# TAREA 5.2b - Definición y ejecución de tests unitarios con Pytest

Vas a trabajar con una clase llamada `Cartera` (archivo `cartera.py`). Esta clase simula una cartera con un **saldo** y permite **ingresar** y **gastar** dinero.

Tu tarea es **crear tests unitarios con pytest** para comprobar que la clase se comporta correctamente en distintos casos.

cartera.py
Text File
968 B

## Antes de empezar

1. Descarga el archivo `cartera.py` y colócalo en una carpeta de proyecto.
2. Crea un archivo nuevo llamado `test_cartera.py` en la misma carpeta.
3. Asegúrate de tener instalado pytest:

```
pip install pytest
```

Para ejecutar los tests usarás:

```
pytest
```

## Importante: comportamiento cuando algo no es válido

- Si se intenta usar un valor incorrecto (tipo no entero o cantidad negativa), el método devuelve `None` y **no cambia el saldo**.
- Si se intenta gastar más dinero del disponible, también devuelve `None` y el saldo se mantiene igual.

📌 Esto es lo que debes comprobar en algunos tests.

## Qué debes comprobar (mínimo 1 test por punto)

Crea **al menos un test unitario** para cada uno de estos casos:

### Constructor (`Cartera(...)`)

1. **Saldo inicial por defecto**

   Comprueba que, si creas una cartera sin indicar saldo, el saldo inicial es **0**.

2. **Tipo incorrecto en el constructor**

   Comprueba que, si al constructor le pasas un tipo incorrecto (por ejemplo un texto), el saldo se queda en **0**.

3. **Saldo negativo en el constructor**

   Comprueba que, si al constructor le pasas un saldo negativo, el saldo se pone en **0**.

4. **Saldo inicial válido (entero positivo)**

   Comprueba que, si al constructor le pasas un entero positivo, el saldo inicial se asigna correctamente.

### Método `ingresar(cantidad)`

1. **Ingresar dinero suma correctamente**

   Comprueba que, al ingresar dinero, el método devuelve el **nuevo saldo** (saldo anterior + ingreso) y el saldo interno queda actualizado.

*(Opcional recomendado)*: comprueba que si se intenta ingresar una cantidad no válida, devuelve `None` y no cambia el saldo.

### Método `gastar(cantidad)`

1. **Gastar dinero resta correctamente**

Comprueba que, al gastar dinero, el método devuelve el **nuevo saldo** (saldo anterior − gasto) y el saldo interno queda actualizado.

2. **Gastar más del saldo disponible**

   Comprueba que, si se intenta gastar más dinero del que hay, el método devuelve `None` y el saldo no cambia.

## Condiciones de entrega

- Trabajo **individual**.
- Los tests deben estar en un archivo llamado `test_cartera.py` incluyendo tu nombre en comentarios.
- Cada test debe comprobar **una idea concreta**.
- Debes adjuntar captura de la terminal de VS Code donde se vea que todos los test pasan.

## Rúbrica

| Criterio de evaluación | Excelente | Adecuado | Básico / Insuficier |
|---|---|---|---|
| **1. Tests del constructor (`Cartera`)** | Incluye tests correctos para los **4 casos**: saldo por defecto, tipo incorrecto, saldo negativo y saldo positivo. Todos funcionan correctamente. | Incluye tests para **3 casos** del constructor o alguno es incompleto. | Incluye **m tests**, hay claros o fa important |
| **2. Test de ingreso de dinero** | Comprueba correctamente que `ingresar()` **devue lve el nuevo saldo** y que el saldo interno se actualiza. | El test existe pero solo comprueba el valor devuelto **o** el saldo interno. | El test es incomple existe. |
| **3. Test de gasto de dinero válido** | Comprueba correctamente que `gastar()` **resta el saldo**, devuelve el nuevo valor y actualiza el saldo interno. | El test existe pero no comprueba todos los aspectos (resultado y saldo). | El test es o no exist |
| **4. Test de gasto no válido (más del saldo)** | Comprueba que el método devuelve `None` y que el saldo **no cambia**. | Comprueba solo una de las dos cosas (`None` o saldo). | El test es o no exist |
| **5. Organización y claridad de los tests** | Nombres claros (`test_...`), un test por idea, código legible y bien estructurado. | Nombres mejorables o mezcla de comprobaciones en algún test. | Nombres tests poc desorden |
| **6. Cumplimiento de condiciones de entrega** | Archivo `test_cartera.py` correcto, trabajo original y **webgrafía incluida**. | Falta la webgrafía o hay pequeños fallos de formato. | Falta el ar correcto, o no cum condicior |