

“Año del Bicentenario, de la consolidación de nuestra Independencia, de la conmemoración de las heroicas batallas de Junín y Ayacucho”



EXTRACT INTERFACE

INTEGRANTES DEL GRUPO

- Albornoz Quiliano Jose Antonio
- CAJAMALQUI DÁVILA, JOSEMARÍA PIERO
- CARBAJAL PACCORI, BRAYAN ERICK
- DE LA TORRE SEGURA, JHONNY ANDRE
- Veliz De La Cruz ,Carlos Adrian
-

“CONSTRUCCIÓN DE SOFTWARE”

HUANCAYO-PERÚ

2024

EXTRACT INTERFACE

Imagina que tienes una clase que gestiona datos, digamos, una clase `BaseDatos`. Esta clase podría tener métodos como `guardarDatos()`, `recuperarDatos()`, y `eliminarDatos()`. Estos métodos permiten interactuar con una base de datos, pero es posible que necesites realizar operaciones similares en otras partes de tu aplicación, en otras clases.

Aquí es donde entran en juego las interfaces. Una interfaz define un contrato: especifica qué métodos deben implementarse, pero no proporciona la implementación en sí. Por ejemplo, podrías definir una interfaz `IDataManager` con los métodos `guardarDatos()`, `recuperarDatos()`, y `eliminarDatos()`. Cualquier clase que implemente esta interfaz estaría obligada a proporcionar su propia versión de estos métodos.

Beneficios de usar interfaces:

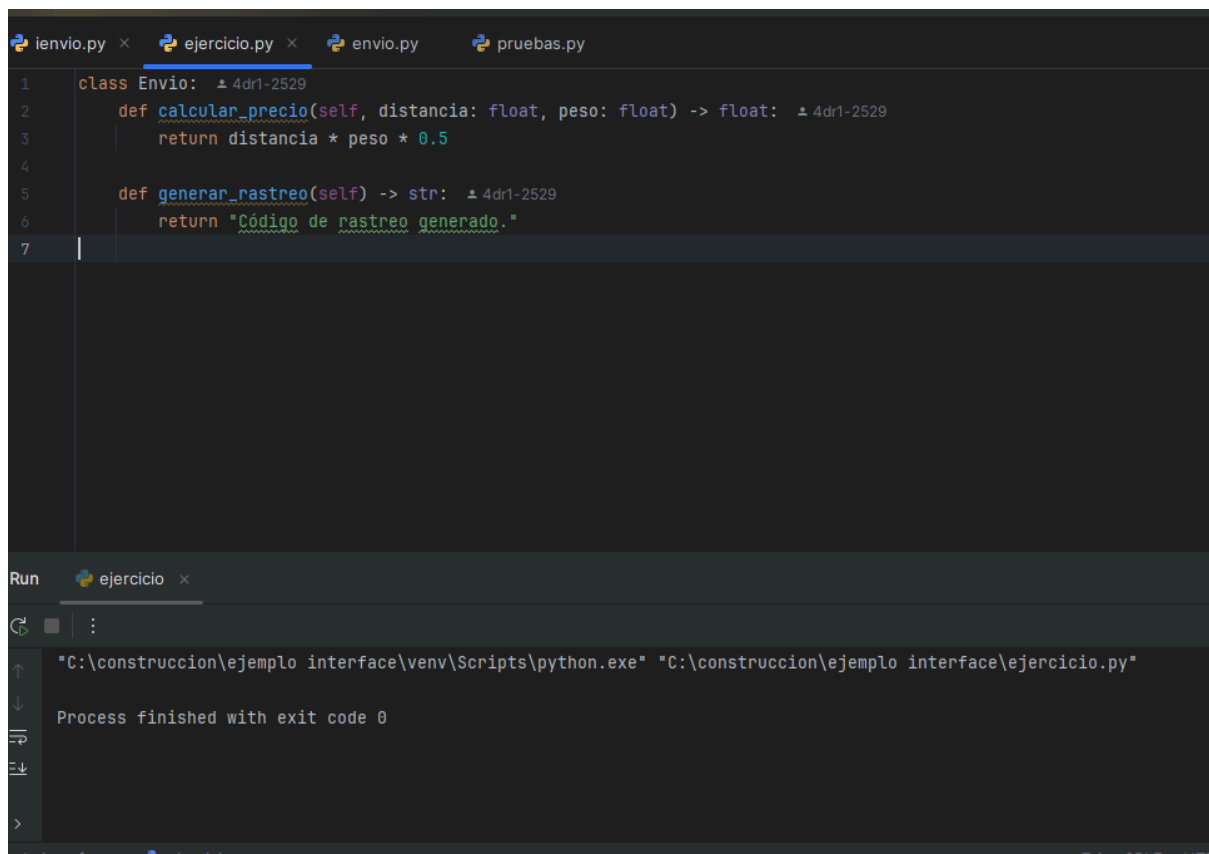
1. Reutilización de código: Las clases pueden implementar la misma interfaz y, por lo tanto, compartir la lógica general de los métodos, lo que evita la duplicación de código.
2. Puedes tratar diferentes clases de manera uniforme. Si varias clases implementan `IDataManager`, puedes crear funciones que acepten esta interfaz como parámetro, permitiendo trabajar con diferentes implementaciones sin saber exactamente qué clase se está utilizando.
3. Flexibilidad: Si decides cambiar la implementación de cómo se almacenan los datos (por ejemplo, cambiar de una base de datos SQL a una NoSQL), puedes hacerlo simplemente creando una nueva clase que implemente la misma interfaz, sin necesidad de modificar el código que utiliza la interfaz.
4. Facilidad de pruebas: Las interfaces permiten crear implementaciones simuladas o "mock" durante las pruebas, lo que facilita comprobar el comportamiento de tu código sin depender de componentes externos.

En resumen, usar interfaces basadas en clases que contienen métodos comunes es una buena práctica que fomenta la reutilización, la flexibilidad y un código más limpio y mantenible.

Ejemplo de Interfaz de Envío

Descripción del Proyecto: Este proyecto es un sistema simple de gestión de envíos de paquetes, donde se pueden calcular precios de envío y generar códigos de rastreo únicos para cada pedido

Descripción del Archivo `ejercicio.py`: Este archivo es el punto de entrada del programa y contiene la lógica para crear instancias de la clase Envío con diferentes parámetros (distancia y peso), y luego imprimir la información de cada envío, incluyendo el precio y el código de rastreo.



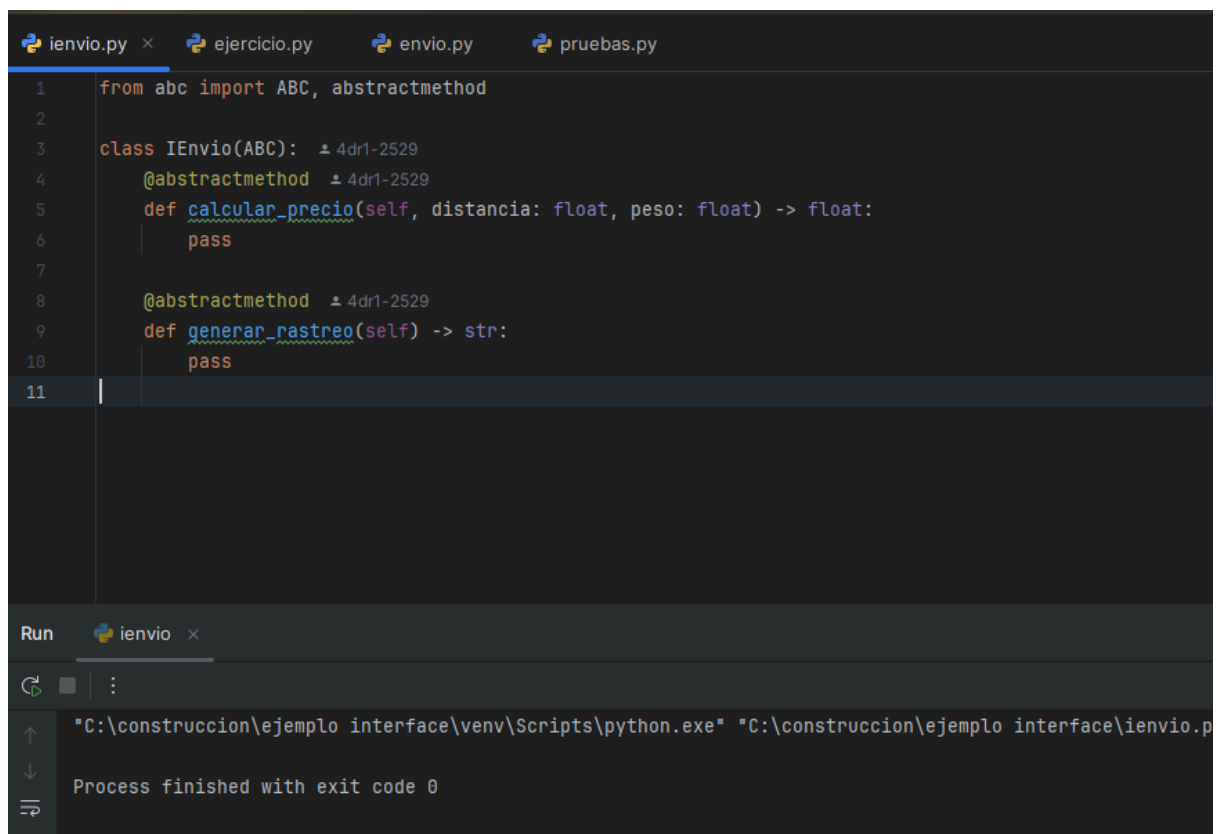
```
1 class Envío:
2     def calcular_precio(self, distancia: float, peso: float) -> float:
3         return distancia * peso * 0.5
4
5     def generar_rastreo(self) -> str:
6         return "Código de rastreo generado."
7
Run ejercicio
"C:\construccion\ejemplo interface\venv\Scripts\python.exe" "C:\construccion\ejemplo interface\ejercicio.py"
Process finished with exit code 0
```

Descripción de la Interfaz: La interfaz `IEnvío` define un conjunto de métodos que cualquier clase de envío debe implementar. Esto es parte de un enfoque orientado a interfaces, lo que ayuda a asegurar que diferentes tipos de envíos sigan una estructura común. Esto es útil para mantener la consistencia y la claridad en el código.

Componentes Clave:

1. Interfaz IEnvio:

- Se define con métodos que las clases que implementan la interfaz deben proporcionar:
 - `calcular_precio(distancia, peso)`: Método que calculará el precio de envío basado en la distancia y el peso del paquete.
 - `generar_codigo_rastreo()`: Método que generará un código único para rastrear el envío.
 - `mostrar_informacion()`: Método que mostrará la información del envío, como el precio y el código de rastreo.



```
1 from abc import ABC, abstractmethod
2
3 class IEnvio(ABC):
4     @abstractmethod
5     def calcular_precio(self, distancia: float, peso: float) -> float:
6         pass
7
8     @abstractmethod
9     def generar_codigo_rastreo(self) -> str:
10         pass
11
```

Run ienvio

```
"C:\construccion\ejemplo interface\venv\Scripts\python.exe" "C:\construccion\ejemplo interface\ienvio.p
Process finished with exit code 0
```

Descripción de la Clase: La clase `Envio` representa un sistema para gestionar el envío de paquetes. Implementa la interfaz `IEnvio`, que establece los métodos que deben estar presentes en cualquier clase de envío. Esta clase permite calcular el costo del envío, generar un código de rastreo y mostrar la información del envío.

Componentes Clave:

1. Clase Envio:

- Atributos:
 - `distancia`: Distancia a la que se enviará el paquete (en kilómetros).

- peso: Peso del paquete (en kilogramos).
- precio: Precio calculado para el envío basado en la distancia y peso.
- codigo_rastreo: Un código único generado para rastrear el envío.
- Métodos:
 - calcular_precio(): Calcula el costo del envío multiplicando la distancia y el peso por un factor (en este caso, 0.5).
 - generar_codigo_rastreo(): Genera un código alfanumérico aleatorio que se utiliza para rastrear el paquete.
 - mostrar_informacion(): Devuelve una cadena que incluye el precio y el código de rastreo del pedido.

The screenshot shows a Python IDE with several tabs: 'ienvio.py', 'ejercicio.py', 'envio.py', and 'pruebas.py'. The 'envio.py' tab is active, displaying the following code:

```

1 import random
2 C:\construccion\ejemplo interface\ienvio.py
3
4 class Envio: 3 usages 4dr1-2529
5     contador = 0
6
7     def __init__(self, distancia: float, peso: float): 4dr1-2529
8         Envio.contador += 1
9         self.distancia = distancia
10        self.peso = peso
11        self.precio = self.calcular_precio()
12        self.codigo_rastreo = self.generar_codigo_rastreo()
13
14    def calcular_precio(self) -> float: 1 usage 4dr1-2529
15        return self.distancia * self.peso * 0.5
16
17    def generar_codigo_rastreo(self) -> str: 1 usage 4dr1-2529
18        codigo = ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(10))

```

Below the code editor, the 'Run' button is visible. The output window shows the command executed: `"C:\construccion\ejemplo interface\venv\Scripts\python.exe" "C:\construccion\ejemplo interface\envio.py"`, followed by the message: `Process finished with exit code 0`.

Descripción del Módulo de Pruebas: El módulo de pruebas verificará que los métodos de la clase `Envio` se comporten como se espera. Utilizaremos el framework de pruebas de Python llamado `unittest`. Este módulo creará diferentes casos de prueba que comprobarán el cálculo de precios, la generación de códigos de rastreo y la presentación de información del envío

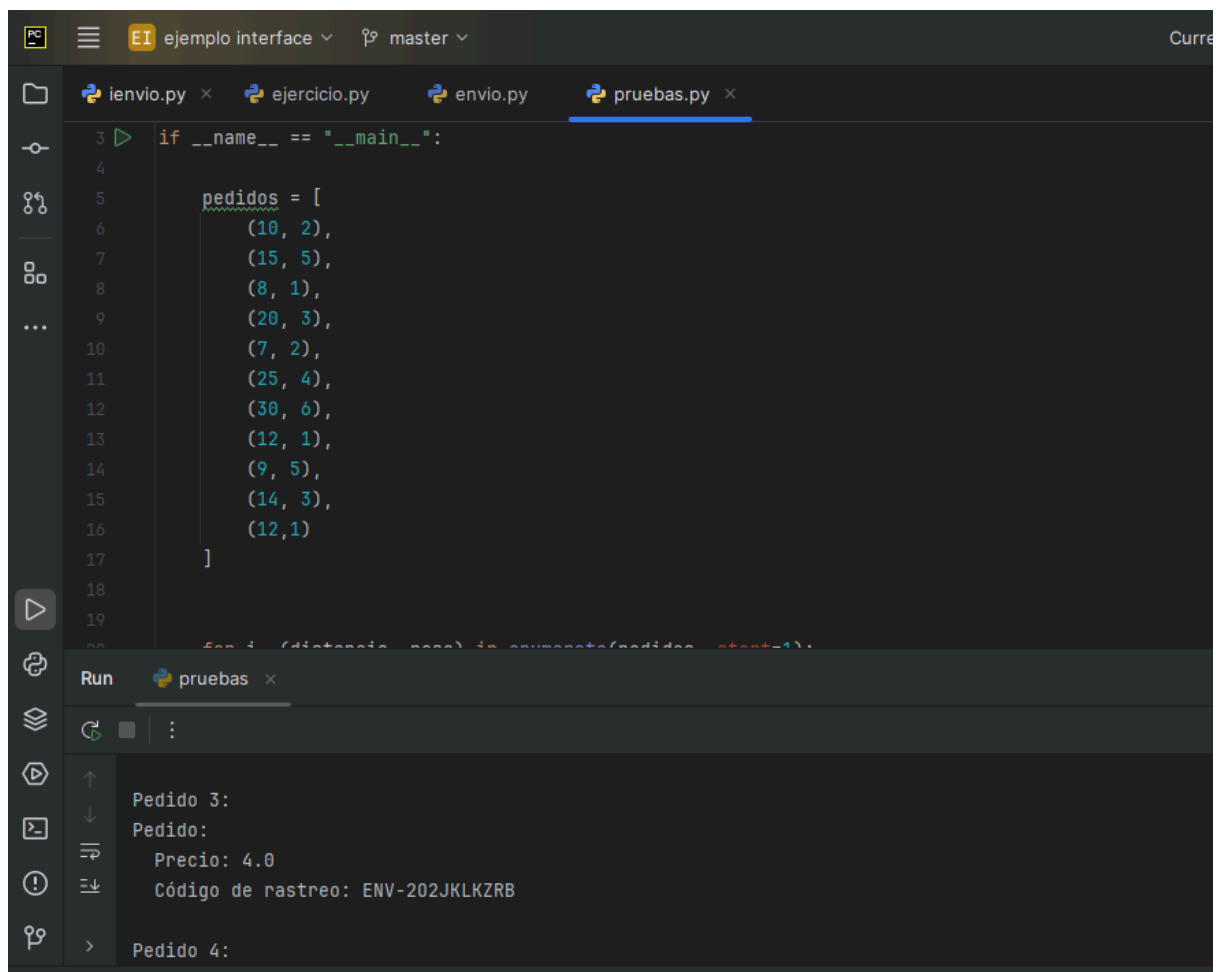
Contenido del Módulo de Pruebas

1. Importaciones Necesarias:

- Se importan los módulos `unittest` y la clase `Envio`.

2. Clase de Prueba:

- Se define una clase de prueba que hereda de `unittest.TestCase`, donde se crean diferentes métodos para probar las funcionalidades de la clase `Envio`



```
if __name__ == "__main__":
    pedidos = [
        (10, 2),
        (15, 5),
        (8, 1),
        (20, 3),
        (7, 2),
        (25, 4),
        (30, 6),
        (12, 1),
        (9, 5),
        (14, 3),
        (12, 1)
    ]

    for i, (distancia, peso) in enumerate(pedidos, start=1):
```

Run pruebas x

Pedido 3:
Pedido:
Precio: 4.0
Código de rastreo: ENV-202JKLKZRB

Pedido 4:

Ejemplo de Uso:

Al crear una nueva instancia de `Envio`, se ingresan la distancia y el peso. Luego, se puede obtener el precio y el código de rastreo

```
"C:\construccion\ejemplo interface\venv\Scripts\python.exe" "C:\construccion\ejemplo interface\pruebas.py"
Pedido 1:
Pedido:
  Precio: 10.0
  Código de rastreo: ENV-DV5SUAP4P2

Pedido 2:
Pedido:
  Precio: 37.5
  Código de rastreo: ENV-LA0Y68K811

Pedido 3:
Pedido:
  Precio: 4.0
  Código de rastreo: ENV-202JKLKZRB

Pedido 4:
```

Este sistema permite a los usuarios calcular rápidamente el costo de envío y obtener un código único para el rastreo de sus paquetes, facilitando la gestión y seguimiento de los envíos.