

Factory Method: Creación de Objetos Simplificada

Factory Method, simplificar la creación de objetos.

Este patrón aporta flexibilidad y claridad al diseño de software.

Paul Yesid Basto Parra

Karol Andrea Hernández Saldarriaga

El Problema: Creación Compleja de Objetos

Dificultad con `new`

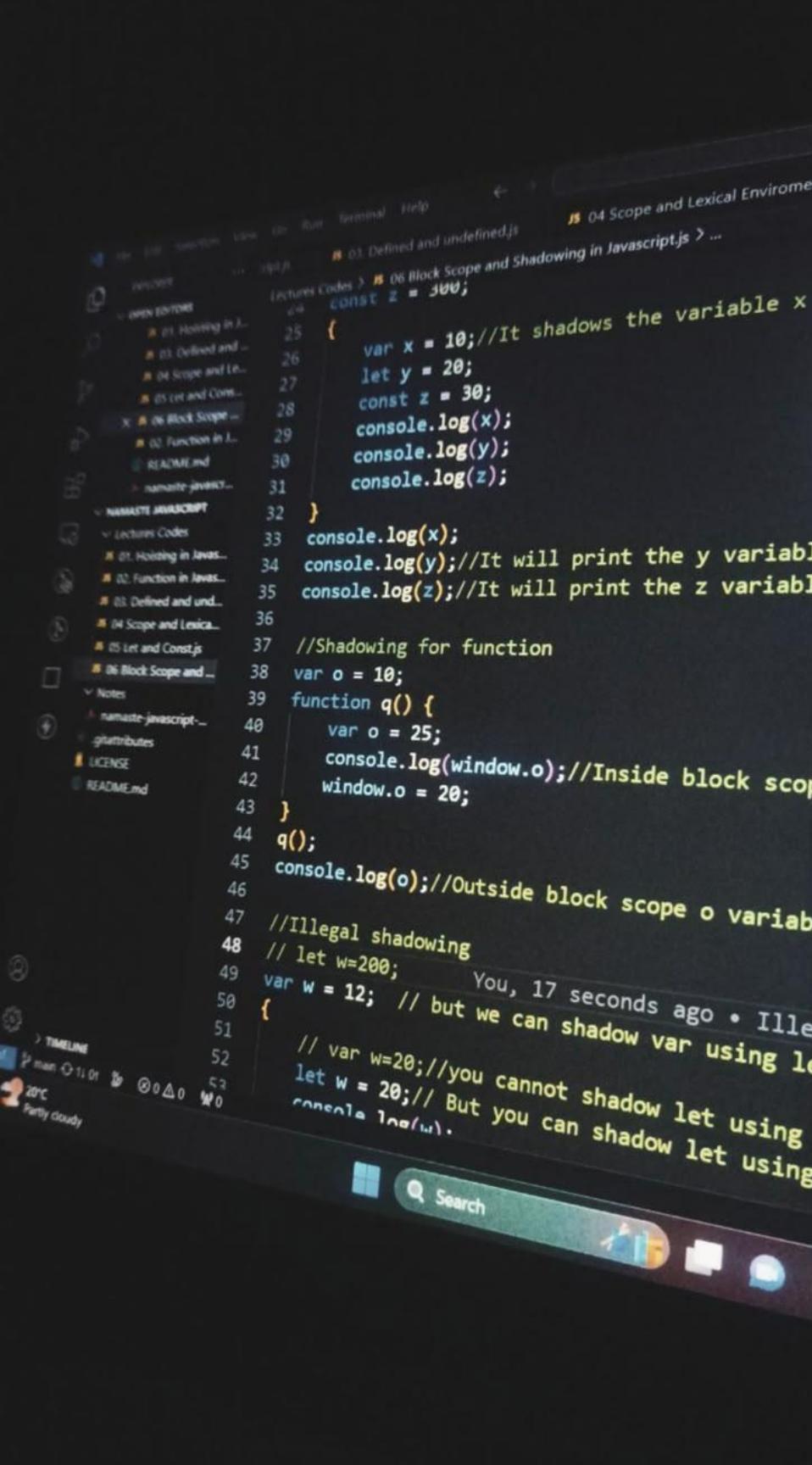
Crear objetos directamente complica la gestión y el mantenimiento del código.

Código Acoplado

El código depende fuertemente de clases concretas, dificultando cambios y extensiones.

Flexibilidad limitada

Agregar nuevas clases implica modificar el código existente, generando rigidez.



La Solución: Factory Method



Interfaz para crear objetos

Se define una interfaz común para la creación de objetos.



Delegación a subclases

Las subclases deciden qué objeto crear, aumentando la flexibilidad.



Encapsulación de la creación

La lógica de creación queda aislada, facilitando mantenimiento y extensión.



Componentes Clave

Producto

Interfaz que define los objetos que se crearán.

ConcreteProduct

Implementaciones concretas de productos.

Creador (Creator)

Declara el método factory para crear objetos del tipo Producto.

ConcreteCreator

Implementa el método factory para instanciar productos específicos.

Beneficios del Factory Method

Inversión de Dependencias

El código depende de abstracciones, no de implementaciones concretas.

Flexibilidad

Fácil añadir nuevos tipos de productos sin cambiar código cliente.

Mantenimiento Limpio

Separación clara de creación y uso del objeto mejora mantenimiento.



Casos de Uso



Diferentes Documentos

Crear distintos formatos de documentos de forma flexible.



Multiplataforma

Implementar distintas plataformas reutilizando interfaces comunes.



Objetos Complejos

Construcción configurable y modular de objetos complejos.



Frameworks UI

Diversos tipos de botones para múltiples sistemas operativos.

Ejemplo de Código

1

Interfaz Animal

Define comportamiento común para `Dog` y `Cat`.

2

AnimalFactory

Método `createAnimal(String type)` crea instancias según tipo.

3

Uso

`Animal animal = factory.createAnimal("dog");` crea un perro sin acoplar código.

```
Q:\000 Python\Администратор|
```

```
main.py          main.py > MusicAlbum > __init__.py
                import random

class MusicAlbum:
    def __init__(self, title, artist, release_year, genre, tracklist):
        self.title = title
        self.artist = artist
        self.release_year = release_year
        self.genre = genre
        self.tracklist = tracklist

    def play_track(self, track_number):
        return f"Воспроизводится трек {track_number}: {self.tracklist[track_number]}"

    def play_random_track(self):
        track_number = random.randint(1, len(self.tracklist))
        self.play_track(track_number)

album4 = MusicAlbum("Deutschland", "Rammstein", 2019, "Neue Deutsche Härte", ["Deutschland", "Radio", "Zeig dich", "Ausländer", "Seine Puppe", "Was ich liebe", "Diamant", "Weit weg", "Tatort Hallmann"])
```



Conclusión

Factory Method es un patrón poderoso y flexible. Facilita la creación de objetos complejos y mejora la mantenibilidad. Es clave para diseñar sistemas escalables y fáciles de extender.

Adoptar este patrón optimiza arquitectura y reduce dependencia de implementaciones concretas.