Ejercicio 5 Individual

ADAM NAVARRO MEGIAS

23 de marzo de 2025

1. Introducción

Este ejercicio se implementa de la mejor forma con el patrón Adaptador, ya que este patrón consiste en crear una clase adaptadora que permitirá el acceso del cliente a los datos de otra clase ya existente en el formato que desee, en este caso es que el cliente (sistema geográfico de España) quiere tener los datos que recibe transformados de millas a kilómetros y este patrón nos permite no tener que modificar la clase del servicio de los ángeles.

2. Iterfaz y clases

Para este ejercicio individual tenemos primero la clase de la que partimos que implementa el servicio de navegación de Los Ángeles, en este caso es una representación sencilla para hacernos a la idea. Simplemente recibe una distancia en el constructor, que se puede cambiar después, y tiene una función que devuelve la distancia:

```python
# Pequeña clase que es el servicio de navegación de los Ángeles, es el adaptee, que queremos adaptar
class LosAngelesNav:  1 usage  Adam Navarro Megías
    def __init__(self, miles):  Adam Navarro Megías
        self.distanceMiles = miles

    def set_distance(self, miles):  1 usage (1 dynamic)  Adam Navarro Megías
        self.distanceMiles = miles

    def obtain_distance_miles(self):  1 usage (1 dynamic)  Adam Navarro Megías
        return self.distanceMiles
```

Luego tenemos la interfaz que espera el cliente para usar y la clase adaptadora que implementa esa interfaz y que recibe una instancia de la clase de servicio de los ángeles, y es la que transforma los datos para el cliente:

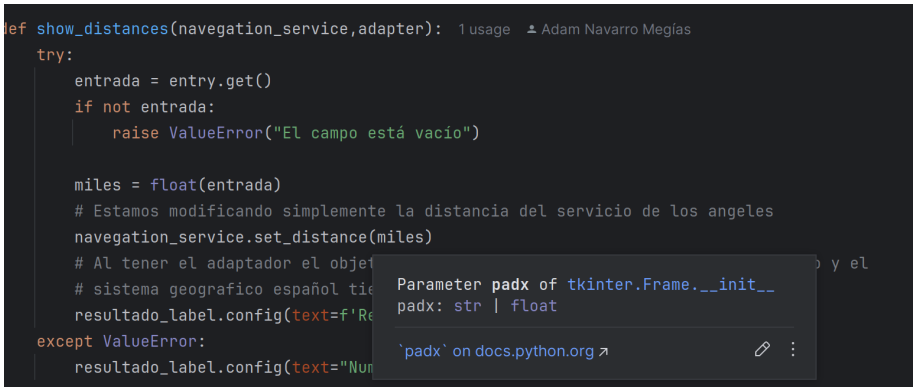
```python
from abc import ABC, abstractmethod


# Interfaz que el cliente va a esperar, siendo el cliente el sistema geográfico de España
class KilometerService(ABC):  1 usage
    @abstractmethod  1 usage (1 dynamic)
    def obtain_distance(self):
        pass

# Adaptador que utiliza la interfaz
class LosAngelesAdapter(KilometerService):  1 usage
    def __init__(self, los_angeles_nav):
        self.los_angeles_nav = los_angeles_nav

    def obtain_distance(self):  1 usage (1 dynamic)
        return (self.los_angeles_nav.obtain_distance_miles())*1.60934
```

3. Main

Por último tenemos el main, que en este caso lo he considerado como si fuera el cliente y que tiene objeto de la clase del servicio y el adaptador. Además he utilizado tkinter para hacer una pequeña y muy simple ventana que permitirá introducir un valor de millas (que se introduce en el objeto de servio de los ángeles) y se modifica automáticamente el adaptador (ya que tiene el objeto como atributo) mostrándose también en la ventana:

```python
if __name__ == '__main__':

    # Definimos los objetos De servicio y adaptador que tendría el Cliente (Sistema geográfico)
    navegation_service_la = LosAngelesNav(10)
    navegation_service_adapter = LosAngelesAdapter(navegation_service_la)


    # Pequeña interfaz gráfica con tkinter para ver el resultado directo
    # Crear la ventana principal
    root = tk.Tk()
    root.title("Conversión de Distancias")

    # Crear y ubicar widgets
    frame = tk.Frame(root, padx=10, pady=10)
    frame.pack()

    entry_label = tk.Label(frame, text="Introduce la distancia en millas:")
    entry_label.grid(row=0, column=0, sticky="w")

    entry = tk.Entry(frame)
    entry.grid(row=0, column=1)

    convertir_button = tk.Button(frame, text="Convertir", command=lambda: show_distances(navegation_service_la,
                                                                                          navegation_service_adapter))
    convertir_button.grid(row=1, columnspan=2, pady=5)

    resultado_label = tk.Label(frame, text="Resultado:")
    resultado_label.grid(row=2, columnspan=2)

    # Ejecutar la ventana
    root.mainloop()
```

Para la interfaz se usa el root que se considera la ventana principal, y luego se crea un frame, que es un widget o subventana, donde se colocan 3 cosas usando grid. Se coloca un label para indicar donde se introduce el dato y luego un Entry para introducirlo, luego se coloca un button que llamará a la función que explicaré más abajo, y luego un texto con Resultado:, que se modificará para colocar el resultado cada vez que se pinche en convertir. Se usa grid con row y column para colocar los dos primeros elementos adyacentes y los dos siguientes elementos debajo el uno del otro y ocupando las dos columnas. La función que se llama es la siguiente:



En esta función simplemente se usa un manejo de excepciones para comprobar si el número está bien, y luego se obtiene el valor, se modifica el objeto de navegación y se usa el adaptador para mostrar el valor en kilómetros.