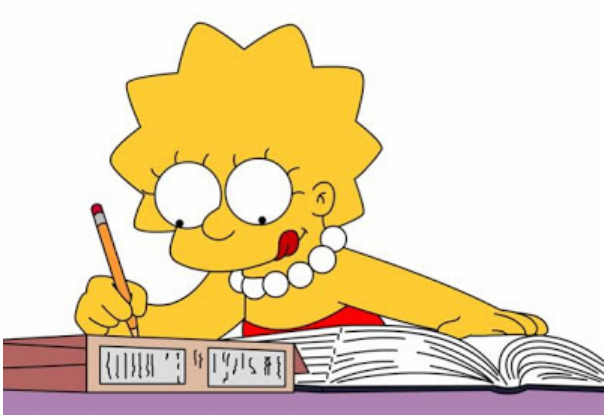
Ejercicios del ejercicio calificable 3.2



Ejecutar los tests

Si quieres ver cómo funcionan los tests, ejecuta en el directorio del proyecto:

```
jpexposito@MacBook-Air-de-Joatham ejercicios-repaso % ls -la
total 120
drwxr-xr-x@  9 jpexposito  staff    288 13 dic 11:35 .
drwx------@ 52 jpexposito  staff   1664 13 dic 11:40 ..
-rw-r--r--@  1 jpexposito  staff   6148 13 dic 11:07 .DS_Store
-rw-r--r--@  1 jpexposito  staff     32 13 dic 11:05 .gitignore
drwxr-xr-x@  3 jpexposito  staff     96 13 dic 11:35 images
-rw-r--r--@  1 jpexposito  staff   1278 13 dic 11:05 pom.xml
-rw-r--r--@  1 jpexposito  staff  42122 13 dic 11:46 README.md
drwxr-xr-x@  5 jpexposito  staff    160 13 dic 11:07 src
drwxr-xr-x@  8 jpexposito  staff    256 13 dic 11:32 target
```

Ejecuta:

```
mvn clean test
```

```
[ERROR]   Ejercicio16Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio17Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio18Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio19Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio2Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio20Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio3Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio4Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio5Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio6Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio7Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio8Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[ERROR]   Ejercicio9Test.debeResolverElEjercicio:24 » UnsupportedOperation No implementado
[INFO]
[ERROR] Tests run: 264, Failures: 123, Errors: 74, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

Puedes ir ejecutando los tests de cada ejercicio:

```
Ejercicio1Test.java
src > test > java > com > docencia > condiciones > ejercicio1 > Ejercicio1Test.java > { } com.docencia.condiciones.ejercicio1
1  package com.docencia.condiciones.ejercicio1;
2
3  import static org.junit.jupiter.api.Assertions.*;
4  import org.junit.jupiter.api.Test;
5
6  class Ejercicio1Test {
7      @Test
8      void caso1() {
9          assertEquals(expected: "SUSPENSO", Ejercicio1.clasificarNota(nota: 4));
10     }
11
12     @Test
13     void caso2() {
14         assertEquals(expected: "APROBADO", Ejercicio1.clasificarNota(nota: 5));
15     }
16
17     @Test
18     void caso3() {
19         assertEquals(expected: "NOTABLE", Ejercicio1.clasificarNota(nota: 7));
```

0) Antes de Comenzar

NORMALIZADO -> Se eliminan espacios, etc.

1) Ejercicio sobre clases (`com.docencia.clases`)

En cada ejercicio hay una clase del dominio con:

- constructor vacío
- constructor con identificador único
- `equals`, `hashCode` y `toString`
- **Ejercicio 10**: implementar `equals/hashCode/toString` en `Mascota` usando `chip` como identificador único.

> **IMPORTANTE**: puede parecer repetitivo (y lo es), pero `constructores/getters/setters/equals/hashCode/toString` es una base fundamental en **Java**.

2) Bucles `for` sobre arrays (`com.docencia.arrays`)

- **Ejercicio 7**: `sonIguales(int[] array1, int[] array2)` → Devuelve `true` si ambos arrays tienen la misma longitud y mismos valores en cada posición.

3) Ejercicio sobre listas (`com.docencia.listas`)

Ejercicio 16

Listas — Sumar longitudes de cadenas no vacías (`com.docencia.listas.ejercicio16.Ejercicio16`)

- Clase: `com.docencia.listas.ejercicio16.Ejercicio16`
- Método:

```
public static int sumarLongitudesNoVacias(java.util.List<String>
textos)
```

- Enunciado:
  Sumar la longitud de todas las cadenas **no nulas y no en blanco**.
  Si lista `null` o sin cadenas válidas → `0`.

4) Ejercicio sobre herencia (`com.docencia.herencia`)

Herencia — Documentos y descripción (`com.docencia.herencia.ejercicio14.Ejercicio14`)

Implementar una jerarquía simple de documentos y una funcionalidad común para obtener sus descripciones, aplicando **programación defensiva** ante datos inválidos.

- Jerarquía:
  - `Documento` (abstracta) — título → `String descripcion()`.

    - Si `titulo` es `null` o `vacío` → lanzar `IllegalArgumentException`
  - `Informe` — páginas
    - Si `paginas <= 0` → lanzar `IllegalArgumentException`
    - 'String descripcion()'. Debe devolver una descripción que incluya el título y el número de páginas
      (el formato exacto será el verificado por los tests).
  - `Carta` — destinatario
    - Si `destinatario` es `null` o `vacío` → lanzar `IllegalArgumentException`
    - `String descripcion()`. Debe devolver una descripción que incluya el título y el destinatario
      (el formato exacto será el verificado por los tests).

Tipo	Condición	Formato exacto de `descripcion()`	Ejemplo (según tests)
Carta	Siempre	`Carta: <TITULO_NORMALIZADO> | Para: <DESTINATARIO_NORMALIZADO>`	`Carta: Mi carta | Para: Ana`
Informe	Siempre	`Informe: <TITULO_NORMALIZADO> (<PAGINAS> páginas)`	`Informe: Informe trimestral (12 páginas)`

---

5) Ejercicio sobre composición (`com.docencia.composicion`)

Composición — Inventario de productos (`com.docencia.composicion.ejercicio9.Ejercicio9`)

- Clases:
  - `Producto` — nombre.
  - `LineaInventario` — producto y cantidad.
  - `Inventario` — lista de líneas.
- Funcionalidad:
  - `void anadirStock(String nombreProducto, int cantidad)` → solo si nombre válido y cantidad > 0; si existe producto (nombre case-insensitive + `trim`) suma stock, si no crea nueva línea.
  - `boolean retirarStock(String nombreProducto, int cantidad)` → retira si hay stock suficiente y datos válidos.
  - `int stockDe(String nombreProducto)` → stock del producto (si nombre inválido/no existe → `0`).
  - `int totalUnidades()` → suma de cantidades de todas las líneas.