Practica 6

```python
# Practica 6. Factory

# Definición de la clase base o "interfaz"
class Animal:
    # El método 'speak' es un contrato: todas las subclases deben implementarlo.
    def speak(self):
        pass

# --- Clases Concretas (Herencia y Polimorfismo) ---

# Clase que hereda de Animal
class Dog(Animal):
    # Sobrescribe el método 'speak' para proporcionar un comportamiento específico.
    def speak(self):
        return "Woof!"

# Clase que hereda de Animal
class Cat(Animal):
    # Sobrescribe el método 'speak' para proporcionar un comportamiento específico.
    def speak(self):
        return "Meow!"

# --- Clase Factoría (Factory) ---

# Clase responsable de crear instancias de animales sin exponer la lógica de instanciación
class AnimalFactory:
    # Método de fábrica que toma una cadena y decide qué clase instanciar
    def create_animal(self, animal_type):
        # Lógica de creación: si el tipo es "Dog", devuelve un nuevo objeto Dog
        if animal_type == "Dog":
            return Dog()
        # Lógica de creación: si el tipo es "Cat", devuelve un nuevo objeto Cat
```

```
        elif animal_type == "Cat":
            return Cat()
        # Si el tipo no es reconocido, lanza un error para detener la ejecución
        else:
            raise ValueError("Invalid animal type")

# --- Uso del Código ---

# 1. Instancia la clase Factoría. Esta es la 'máquina' que crea objetos.
factory = AnimalFactory()

# 2. Llama al método de fábrica para crear un objeto.
# La fábrica devuelve un objeto Dog, pero la variable 'animal' solo sabe que es un Animal.
animal = factory.create_animal("Dog")

# 3. Llama al método 'speak' del objeto devuelto.
# Gracias al polimorfismo, el método específico de Dog ("Woof!") se ejecuta.
# Esta línea devuelve la cadena "Woof!"
animal.speak()



# Practica 6. Observer

# La clase Sujeto mantiene la lista de observadores y notifica los cambios de estado.
class Subject:
    def __init__(self):
        # Lista privada para almacenar los objetos Observador suscritos.
        self._observers = []
        # El Sujeto podría tener atributos de estado que cambian (aunque 'state' se añade di

    # Método para que un Observador se suscriba al Sujeto.
    def attach(self, observer):
        self._observers.append(observer)

    # Método para que un Observador se desuscriba del Sujeto.
    def detach(self, observer):
        self._observers.remove(observer)

    # Método clave: Notifica a todos los observadores sobre un cambio de estado.
    def notify(self):
        # Itera sobre la lista de observadores...
```

```
        for observer in self._observers:
            # ...y llama al método 'update' de cada uno, pasándose a sí mismo (el Sujeto) co
            observer.update(self)

# --- Patrón Observer: La Interfaz (Observer) ---

# Clase base o interfaz para todos los Observadores.
class Observer:
    # Define el método 'update' que será llamado por el Sujeto.
    # El 'pass' indica que es una plantilla para las clases concretas.
    def update(self, subject):
        pass

# --- Patrón Observer: El Observador Concreto (ConcreteObserver) ---

# Implementación específica del Observador.
class ConcreteObserver(Observer):
    # La lógica de reacción al cambio de estado del Sujeto.
    def update(self, subject):
        # Accede y utiliza la información del Sujeto (su estado).
        # Esto es lo que sucede cuando el estado del Sujeto cambia.
        print("State changed:", subject.state)

# --- Uso del Patrón ---

# 1. Creación del Sujeto que será observado.
subject = Subject()

# 2. Creación de una instancia del Observador Concreto.
observer = ConcreteObserver()

# 3. Suscripción: el Observador se 'adjunta' al Sujeto para recibir notificaciones.
subject.attach(observer)

# 4. Cambio de Estado: Se asigna un atributo de estado al Sujeto.
# NOTA: Este cambio en sí mismo aún no notifica a nadie.
subject.state = 42

# 5. Notificación: El Sujeto anuncia explícitamente el cambio de estado.
subject.notify()
# Resultado: El método 'update' del 'observer' se ejecuta y consulta el nuevo estado (42).
```

```
State changed: 42
```