Agregamos la función vaciar en carrito

```
23
24  class Carrito:
25      def __init__(self):
26          self.items = []
27
28      def vaciar(self):
29          """
30          Elimina todos los items del carrito.
31          """
32          self.items = []
33
34      def agregar_producto(self, producto, cantidad=1):
35          """
```

Agregamos la función para testear el vaciado del carrito

```
189
190 def test_vaciar_carrito():
191     """
192     AAA:
193     Arrange: Se crea un carrito y se agregan varios productos.
194     Act:    Se invoca vaciar() en el carrito.
195     Assert: obtener_items() retorna [], calcular_total() retorna 0.
196     """
197     # Arrange
198     carrito = Carrito()
199     producto1 = ProductoFactory(nombre="Libro", precio=20.0)
200     producto2 = ProductoFactory(nombre="Bolígrafo", precio=2.5)
201     carrito.agregar_producto(producto1, cantidad=2)
202     carrito.agregar_producto(producto2, cantidad=3)
203
204     # Act
205     carrito.vaciar()
206
207     # Assert
208     assert carrito.obtener_items() == []
209     assert carrito.calcular_total() == 0
```

Testeamos

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\windows10\Desktop\actiividad8> pytest
============================================================================
platform win32 -- Python 3.12.5, pytest-8.3.5, pluggy-1.5.0
rootdir: C:\Users\windows10\Desktop\actiividad8
plugins: Faker-37.1.0
collected 10 items

Ejemplo\tests\test_carrito.py ..........

============================================================================
PS C:\Users\windows10\Desktop\actiividad8>
```

Agregamos la función aplicar_descuento_condicional a la clase carrito

```python
    def aplicar_descuento_condicional(self, porcentaje, minimo):
        """
        Aplica un descuento al total del carrito solo si supera el mínimo.
        Si el total es menor que el mínimo, retorna el total sin descuento.
        """
        if porcentaje < 0 or porcentaje > 100:
            raise ValueError("El porcentaje debe estar entre 0 y 100")

        total = self.calcular_total()
        if total >= minimo:
            descuento = total * (porcentaje / 100)
            return total - descuento
        return total
```

Agregamos los testeos y ejecutamos

```python
def test_descuento_condicional_aplicado_si_supera_minimo():
    """
    AAA:
    Arrange: Se crea un carrito con total mayor al mínimo requerido.
    Act: Se aplica un descuento condicional.
    Assert: Se verifica que el descuento se aplica correctamente.
    """
    # Arrange
    carrito = Carrito()
    producto = ProductoFactory(nombre="Laptop", precio=600.00)
    carrito.agregar_producto(producto, cantidad=1)
    # Act
    total_con_descuento = carrito.aplicar_descuento_condicional(15, minimo=500)
    # Assert
    assert total_con_descuento == 510.00  # 600 - 15%


def test_descuento_condicional_no_aplicado_si_no_supera_minimo():
    """
    AAA:
    Arrange: Se crea un carrito con total menor al mínimo requerido.
    Act: Se aplica un descuento condicional.
    Assert: Se verifica que el total permanece sin descuento.
    """
    # Arrange
    carrito = Carrito()
    producto = ProductoFactory(nombre="Mouse", precio=100.00)
    carrito.agregar_producto(producto, cantidad=1)
    # Act
    total_con_descuento = carrito.aplicar_descuento_condicional(15, minimo=500)
    # Assert
    assert total_con_descuento == 100.00
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\windows10\Desktop\actiividad8> pytest
========================================================================== test session starts ==========
platform win32 -- Python 3.12.5, pytest-8.3.5, pluggy-1.5.0
rootdir: C:\Users\windows10\Desktop\actiividad8
plugins: Faker-37.1.0
collected 12 items

Ejemplo\tests\test_carrito.py ............

========================================================================== 12 passed in 0.21s ==========
PS C:\Users\windows10\Desktop\actiividad8>
```

Le agregamos el atributo stock

```python
class Producto:
    def __init__(self, nombre, precio, stock):
        self.nombre = nombre
        self.precio = precio
        self.stock = stock  # Nuevo atributo

    def __repr__(self):
        return f"Producto({self.nombre}, {self.precio}, stock={self.stock})"
```

Y le agregamos la siguiente función a la clase carrito

```python
def agregar_producto(self, producto, cantidad=1):
    """
    Agrega un producto al carrito. Si el producto ya existe, incrementa la cantidad.
    Verifica que la cantidad total no supere el stock.
    """
    cantidad_en_carrito = 0
    for item in self.items:
        if item.producto.nombre == producto.nombre:
            cantidad_en_carrito = item.cantidad
            break

    if cantidad_en_carrito + cantidad > producto.stock:
        raise ValueError(f"No hay suficiente stock disponible. Stock: {producto.stock}, en carrito: {cantidad_en_carrito}, intento de agregar: {cantidad}")

    for item in self.items:
        if item.producto.nombre == producto.nombre:
            item.cantidad += cantidad
            return

    self.items.append(ItemCarrito(producto, cantidad))
```

Agregamos el stock como un random entero del 1 al 100

factories.py | carrito.py | test_carrito.py

Ejemplo > src > factories.py > ...

```python
# src/factories.py
import factory
from .carrito import Producto

class ProductoFactory(factory.Factory):
    class Meta:
        model = Producto

    nombre = factory.Faker("word")
    precio = factory.Faker("pyfloat", left_digits=2, right_digits=2, positive=True)
    stock = factory.Faker("random_int", min=1, max=100)  # Nuevo
```

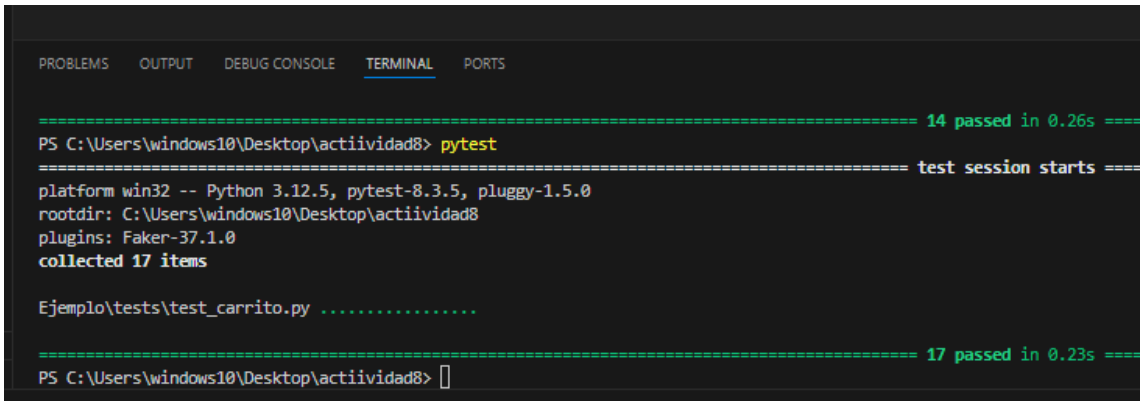
testeamos el código

```
PS C:\Users\windows10\Desktop\actiividad8> pytest
============================= test session starts =============================
platform win32 -- Python 3.12.5, pytest-8.3.5, pluggy-1.5.0
rootdir: C:\Users\windows10\Desktop\actiividad8
plugins: Faker-37.1.0
collected 14 items

Ejemplo\tests\test_carrito.py ..............

============================== 14 passed in 0.07s =============================
PS C:\Users\windows10\Desktop\actiividad8>
```

}

Agregamos la función (método) obtener_items_ordenados a la clase carrito

```python
def obtener_items_ordenados(self, criterio: str):
    """
    Retorna la lista de items ordenada según el criterio:
    - "precio": ordena por precio unitario del producto.
    - "nombre": ordena alfabéticamente por nombre del producto.
    """
    if criterio == "precio":
        return sorted(self.items, key=lambda item: item.producto.precio)
    elif criterio == "nombre":
        return sorted(self.items, key=lambda item: item.producto.nombre)
    else:
        raise ValueError("Criterio inválido. Usa 'precio' o 'nombre'.")
```



Creamos el archivo conftest.py

```python
# tests/conftest.py
import pytest
from src.carrito import Carrito
from src.factories import ProductoFactory

@pytest.fixture
def carrito():
    return Carrito()

@pytest.fixture
def producto_generico():
    return ProductoFactory(nombre="Genérico", precio=100.0, stock=10)
```

Ejecutamos el testeo

```
PS C:\Users\windows10\Desktop\actiividad8> pytest Ejemplo/tests
================================= test session starts =================================
platform win32 -- Python 3.12.5, pytest-8.3.5, pluggy-1.5.0
rootdir: C:\Users\windows10\Desktop\actiividad8
plugins: Faker-37.1.0
collected 21 items

Ejemplo\tests\test_carrito.py .....................                              [100%]

================================== 21 passed in 0.18s ==================================
PS C:\Users\windows10\Desktop\actiividad8>
```

Agregamos los testeos a test_carrito.py

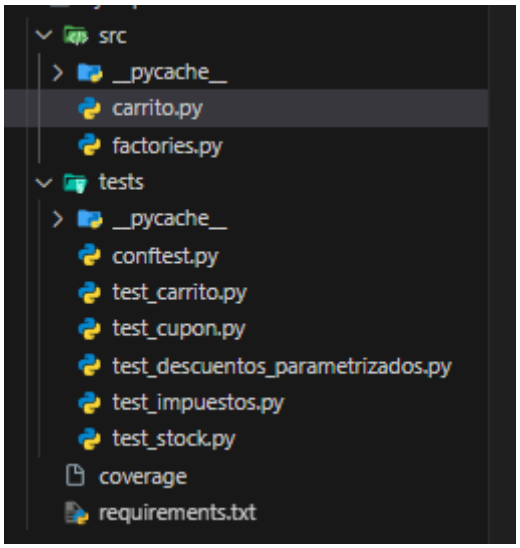
```python
def test_agregar_producto_al_carrito(carrito, producto_generico):
    carrito.agregar_producto(producto_generico, cantidad=2)
    assert len(carrito.items) == 1
    assert carrito.items[0].cantidad == 2


def test_no_agregar_mas_del_stock_disponible(carrito):
    producto = ProductoFactory(nombre="Limitado", precio=50.0, stock=3)
    carrito.agregar_producto(producto, cantidad=3)
    with pytest.raises(ValueError, match="No hay suficiente stock disponible"):
        carrito.agregar_producto(producto, cantidad=1)


def test_aplicar_descuento_condicional(carrito):
    producto = ProductoFactory(precio=300.0, stock=10)
    carrito.agregar_producto(producto, cantidad=2)
    total_con_descuento = carrito.aplicar_descuento_condicional(15, minimo=500)
    assert total_con_descuento == pytest.approx(510.0)


def test_aplicar_descuento_condicional_no_aplica(carrito):
    producto = ProductoFactory(precio=100.0, stock=10)
    carrito.agregar_producto(producto, cantidad=2)
    total = carrito.aplicar_descuento_condicional(15, minimo=500)
    assert total == 200.0


def test_obtener_items_ordenados_por_precio(carrito):
    p1 = ProductoFactory(nombre="A", precio=30.0, stock=10)
    p2 = ProductoFactory(nombre="B", precio=10.0, stock=10)
    p3 = ProductoFactory(nombre="C", precio=20.0, stock=10)

    carrito.agregar_producto(p1)
    carrito.agregar_producto(p2)
    carrito.agregar_producto(p3)

    items = carrito.obtener_items_ordenados("precio")
    precios = [item.producto.precio for item in items]
    assert precios == sorted(precios)
```

```python
def test_obtener_items_ordenados_por_precio(carrito):
    p1 = ProductoFactory(nombre="A", precio=30.0, stock=10)
    p2 = ProductoFactory(nombre="B", precio=10.0, stock=10)
    p3 = ProductoFactory(nombre="C", precio=20.0, stock=10)

    carrito.agregar_producto(p1)
    carrito.agregar_producto(p2)
    carrito.agregar_producto(p3)

    items = carrito.obtener_items_ordenados("precio")
    precios = [item.producto.precio for item in items]
    assert precios == sorted(precios)


def test_obtener_items_ordenados_por_nombre(carrito):
    p1 = ProductoFactory(nombre="Zanahoria", precio=10.0, stock=10)
    p2 = ProductoFactory(nombre="Manzana", precio=10.0, stock=10)
    p3 = ProductoFactory(nombre="Banana", precio=10.0, stock=10)

    carrito.agregar_producto(p1)
    carrito.agregar_producto(p2)
    carrito.agregar_producto(p3)

    items = carrito.obtener_items_ordenados("nombre")
    nombres = [item.producto.nombre for item in items]
    assert nombres == sorted(nombres)


def test_obtener_items_ordenados_con_criterio_invalido(carrito):
    with pytest.raises(ValueError, match="Criterio inválido"):
        carrito.obtener_items_ordenados("peso")
```

Agregamos varias funciones

```python
    def agregar_producto(self, producto, cantidad=1):
        """
        Agrega un producto al carrito, validando stock.
        """
        item = self._buscar_item(producto)
        cantidad_actual = item.cantidad if item else 0
        if cantidad_actual + cantidad > producto.stock:
            raise ValueError("Cantidad a agregar excede el stock disponible")
        if item:
            item.cantidad += cantidad
        else:
            self.items.append(ItemCarrito(producto, cantidad))
```

Creamos nuevos archivos .py para testeo



Y testetamos