Wzorzec projektowy Singleton

Aleksandra Połacik
Uniwersytet im. Jana Długosza w Częstochowie

29 stycznia 2024

Wzorzec Singleton w inżynierii oprogramowania

- Wzorzec Singleton to jedno z podstawowych podejść w projektowaniu oprogramowania.
  - Jest to wzorzec projektowy, który ma na celu ograniczenie instancjonowania danej klasy do pojedynczej instancji.
  - Oznacza to, że w ramach danej aplikacji istnieje tylko jedna instancja tej klasy, niezależnie od liczby prób utworzenia innych instancji.
  - Wzorzec został wymyślony głównie po to, aby umożliwić programiście stworzenie jednego obiektu do obsługi zadań przekrojowych dla całej aplikacji. Na przykład, jeśli nasz program nawiązuje połączenie z bazą danych, to będziemy chcieli mieć tylko jeden obiekt utrzymujący to połączenie.

Wzorzec Singleton w inżynierii oprogramowania (cd.)

- Celem wzorca Singleton jest zapewnienie, że klasa ma tylko jedną instancję i dostarcza globalny punkt dostępu do tej instancji.
  - Zapobiega to przypadkowemu tworzeniu wielu instancji klasy i pomaga w zarządzaniu globalnym stanem aplikacji.
  - Umożliwia jednolity dostęp do instancji klasy z dowolnego miejsca w programie, co jest szczególnie przydatne w przypadku, gdy wiele komponentów musi współpracować z tą samą instancją.

Wprowadzenie do problemu (Część 1/2)

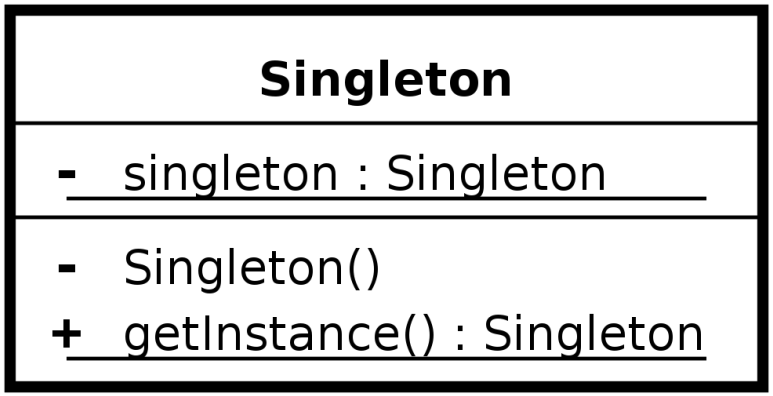
- Często w projektach występuje potrzeba posiadania tylko jednej instancji konkretnej klasy.
  - Przy projektowaniu oprogramowania często pojawia się sytuacja, w której chcielibyśmy, aby tylko jedna instancja danej klasy istniała w ramach aplikacji.
  - Przykładem takiej sytuacji może być potrzeba stworzenia jednego obiektu odpowiedzialnego za zarządzanie konfiguracją, jednolite logowanie, dostęp do wspólnych zasobów lub inny globalny stan.

Wprowadzenie do problemu (Część 2/2)

- Problem polega na tym, jak zapewnić, że tylko jedna instancja klasy zostanie utworzona i dostarczona w trakcie działania programu.
  - W jaki sposób możemy zagwarantować, że w systemie nie zostaną utworzone przypadkowe lub niepotrzebne instancje tej klasy?
  - Jak zapewnić, że dostęp do tej instancji będzie łatwy i globalny, aby inne komponenty mogły z niej korzystać, gdy jest to potrzebne?
  - Właśnie tutaj wzorzec Singleton staje się rozwiązaniem. Pozwala on na kontrolowane tworzenie i dostęp do jednej instancji klasy w sposób, który jest zarówno bezpieczny, jak i wydajny.

Diagram UML

- Diagram UML przedstawia klasę Singleton, zaznaczając, że istnieje tylko jedna instancja tej klasy.
- Diagram pokazuje także metodę dostępu do instancji Singletona.



Rysunek: Diagram UML

Implementacja wzorca Singleton

- Prywatny konstruktor klasy, aby uniemożliwić tworzenie instancji z zewnątrz.
  - Wzorzec Singleton wymaga, aby konstruktor klasy był prywatny, co oznacza, że nie można go wywołać spoza klasy. To zapobiega przypadkowemu tworzeniu wielu instancji.
- Prywatne pole przechowujące instancję klasy.
  - Klasa Singleton posiada prywatne pole, które przechowuje jedyną instancję klasy. To pole jest inicjowane wewnętrznie i jest niedostępne z zewnątrz.

Implementacja wzorca Singleton cd.

- Metoda dostępu, która tworzy instancję, jeśli nie istnieje, lub zwraca istniejącą.
  - Singleton udostępnia publiczną metodę dostępu, np. "getInstance(),"(nazwa może być inna lecz takie nazewnictwo przyjęło się i stanowi niepisaną regułę, którą warto stosować. To, co jest ważne, to brak parametrów) która sprawdza, czy instancja już istnieje. Jeśli nie istnieje, tworzy ją, a następnie zwraca istniejącą instancję. To zapewnia, że zawsze korzystamy z jednej instancji.
- Zabezpieczenia wielowątkowości, aby uniknąć równoczesnego tworzenia wielu instancji.
  - Aby zapewnić, że Singleton działa poprawnie w środowiskach wielowątkowych, można zastosować mechanizmy zabezpieczające, takie jak podwójne sprawdzenie (double-checked locking) lub wykorzystanie klasy synchronizującej.
  - Przykład użycia wzorca Singleton w języku Java Singleton.java

Przykład implementacji w języku Java (część 1/3)

```java
public class Singleton {
    // Prywatne pole przechowujace instancje klasy
    private static Singleton instance;

    private Singleton() {
        // Inicjalizacja instancji, mozna dodac dodatkowa logike inicjalizacji
    }
    // Metoda dostepu, ktora tworzy instancje, jesli nie istnieje, lub zwraca istniejaca
    public static Singleton getInstance() {
        if (instance == null) {
            synchronized (Singleton.class) {
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }
```

Klasa Singleton w języku Java (część 2/3)

```
1       // Dodatkowe metody klasy Singleton
2       public void doSomething() {
3        // Logika operacji wykonywanej przez Singleton
4           System.out.println("Singleton wykonuje jakas operacje.");
5       }
6   }
```

Klasa Main w języku Java (część 3/3)

```
1    public class Main {
2     public static void main(String[] args) {
3         // Uzyskanie instancji Singleton
4         Singleton singleton = Singleton.getInstance();
5
6         // Wywolanie metody na instancji Singleton
7         singleton.doSomething();
8     }
9 }
```

Zalety wzorca Singleton

- Oszczędność zasobów, ponieważ istnieje tylko jedna instancja.
  - Dzięki wzorcowi Singleton unikamy nadmiernego zużycia pamięci i zasobów, ponieważ istnieje tylko jedna instancja klasy w czasie działania programu. To szczególnie istotne w przypadku obiektów, które zużywają znaczące zasoby, takie jak połączenia do baz danych.
- Łatwy dostęp do tej instancji w dowolnym miejscu w kodzie.
  - Dzięki metodzie dostępu Singletonu, można łatwo uzyskać dostęp do jednej instancji klasy z dowolnego miejsca w kodzie. To ułatwia korzystanie z globalnych zasobów i zarządzanie nimi.
- Minimalizacja konfliktów w przypadku zarządzania współdzielonymi zasobami.
  - Singleton jest przydatny w sytuacjach, gdzie wiele części programu musi współdzielić dostęp do określonych zasobów. Dzięki Singletonowi możemy uniknąć konfliktów i zapewnić jednolity dostęp do tych zasobów.

Wady wzorca Singleton (Część 1/2)

- Może prowadzić do utrudnionego testowania, ze względu na trudności w zamianie instancji Singletona na fałszywą (mock) instancję w testach jednostkowych.
  - Wzorzec Singleton utrudnia testowanie jednostkowe, ponieważ instancja Singletona jest dostępna globalnie i nie można jej łatwo zastąpić fałszywą (mock) instancją podczas testów. To może skomplikować sprawdzanie zachowania innych komponentów w izolacji.

Wady wzorca Singleton (Część 2/2)

- Może wprowadzać punkt dostępu do stanu globalnego, co może prowadzić do trudniejszego zarządzania stanem aplikacji.
  - Korzystanie z Singletona może wprowadzać stan globalny, który jest dostępny z dowolnego miejsca w aplikacji. To może prowadzić do trudniejszego zarządzania stanem i rozwiązywania konfliktów związanymi z dostępem do globalnych zasobów. Dlatego ważne jest ostrożne wykorzystywanie wzorca Singleton i rozważenie, czy faktycznie jest on potrzebny w danym przypadku.
- Może być źródłem potencjalnych konfliktów w wielowątkowym środowisku.
  - W przypadku wielowątkowego środowiska, Singleton może wymagać dodatkowych zabezpieczeń, aby zapobiec równoczesnemu tworzeniu wielu instancji. To może wprowadzać dodatkową złożoność w kodzie.

Przykłady zastosowań

- Logger: Singleton może być użyty do rejestrowania zdarzeń w aplikacji.
    - W przypadku logowania zdarzeń w aplikacji, Singleton może zapewnić jednolity punkt dostępu do mechanizmu logowania. Każda część aplikacji może korzystać z Singletona, aby zapisywać informacje o zdarzeniach, błędach lub innym istotnym działaniu. Dzięki temu logi są spójne i zarządzane w jednym miejscu.
- Manager połączeń do bazy danych: Singleton może zarządzać i dostarczać połączenia do bazy danych.
    - Singleton może być użyty do stworzenia managera połączeń do bazy danych. Wielu komponentów w aplikacji może potrzebować dostępu do bazy danych, i Singleton może zapewnić jednolity punkt dostępu do zarządzania połączeniami. To pomaga uniknąć nadmiernego obciążenia bazy danych i zapewnia efektywne wykorzystanie połączeń.

Przykłady zastosowań cd.

- Konfiguracja aplikacji: Singleton może przechowywać konfigurację aplikacji.
  - Singleton może być wykorzystany do przechowywania globalnej konfiguracji aplikacji, takiej jak ustawienia ścieżek, parametry połączeń lub inne opcje. Dzięki temu konfiguracja jest jednolita i dostępna z każdego miejsca w aplikacji.

Alternatywne podejścia

- Inne wzorce projektowe, takie jak Dependency Injection, mogą zastępować Singleton w pewnych przypadkach.
  - W zależności od kontekstu i potrzeb projektu, istnieją inne wzorce projektowe, które mogą zapewnić elastyczniejsze i bardziej testowalne rozwiązania. Dependency Injection (DI) to jedno z takich podejść, które pozwala na wstrzykiwanie zależności do komponentów aplikacji. Jest to szczególnie przydatne w przypadku testowania jednostkowego, a także w sytuacjach, gdzie wymagane są zmienne implementacje interfejsów.
- Unikanie wzorca Singleton w sytuacjach, gdzie nie jest naprawdę potrzebny.
  - Warto rozważyć, czy wzorzec Singleton jest rzeczywiście konieczny w danym przypadku. Nie zawsze istnieje potrzeba globalnego punktu dostępu do jednej instancji klasy, a jego nadmierne użycie może prowadzić do skomplikowanego kodu i trudności w testowaniu. Przed zastosowaniem wzorca Singleton warto dokładnie ocenić potrzeby projektu.

Podsumowanie

- Wzorzec Singleton to fundamentalne podejście w projektowaniu oprogramowania, które pozwala na utworzenie tylko jednej instancji klasy.
- Kluczowe zalety wzorca Singleton to oszczędność zasobów, łatwy dostęp do instancji i minimalizacja konfliktów w zarządzaniu zasobami.
- Warto rozważyć jego wady, takie jak utrudnione testowanie i wprowadzenie stanu globalnego, zanim zdecydujesz się na jego użycie.
- Alternatywne wzorce projektowe, takie jak Dependency Injection, mogą być bardziej elastycznymi rozwiązaniami w niektórych przypadkach.
- Rozważ zastosowanie wzorca Singleton w kontekście konkretnych potrzeb projektu, aby zapewnić optymalne i elastyczne rozwiązania.

Dziękuję

Dziękuję za uwagę.