

WZORCE PROJEKTOWE

SPRAWOZDANIE

ZADANIE SINGLETON/2

Patryk Figas
Informatyka, programowanie
Grupa 34_Inf_P_NW_6

1. Cel

Celem tego dokumentu jest przedstawienie rozwiązania zadania polegającego na implementacji bufora wydruku zarządzającego zadaniami drukowania.

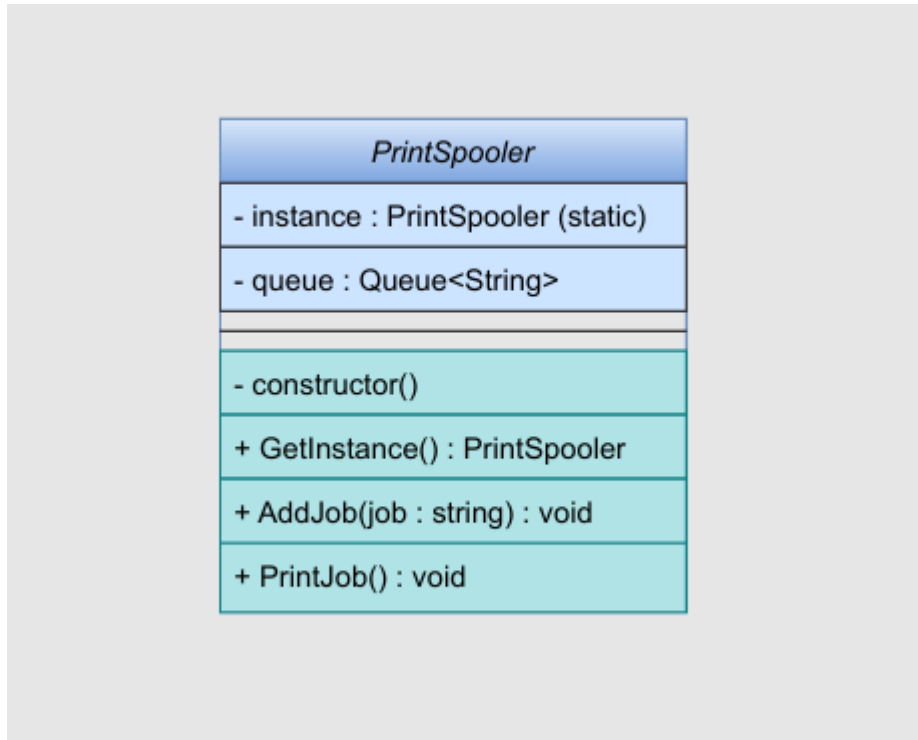
W ramach ćwiczenia zaprojektowano klasę zarządzającą kolejką zadań drukowania za pomocą „pseudokodu”, diagramu UML i implementacji klasy do programu oraz użycie jej w programie Main.

Użyto wzorca projektowego Singleton w celu zapewnienia, że istnieje tylko jedna instancja bufora wydruku w systemie.

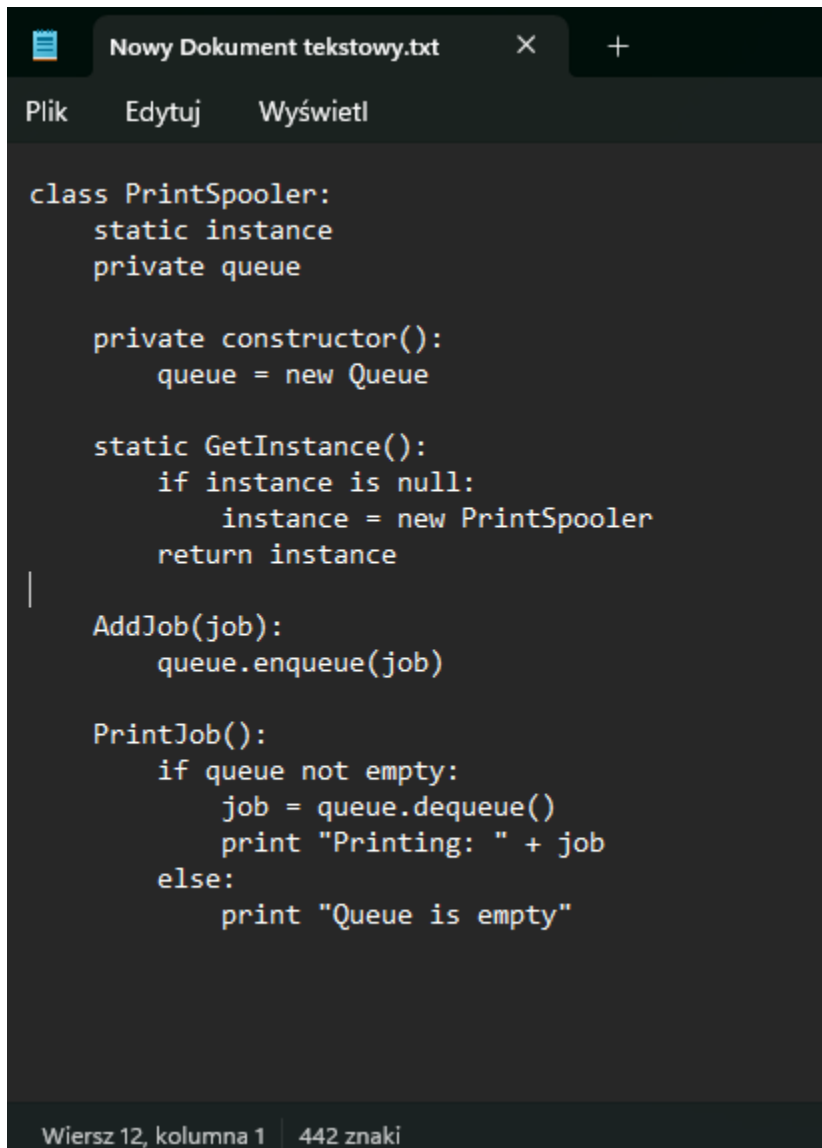
2. Opis rozwiązania

W celu kontrolowania dostępu do drukarki, zaprojektowano klasę **PrintSpooler**, która zarządza kolejką zadań drukowania. **Zastosowano wzorzec Singleton**, aby zagwarantować, że **tylko jedna instancja** tej klasy istnieje w **aplikacji**.

- diagram klasy **PrintSpooler**



- pseudokod klasy **PrintSpooler**



The image shows a screenshot of a text editor window titled "Nowy Dokument tekstowy.txt". The editor has a menu bar with "Plik", "Edytuj", and "Wyświetl". The main area contains the following pseudocode for the **PrintSpooler** class:

```
class PrintSpooler:
    static instance
    private queue

    private constructor():
        queue = new Queue

    static GetInstance():
        if instance is null:
            instance = new PrintSpooler
        return instance

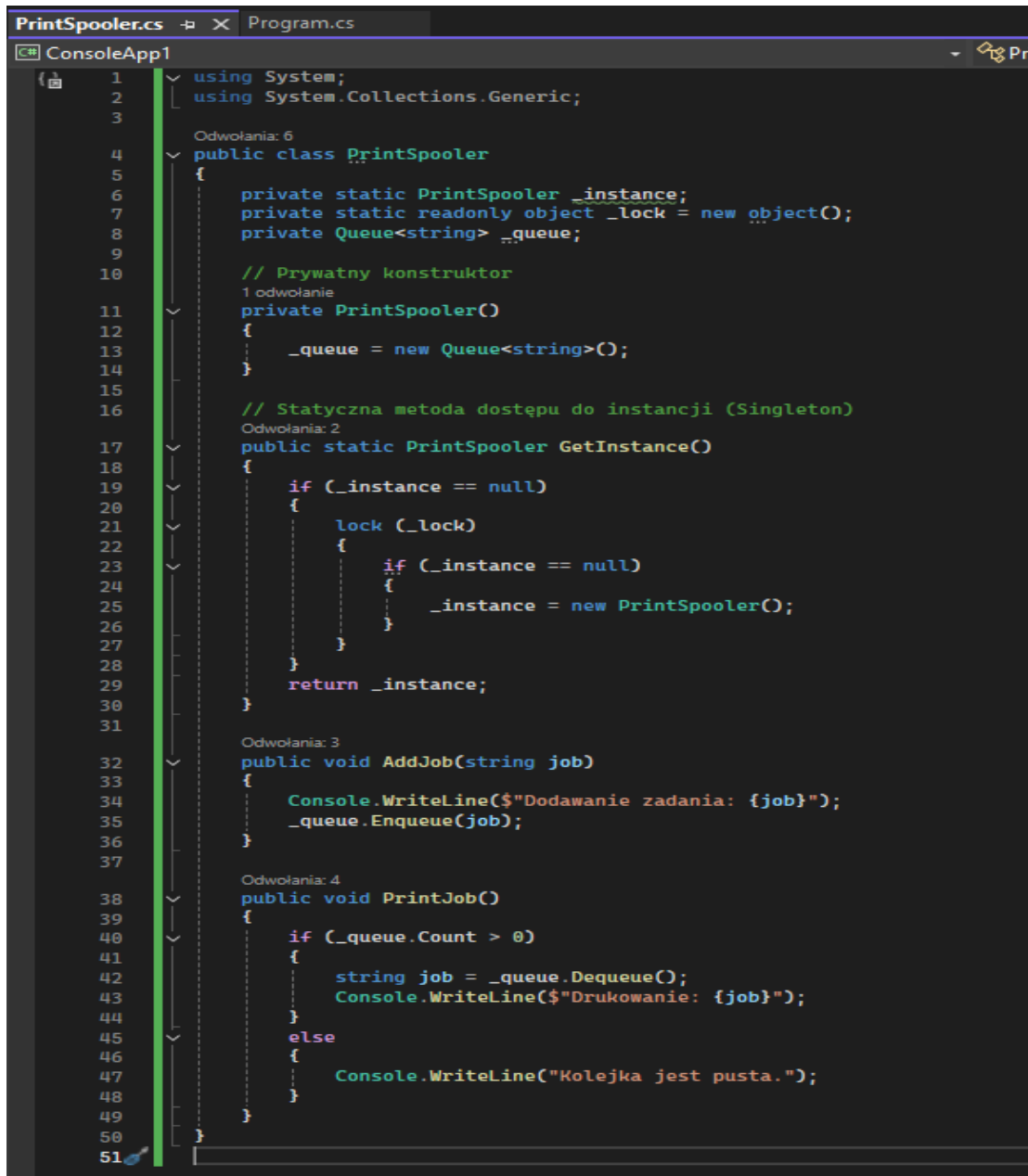
    AddJob(job):
        queue.enqueue(job)

    PrintJob():
        if queue not empty:
            job = queue.dequeue()
            print "Printing: " + job
        else:
            print "Queue is empty"
```

At the bottom of the editor, the status bar shows "Wiersz 12, kolumna 1" and "442 znaki".

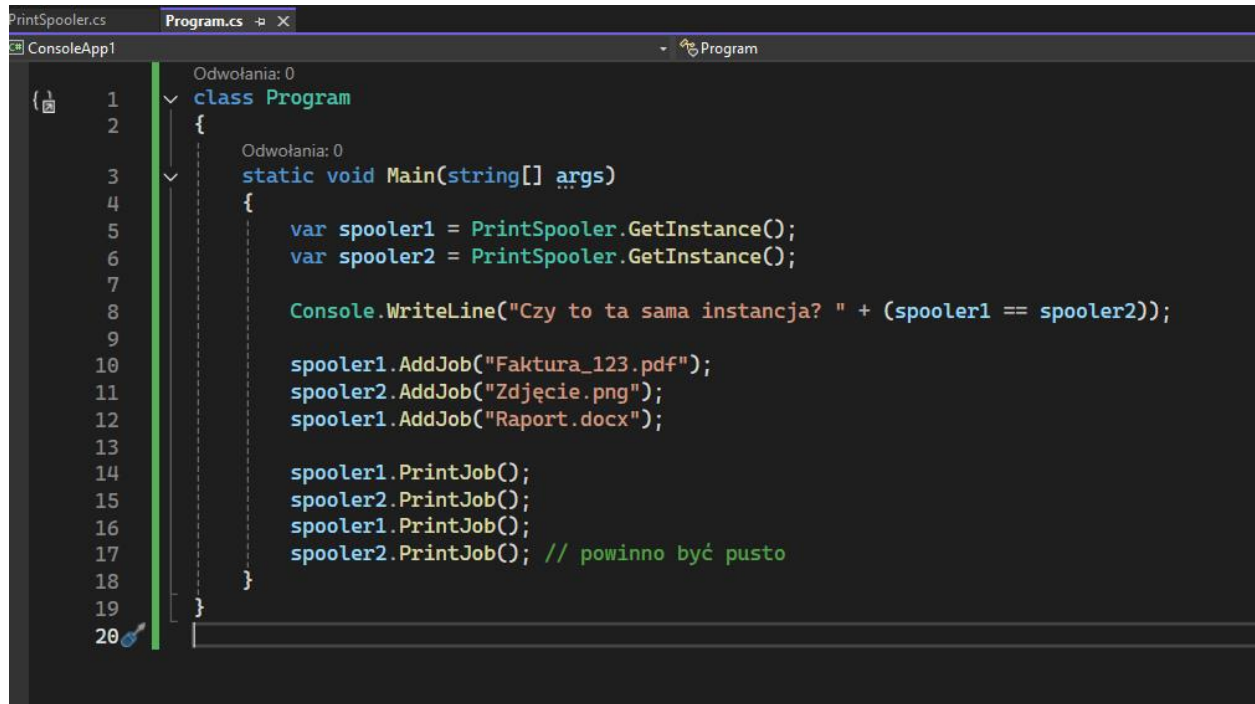
3. Implementacja

- kod klasy **PrintSpooler**



```
1 using System;
2 using System.Collections.Generic;
3
4 Odwołania: 6
5 public class PrintSpooler
6 {
7     private static PrintSpooler _instance;
8     private static readonly object _lock = new object();
9     private Queue<string> _queue;
10
11     // Prywatny konstruktor
12     1 odwołanie
13     private PrintSpooler()
14     {
15         _queue = new Queue<string>();
16     }
17
18     // Statyczna metoda dostępu do instancji (Singleton)
19     Odwołania: 2
20     public static PrintSpooler GetInstance()
21     {
22         if (_instance == null)
23         {
24             lock (_lock)
25             {
26                 if (_instance == null)
27                 {
28                     _instance = new PrintSpooler();
29                 }
30             }
31             return _instance;
32         }
33     }
34
35     Odwołania: 3
36     public void AddJob(string job)
37     {
38         Console.WriteLine($"Dodawanie zadania: {job}");
39         _queue.Enqueue(job);
40     }
41
42     Odwołania: 4
43     public void PrintJob()
44     {
45         if (_queue.Count > 0)
46         {
47             string job = _queue.Dequeue();
48             Console.WriteLine($"Drukowanie: {job}");
49         }
50         else
51         {
52             Console.WriteLine("Kolejka jest pusta.");
53         }
54     }
55 }
```

- kod programu **Main**



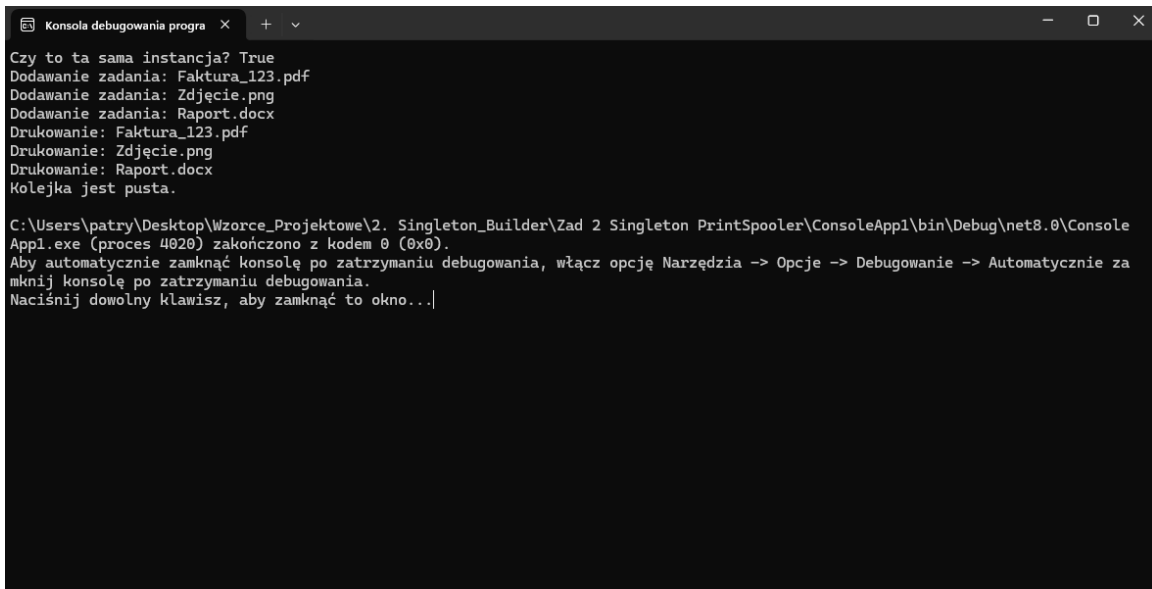
```
PrintSpooler.cs Program.cs X
ConsoleApp1 - Program

Odwolań: 0
1 class Program
2 {
3     Odwolań: 0
4     static void Main(string[] args)
5     {
6         var spooler1 = PrintSpooler.GetInstance();
7         var spooler2 = PrintSpooler.GetInstance();
8
9         Console.WriteLine("Czy to ta sama instancja? " + (spooler1 == spooler2));
10
11         spooler1.AddJob("Faktura_123.pdf");
12         spooler2.AddJob("Zdjęcie.png");
13         spooler1.AddJob("Raport.docx");
14
15         spooler1.PrintJob();
16         spooler2.PrintJob();
17         spooler1.PrintJob();
18         spooler2.PrintJob(); // powinno być pusto
19     }
20 }
```

4. Podsumowanie

W rozwiązaniu zadania zastosowano wzorec projektowy Singleton, ponieważ gwarantuje on, że tylko jedna instancja bufora drukowania będzie istniała w całej aplikacji.

Bufor drukowania działa prawidłowo i kontroluje kolejkę zadań. Wszystkie operacje drukowania przechodzą przez jedną instancję klasy, co zapewnia spójność.



```
Konsola debugowania progra x + v
Czy to ta sama instancja? True
Dodawanie zadania: Faktura_123.pdf
Dodawanie zadania: Zdjęcie.png
Dodawanie zadania: Raport.docx
Drukowanie: Faktura_123.pdf
Drukowanie: Zdjęcie.png
Drukowanie: Raport.docx
Kolejka jest pusta.

C:\Users\patry\Desktop\Wzorce_Projektowe\2. Singleton_Builder\Zad 2 Singleton PrintSpooler\ConsoleApp1\bin\Debug\net8.0\Console
App1.exe (proces 4020) zakończono z kodem 0 (0x0).
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie za
mknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...|
```

Alternatywnym wzorcem mógłby być **Factory Method** do tworzenia różnych typów drukarek, jednak w tym przypadku **Singleton** jest **najbardziej odpowiedni**, ponieważ **zależało mi na jednej, wspólnej instancji** zarządzającej wszystkimi zadaniami drukowania.

Lista załączników

Repozytorium GITHUB z projektem:

<https://github.com/PatrykFigas/Wzorce-projektowe.git>