```python
# Patrón Factory - Fábrica de vehículos
# La idea es que una "fábrica" decida qué objeto crear
# en lugar de que el programador lo instancie directamente.

# Clase base: define el comportamiento común de los vehículos
class Vehiculo:
    def mover(self):
        pass  # Método que cada vehículo deberá implementar

# Clases concretas que heredan de Vehiculo
class Auto(Vehiculo):
    def mover(self):
        return "El auto está manejando por la carretera"

class Camion(Vehiculo):
    def mover(self):
        return "El camión está transportando mercancía"

class Moto(Vehiculo):
    def mover(self):
        return "La moto está corriendo por la ciudad"

# Fábrica: centraliza la creación de objetos
class VehiculoFactory:
    @staticmethod
    def crear_vehiculo(tipo):
        # Según el tipo recibido, crea un objeto distinto
        if tipo == "auto":
            return Auto()
        elif tipo == "camion":
            return Camion()
        elif tipo == "moto":
            return Moto()
        else:
            raise ValueError("Tipo de vehículo no reconocido")

# ---- Simulación ----

# Creamos un auto usando la fábrica
vehiculo1 = VehiculoFactory.crear_vehiculo("auto")
# Creamos un camión usando la fábrica
vehiculo2 = VehiculoFactory.crear_vehiculo("camion")

# Ambos son vehículos, pero se comportan distinto según su clase
print(vehiculo1.mover())  # "El auto está manejando por la carretera"
print(vehiculo2.mover())  # "El camión está transportando mercancía"


#_____________________________________________________________________________


# Patrón Observer - Canal de YouTube y suscriptores
# La idea es que cuando un "sujeto" cambia (el canal sube un video),
# notifica automáticamente a todos los "observadores" (usuarios).

# Clase sujeto observado (Publisher)
class CanalYoutube:
    def __init__(self, nombre):
        self.nombre = nombre
        self.suscriptores = []  # Lista de observadores (usuarios)

    def suscribir(self, usuario):
        # Agrega un usuario a la lista de suscriptores
        self.suscriptores.append(usuario)

    def subir_video(self, titulo):
        # Cuando se sube un video, se notifica a todos los suscriptores
        print(f"\n{self.nombre} subió un nuevo video: {titulo}")
        self.notificar(titulo)

    def notificar(self, titulo):
        # Recorremos la lista y avisamos a cada observador
        for suscriptor in self.suscriptores:
            suscriptor.actualizar(self.nombre, titulo)

# Clase observador
class Usuario:
    def __init__(self, nombre):
        self.nombre = nombre

    def actualizar(self, canal, video):
        # Cada vez que el canal sube un video, este método se ejecuta
        print(f"{self.nombre} recibió notificación: Nuevo video '{video}' en {canal}")

# ---- Simulación ----

# Creamos el canal
canal = CanalYoutube("TechCode")

# Creamos usuarios (observadores)
u1 = Usuario("Ana")
u2 = Usuario("Carlos")
u3 = Usuario("Sofía")

# Los usuarios se suscriben al canal
canal.suscribir(u1)
canal.suscribir(u2)
canal.suscribir(u3)

# El canal sube un video y automáticamente notifica a todos los usuarios
canal.subir_video("Patrón Observer explicado fácil")
```