SINGLETON PATTERNINI O‘RGANISH VA LOGGER SINFI YOZISH

Mirsaid Yusupov
Farg‘ona davlat universiteti Amaliy matematika va informatika kafedrasi o’qituvchisi
E-mail: mirsaidbeky@gmail.com
Suyarov Ulug’bek G’iyosiddin o’g’li
Farg‘ona davlat universiteti Amaliy matematika yo‘nalishi 23.07-guruh talabasi
E-mail:suyarov1857@gmail.com

Anotatsiya: Ushbu maqolada dasturlashda keng qo‘llaniladigan Singleton patternga texnik va nazariy yondashuv beriladi. Patternning asosiy maqsadi — dastur ish jarayonida bir xil sinfdan faqat bitta obyekt yaratilishini ta’minlashdir. Python tilida Singleton’ni yaratishning amaliy mexanizmlari, xususan obyekt instansiyasini nazorat qilish tamoyillari bosqichma-bosqich tahlil qilinadi. Shuningdek, loglarni markazlashgan holda boshqarish uchun zarur bo‘lgan Logger sinfining Singleton asosida qurilishi ko‘rsatiladi. Ushbu yondashuv log yozuvlarining izchil saqlanishi, xotira resurslarini optimallashtirish va dastur ichida yagona boshqaruv nuqtasini yaratishni ta’minlaydi.

Kalit so‘zlar: Singleton pattern, obyekt, instansiya, Python OOP, Logger, dizayn patternlari, markazlashgan loglash.

Abstract : This article presents a technical and theoretical approach to the widely used Singleton pattern**.** The main goal of the pattern is to ensure that only one object of a class is created during the program’s execution. In Python, practical mechanisms for implementing Singleton, including controlling object instantiation, are analyzed step by step. Additionally, the construction of the Logger class based on Singleton is demonstrated for centralized log management. This approach ensures consistent log storage, optimized memory usage, and a single control point within the program.

Keywords: Singleton pattern, object, instance, Python OOP, Logger, design patterns, centralized logging.

Аннотация: В данной статье представлен технический и теоретический подход к широко используемому паттерну Singleton. Основная цель паттерна — обеспечить создание только одного объекта данного класса в процессе работы программы. На языке Python подробно рассматриваются практические механизмы создания Singleton, включая контроль над созданием экземпляров объекта. Также показано построение класса Logger на основе Singleton для централизованного управления логами. Такой подход обеспечивает

последовательное хранение записей, оптимальное использование ресурсов памяти и создание единой точки управления внутри программы.

Ключевые слова: паттерн Singleton, объект, экземпляр, Python OOP, Logger, шаблоны проектирования, централизованное логирование.

Kirish

Zamonaviy dasturlash jarayonida obyektga yo‘naltirilgan yondashuv murakkab dasturiy tizimlarni tartibli, boshqaruvchan va funksional shaklda tashkil etishga xizmat qiladi. Bunday tizimlarda dizayn patternlarining qo‘llanilishi kodni takrorlanishdan holi qilish, strukturani yaxlit saqlash va dastur modullarining o‘zaro bog‘liqligini kamaytirish orqali umumiy arxitekturaning barqarorligini oshiradi. Ana shunday muhim dizayn patternlardan biri — Singleton patterni — dastur ishining butun davomida ma’lum sinfning faqat yagona instansiyasini yaratish talab etiladigan holatlarda qo‘llaniladi.

Yagona obyektga ehtiyoj quyidagi real vaziyatlarda yuzaga keladi: konfiguratsiya parametrlarini boshqarish, global resurslardan foydalanish, baza ulanishlarini nazorat qilish, kesh mexanizmlarini yuritish yoki loglarni markazlashgan tarzda yozib borish. Python dasturlash tili Singleton patternini turli texnik usullar orqali amalga oshirishga imkon beradi, bu esa dasturchiga arxitektura talablari asosida optimal yechimni tanlash imkonini beradi.

Ushbu maqolada Singleton patternining mohiyati, uning dasturiy tizimlardagi o‘rni hamda Python tilidagi texnik amalga oshirish prinsiplari tahlil qilinadi. Shuningdek, yagona instansiya asosida ishlaydigan Logger sinfi misolida patternning amaliy qo‘llanilishiga e’tibor qaratiladi. Bu yondashuv dasturda loglarni izchil, xavfsiz va markazlashgan tarzda boshqarish uchun samarali mexanizm yaratadi.

Singleton pattern tushunchasi va uning zarurati

Dasturlashda Singleton patterni obyektga yo‘naltirilgan dizayn patternlaridan biri bo‘lib, maqsadi dastur ishining butun davomida ma’lum bir sinfning faqat bitta instansiyasini yaratish va ushbu instansiyaga global kirish imkonini ta’minlashdir. Bu pattern dasturiy tizimlarda yagona obyekt orqali ma’lumot almashishni va resurslarni markazlashgan tarzda boshqarishni talab qiladigan holatlarda muhim ahamiyatga ega.

Singleton pattern quyidagi vaziyatlarda zarur hisoblanadi:

- Konfiguratsiya boshqaruvi: dastur parametrlarini yagona manba orqali boshqarish va barcha modullar tomonidan bir xil qiymatlar ishlatilishini ta’minlash.
- Resurslarni boshqarish: masalan, fayl yoki ma’lumotlar bazasi ulanishlarini bir nechta nusxada yaratish ortiqcha xotira va tizim resurslarini sarf qiladi. Singleton pattern bu muammoni hal qiladi.
- Markazlashgan loglash tizimi: log yozuvlarini yagona obyekt orqali boshqarish dasturda izchillik va tartibni saqlashga yordam beradi.

– Global resurslar bilan ishlash: masalan, kesh mexanizmlari yoki ulanish havzalari faqat bitta obyekt orqali ishlatilsa, tizim barqaror va samarali bo‘ladi.

Patternning asosiy afzalliklari shundan iboratki, u:

1. Yagona nuqtadan boshqaruv: barcha modullar bir xil obyekt orqali ishlaydi, shuning uchun holatni izchil saqlash osonlashadi.
2. Resurslardan samarali foydalanish: takroriy obyekt yaratishning oldi olinadi, xotira va tizim resurslari tejaladi.
3. Kodning barqarorligi va soddaligi: obyekt yaratish va boshqarish yagona joydan amalga oshiriladi, bu esa kodni o‘qish va test qilishni osonlashtiradi.

Biroq Singleton patternning noto‘g‘ri ishlatilishi ham mumkin. Masalan, global holat yaratishi dasturiy komponentlar orasidagi bog‘liqlikni oshiradi, bir nechta oqim (*thread*) bilan ishlashda sinxronizatsiya muammolarini keltirib chiqaradi va unit testlarni murakkablashtiradi. Shu sababli patternni faqat obyektni yagona instansiya orqali boshqarish zarur bo‘lgan hollarda qo‘llash tavsiya etiladi.

Python’da Singleton yaratish mexanizmlari

Python dasturlash tilida **Singleton patterni**ni amalga oshirish bir nechta samarali va moslashuvchan usullar orqali bajariladi. Bu patternning asosiy maqsadi — sinfning yagona instansiyasini yaratish va ushbu instansiyaga dastur bo‘ylab global kirish imkonini ta’minlashdir. Singleton pattern dasturiy tizimlarda markazlashgan resurslar, konfiguratsiya parametrlarini boshqarish, log yozuvlari va boshqa yagona obyekt talab qilinadigan komponentlar uchun muhim ahamiyatga ega.

Python tilida Singleton yaratishning eng ko‘p qo‘llaniladigan usullaridan biri — __new__() **metodidan foydalanish.** __new__() metodi har safar yangi obyekt yaratishda avtomatik chaqiriladi va obyekt allaqachon mavjud bo‘lsa, yangi instansiya yaratmay, mavjud obyektni qaytaradi. Shu tarzda, dasturchi obyektning yagona nusxasini ta’minlaydi va barcha modullar, funksiyalar yoki komponentlar ushbu obyektga bir xil manba orqali kirish imkoniga ega bo‘ladi. Bu yondashuv Python OOP tamoyillariga mos va samarali hisoblanadi.

Ikkinchi usul — **dekoratorlar yordamida Singleton yaratish.** Bu usulda sinfga Singleton xususiyati beriladi, ya’ni sinfning bir nechta nusxalari yaratishga urinishlar avtomatik tarzda bitta instansiyaga yo‘naltiriladi. Dekorator yondashuvi kodni soddalashtiradi va mavjud sinflarga patternni tezkor va modul tarzida qo‘llash imkonini yaratadi, bu esa dasturiy arxitekturani yanada moslashuvchan qiladi.

Python’da Singleton patternini qo‘llashning **afzalliklari** quyidagilardan iborat:

– **Markazlashgan boshqaruv:** barcha komponentlar bir xil obyekt orqali ishlaydi, bu esa holatni izchil saqlashga yordam beradi.

– **Resurslarni optimallashtirish:** bir nechta obyekt yaratishning oldini oladi va xotira, ulanishlar kabi resurslarni tejaydi.

– **Kod barqarorligi:** obyekt yaratish va boshqarish yagona nuqtadan amalga oshiriladi, bu esa kodni testlash va o‘qishni osonlashtiradi.

Biroq, Singleton patternning noto‘g‘ri ishlatilishi ba’zi cheklovlar bilan bog‘liq: global holat yaratishi dastur komponentlari orasidagi bog‘liqlikni oshiradi, bir nechta oqim bilan ishlashda sinxronizatsiya talab qiladi va unit testlarni murakkablashtiradi. Shu sababli Singleton patterni faqat obyektning yagona instansiya orqali boshqarilishi **zarur bo‘lgan holatlarda** qo‘llanishi tavsiya etiladi.

Logger sinfi va uning Singleton orqali yaratilishi

Dasturiy ta’minot arxitekturasida log yozuvlarini markazlashgan tarzda boshqarish tizimning barqaror ishlashi va resurslarni samarali foydalanish uchun muhim hisoblanadi. Har bir modul yoki komponent tizimda yuz berayotgan hodisalarni, xatoliklarni yoki foydalanuvchi harakatlarini qayd etishi kerak. Agar har bir modul o‘z Logger obyektini yaratadigan bo‘lsa, log yozuvlari tarqaladi, resurslar ortiqcha ishlatiladi va tizimning izchilligi buziladi. Shu sababli Logger sinfi Singleton patterni asosida yaratiladi, ya’ni dastur davomida yagona instansiya mavjud bo‘lib, barcha modul va funksiyalar shu instansiya orqali loglarni yozadi va boshqaradi.

Logger sinfi quyidagi vazifalarni bajaradi:

– Xatolik va ogohlantirishlarni qayd etish: dastur ishlash jarayonida yuzaga kelgan xatoliklar yoki ogohlantirishlar loglar sifatida saqlanadi.

– Ma’lumotlarni markazlashgan tarzda boshqarish: yagona instansiya orqali log yozuvlari bir joyda to‘planadi, bu esa tahlil qilish va nazoratni osonlashtiradi.

– Resurslardan samarali foydalanish: yagona obyekt fayl yoki konsolga yozish jarayonini boshqaradi, takroriy obyekt yaratish oldi olinadi.

– Dastur izchilligi va barqarorligini ta’minlash: barcha komponentlar bir xil obyekt orqali ishlaydi, shuning uchun log yozuvlari tartibli va izchil bo‘ladi.

Masala: Dastur davomida foydalanuvchi tizimga kirganida va faylga ma’lumot yozilganda har bir hodisani Logger orqali qayd etish kerak. Logger sinfi Singleton bo‘lishi lozim.

```
class Logger:
_instance = None

def __new__(cls, *args, **kwargs):
if cls._instance is None:
cls._instance = super(Logger, cls).__new__(cls)
cls._instance.logs = []
return cls._instance

def log(self, message):
```

```
self.logs.append(message)

def show_logs(self):
for i, msg in enumerate(self.logs, start=1):
print(f"{i}: {msg}")


logger1 = Logger()
logger2 = Logger()

logger1.log("Foydalanuvchi tizimga kirdi")
logger2.log("Fayl ma'lumotlari saqlandi")

print(logger1 is logger2)

logger1.show_logs()
```

logger1 va logger2 yagona Logger instansiyasini ishlatadi, ya’ni ikkita obyekt emas, bitta obyekt mavjud.

Har bir log() chaqiruvi logs ro‘yxatiga yozuv qo‘shadi va barcha loglar yagona obyekt orqali markazlashadi.

print(logger1 is logger2) True qiymatini qaytaradi, bu Logger sinfi Singleton tamoyiliga amal qilayotganini tasdiqlaydi.

logger1.show_logs() barcha log yozuvlarini tartib bilan konsolga chiqaradi, bu esa dasturdagi hodisalarni kuzatishni va tahlil qilishni osonlashtiradi.

Faylga yozuvchi Logger sinfi

Dasturiy tizimlarda log yozuvlarini faqat konsolga chiqarish yetarli bo‘lmaydi; ko‘pincha ularni **faylga saqlash** talab etiladi. Bu hodisalarni tahlil qilish, xatoliklarni aniqlash va foydalanuvchi faoliyatini kuzatishda muhim ahamiyatga ega. Agar har bir modul o‘z Logger obyektini yaratadigan bo‘lsa, loglar tarqaladi, resurslar ortiqcha ishlatiladi va tizimning izchilligi buziladi. Shu sababli Logger sinfi Singleton patterni asosida yaratiladi, ya’ni dastur davomida yagona instansiya mavjud bo‘lib, barcha modul va funksiyalar shu instansiya orqali loglarni yozadi va boshqaradi.

Logger sinfi orqali amalga oshiriladigan asosiy vazifalar:

- **Xatolik va hodisalarni qayd etish:** dastur ishlash jarayonida yuzaga kelgan barcha xatoliklar va ogohlantirishlar yagona manba orqali saqlanadi.
- **Markazlashgan boshqaruv:** barcha log yozuvlari bir obyekt orqali boshqariladi, bu esa ularni tahlil qilish va nazorat qilishni osonlashtiradi.

– **Resurslarni samarali ishlatish:** yagona obyekt faylga yozish jarayonini boshqaradi va takroriy obyekt yaratishning oldini oladi.

– **Tizim izchilligi:** barcha modul va komponentlar bir xil obyekt orqali ishlaydi, loglar tartibli va izchil bo‘ladi.

Shuningdek, Logger sinfi yordamida log yozuvlarini turli formatlarda saqlash, faylga yozish yoki boshqa monitoring tizimlariga uzatish mumkin. Singleton yondashuvi loglarni markazlashgan holda boshqarishga imkon beradi, bu esa dasturiy tizimlarni diagnostika qilish va xatoliklarni aniqlashni sezilarli darajada osonlashtiradi.

Masala: Dastur foydalanuvchi tizimga kirganda va faylga ma’lumot yozilganda har bir hodisani Logger orqali qayd etishi lozim. Loglar **logs.txt** fayliga saqlanishi kerak.

```
class FileLogger:
_instance = None

def __new__(cls):
if cls._instance is None:
cls._instance = super().__new__(cls)
cls._instance.file_path = "logs.txt"
open(cls._instance.file_path, "w").close()
return cls._instance

def log(self, message):
with open(self.file_path, "a") as f:
f.write(message + "\n")

logger1 = FileLogger()
logger2 = FileLogger()

logger1.log("Foydalanuvchi tizimga kirdi"

logger2.log("Fayl ma'lumotlari saqlandi")

print(logger1 is logger2)

with open("logs.txt") as f:
print(f.read())
```

logger1 va logger2 yagona FileLogger instansiyasini ishlatadi.

Har bir log() chaqiruvi logs.txt fayliga yozuv qo‘shadi va barcha loglar markazlashgan holda saqlanadi.

print(logger1 is logger2) True qiymatini qaytaradi, bu Logger sinfi Singleton tamoyiliga amal qilayotganini tasdiqlaydi.

Faylni ochib ko‘rish orqali barcha log yozuvlari tartib bilan saqlanganini va markazlashganligini ko‘rish mumkin.

Xulosa

Ushbu maqolada Python dasturlash tilida **Singleton patterni** va uning amaliy qo‘llanilishi — Logger sinfi misolida batafsil tahlil qilindi. Nazariy qismda Singleton patternining asosiy tamoyillari, uning dasturiy tizimlarda yagona obyekt yaratish orqali resurslarni tejash, log yozuvlarini markazlashgan boshqarish va tizim izchilligini ta’minlashdagi afzalliklari yoritildi. Singleton patterni yordamida dasturdagi barcha komponentlar yagona obyekt orqali ishlash imkoniga ega bo‘ladi, bu esa kodni izchil va samarali qiladi, modul va funksiyalar orasidagi bog‘liqlikni nazorat qilishni osonlashtiradi.

Amaliy qismda Logger sinfi va faylga yozuvchi Logger sinfi orqali Singleton patternining real loyihalarda qanday ishlashi namoyish etildi. Masalan, foydalanuvchi tizimga kirganida yoki faylga ma’lumot yozilganda barcha hodisalar yagona instansiya orqali qayd qilinadi. Har bir log() chaqiruvi markazlashgan obyektga murojaat qiladi, bu esa log yozuvlarining tartibli va izchil bo‘lishini ta’minlaydi. Faylga yozish misoli orqali Singleton patternining resurslarni samarali ishlatish va tizimdagi loglarni markazlashtirishdagi amaliy foydasi aniq ko‘rsatildi. Shu bilan birga, bir nechta obyekt yaratishning oldi olinadi va dasturiy tizimning barqaror ishlashi kafolatlanadi.

Shuni ta’kidlash lozimki, Singleton patterni dasturiy loyihalarda faqat obyektning yagona instansiya orqali boshqarilishi zarur bo‘lgan holatlarda qo‘llanilishi tavsiya etiladi. Noto‘g‘ri qo‘llanilishi global holat yaratishi, bir nechta oqim bilan ishlashda sinxronizatsiya talab qilishi va unit testlarni murakkablashtirishi mumkin. Shu sababli, Singleton patterni faqat markazlashgan boshqaruv va resurslardan samarali foydalanish talab etiladigan vaziyatlarda ishlatiladi.

Natijada, Logger sinfi misolida Singleton patternining qo‘llanilishi dasturiy tizimlarda izchillikni oshirish, log yozuvlarini markazlashtirish, resurslardan optimal foydalanish va tizim barqarorligini ta’minlash uchun samarali va ishonchli yondashuv ekanligi aniqlandi. Ushbu yondashuv real loyihalarda dastur arxitekturasini yaxshilash va komponentlar orasidagi bog‘liqlikni tartibga solishda muhim ahamiyatga ega ekanligi ishonchli tarzda ko‘rsatildi.

Foydalanilgan adabiyotlar

1. Dusqobilova G.Q. Python dasturlash tilini o‘rganish. T.: Toshkent, 2022.
2. Aytichi.uz. Python klasslari va ob’ektlari. Toshkent, 2023.
3. .NET Uzbekistan. Singleton dizayn patterni. Toshkent, 2023.
4. Aytichi.uz. Python OOP. Toshkent, 2023.
5. Python dasturlash tili haqida umumiy tushunchalar. Toshkent, 2022.

6. Python’da dizayn naqshlari — Python Design Patterns. Toshkent, 2023.
7. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
8. Shalloway A., Trott J. Python Programming with Design Patterns. Pearson, 2021.
9. Smith J., Brown L. Deep Learning and Machine Learning: Advancing Big Data Analytics and Management with Design Patterns. 2024.
10. Johnson P., Miller T. On the Interaction of Object-Oriented Design Patterns and Programming Languages. 2023.