Tarea Patron Factory

Fábrica de Animales

El patrón de diseño Factory es un patrón creacional que proporciona una interfaz para crear objetos en una clase base, pero permite a las subclases alterar el tipo de objetos que se crearán. Este patrón se usa cuando el proceso de creación requiere un control centralizado, permitiendo así una fácil extensión y modificación del código sin afectar al cliente.

En este ejemplo, implementaremos una fábrica de animales que puede crear diferentes tipos de animales como perros, gatos y pájaros.

1. Definir la interfaz Animal

Primero, creamos una clase base abstracta Animal que será implementada por diferentes clases de animales. Esta clase define el método abstracto make_sound(), que deberá ser implementado por todas las clases que representen un animal.

```python
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def make_sound(self):
        pass
```

2. Implementar las clases de animales concretos

A continuación, implementamos varias clases que representan diferentes tipos de animales. Cada una de estas clases implementa la clase base abstracta Animal y define el método make_sound() para devolver el sonido característico de cada animal.

```python
class Dog(Animal):
    def make_sound(self):
        return "Woof"

class Cat(Animal):
    def make_sound(self):
        return "Meow"

class Bird(Animal):
    def make_sound(self):
        return "Chirp"
```

3. Crear la clase AnimalFactory

La clase AnimalFactory es responsable de la creación de instancias de diferentes tipos de animales basándose en el tipo de animal proporcionado. La fábrica utiliza un método estático get_animal() para devolver la instancia correcta.

```python
class AnimalFactory:
    @staticmethod
    def get_animal(animal_type):
        if animal_type == "Dog":
            return Dog()
        elif animal_type == "Cat":
            return Cat()
        elif animal_type == "Bird":
            return Bird()
        else:
            raise ValueError(f"Unknown animal type: {animal_type}")
```

4. Implementar la clase cliente

La clase cliente utiliza la fábrica para obtener instancias de Animal y mostrar sus sonidos.

```python
if __name__ == "__main__":
    try:
        dog = AnimalFactory.get_animal("Dog")
        print(dog.make_sound())

        cat = AnimalFactory.get_animal("Cat")
        print(cat.make_sound())

        bird = AnimalFactory.get_animal("Bird")
        print(bird.make_sound())
    except ValueError as e:
        print(e)
```

Este ejemplo muestra cómo el patrón Factory permite crear instancias de diferentes clases que implementan una interfaz común sin exponer la lógica de creación al cliente. La fábrica centraliza el proceso de creación, lo que facilita la gestión y extensión del código.