PATRONES DE DISEÑO CREACIONALES

MARTÍNEZ COSIO JOSÉ ALFREDO

Aarón Hernández García

Factory Method

09/11/2024

El **Factory Method** es un patrón de diseño creacional en programación que se usa para definir una interfaz o método abstracto para crear un objeto, pero dejando que las subclases decidan qué clase concreta instanciar.

¿Por Qué Usar el Factory Method?

- **Desacoplamiento**: Permite crear objetos sin especificar la clase exacta de estos, lo cual hace que el código sea más flexible y fácil de mantener.
- **Extensibilidad**: Si necesitas agregar nuevos tipos de objetos, puedes hacerlo creando nuevas subclases en lugar de modificar el código existente.
- **Cambio Dinámico**: Facilita la creación de diferentes tipos de objetos en tiempo de ejecución, dependiendo de la situación o los parámetros.

Código en Python ->

salones.py

```python
from abc import ABC, abstractmethod

# Clase Producto Abstracta
class Salon(ABC):
    @abstractmethod
    def configuracion(self):
        pass

# Clases Producto Concretas
class SalonBodas(Salon):
    def configuracion(self):
        return "Salón de Bodas: Incluye decoraciones especiales."

class SalonConferencias(Salon):
    def configuracion(self):
        return "Salón de Conferencias: Equipado con TV gigantes."

class SalonBaile(Salon):
    def configuracion(self):
        return "Salón de Baile: Piso encerado ideal para bailar."

# Clase Factory Abstracta
class SalonFactory(ABC):
    @abstractmethod
    def crear_salon(self):
        pass

# Clases Factory Concretas
class SalonBodasFactory(SalonFactory):
    def crear_salon(self):
        return SalonBodas()

class SalonConferenciasFactory(SalonFactory):
    def crear_salon(self):
        return SalonConferencias()

class SalonBaileFactory(SalonFactory):
    def crear_salon(self):
        return SalonBaile()

```

Código en Python ->

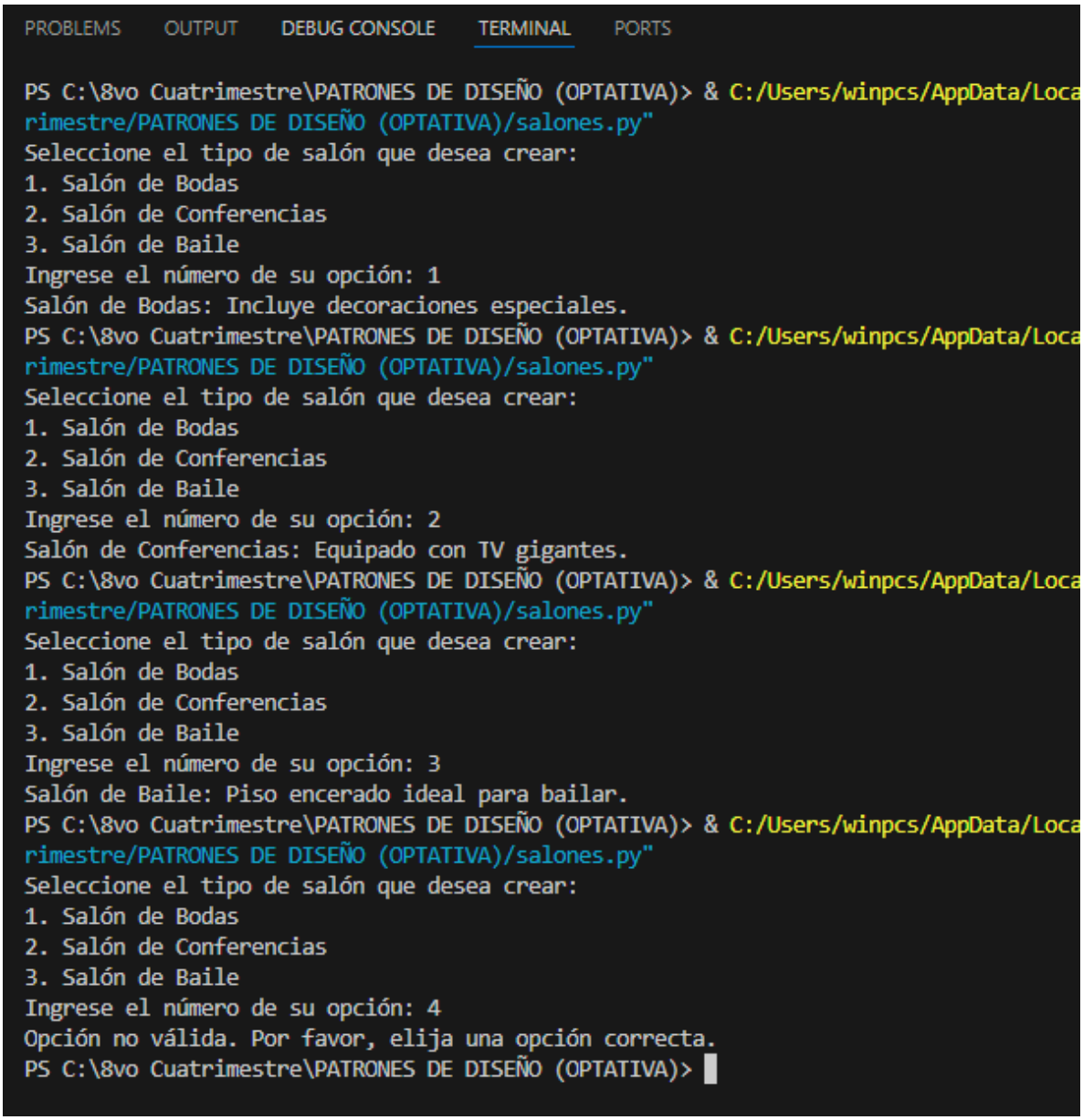
```python
# Función para mostrar el menú y crear el salón solicitado
def gestionar_salon():
    print("Seleccione el tipo de salón que desea crear:")
    print("1. Salón de Bodas")
    print("2. Salón de Conferencias")
    print("3. Salón de Baile")
    opcion = input("Ingrese el número de su opción: ")

    # Diccionario de fábricas
    fabricas = {
        "1": SalonBodasFactory(),
        "2": SalonConferenciasFactory(),
        "3": SalonBaileFactory()
    }

    # Crear salón según la opción del usuario
    fabrica = fabricas.get(opcion)
    if fabrica:
        salon = fabrica.crear_salon()
        print(salon.configuracion())
    else:
        print("Opción no válida. Por favor, elija una opción correcta.")

# Ejecutar la función
gestionar_salon()
```

Respuesta ->


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\8vo Cuatrimestre\PATRONES DE DISEÑO (OPTATIVA)> & C:/Users/winpcs/AppData/Loca
rimestre/PATRONES DE DISEÑO (OPTATIVA)/salones.py"
Seleccione el tipo de salón que desea crear:
1. Salón de Bodas
2. Salón de Conferencias
3. Salón de Baile
Ingrese el número de su opción: 1
Salón de Bodas: Incluye decoraciones especiales.
PS C:\8vo Cuatrimestre\PATRONES DE DISEÑO (OPTATIVA)> & C:/Users/winpcs/AppData/Loca
rimestre/PATRONES DE DISEÑO (OPTATIVA)/salones.py"
Seleccione el tipo de salón que desea crear:
1. Salón de Bodas
2. Salón de Conferencias
3. Salón de Baile
Ingrese el número de su opción: 2
Salón de Conferencias: Equipado con TV gigantes.
PS C:\8vo Cuatrimestre\PATRONES DE DISEÑO (OPTATIVA)> & C:/Users/winpcs/AppData/Loca
rimestre/PATRONES DE DISEÑO (OPTATIVA)/salones.py"
Seleccione el tipo de salón que desea crear:
1. Salón de Bodas
2. Salón de Conferencias
3. Salón de Baile
Ingrese el número de su opción: 3
Salón de Baile: Piso encerado ideal para bailar.
PS C:\8vo Cuatrimestre\PATRONES DE DISEÑO (OPTATIVA)> & C:/Users/winpcs/AppData/Loca
rimestre/PATRONES DE DISEÑO (OPTATIVA)/salones.py"
Seleccione el tipo de salón que desea crear:
1. Salón de Bodas
2. Salón de Conferencias
3. Salón de Baile
Ingrese el número de su opción: 4
Opción no válida. Por favor, elija una opción correcta.
PS C:\8vo Cuatrimestre\PATRONES DE DISEÑO (OPTATIVA)>

Referencias ->

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Freeman, E., & Robson, E. (2004). Head First Design Patterns. O'Reilly Media.

Refactoring Guru. (n.d.). Factory Method. Recuperado de https://refactoring.guru/design-patterns/factory-method.