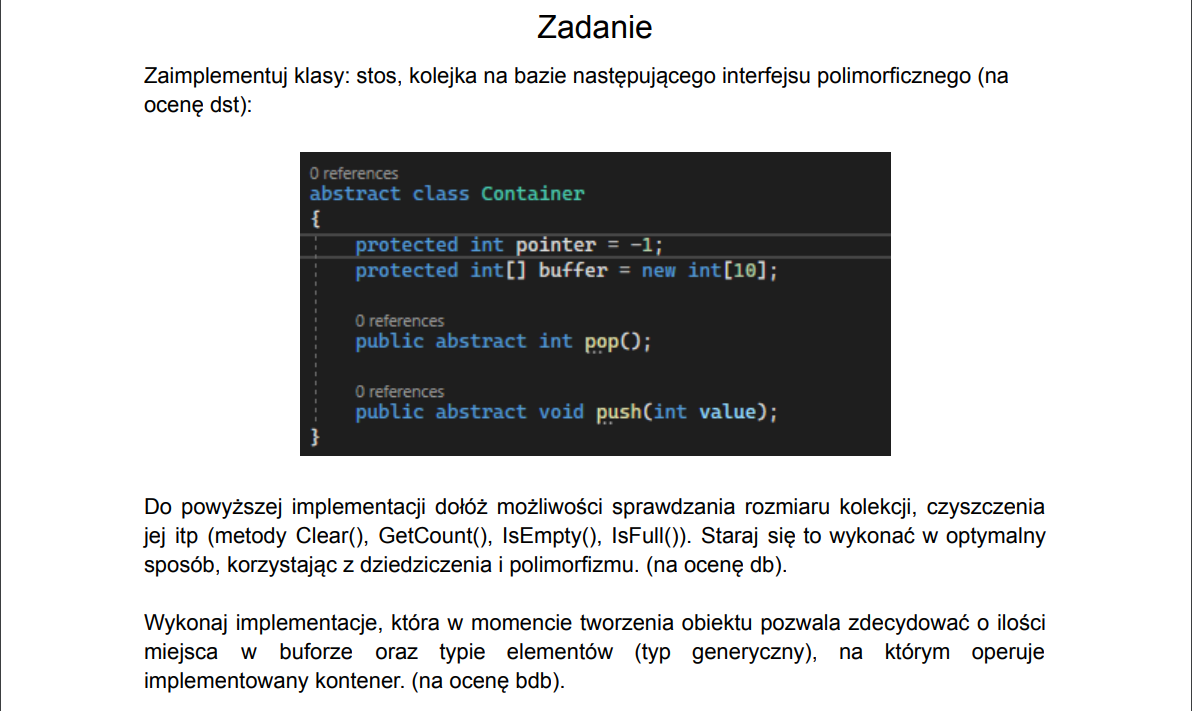
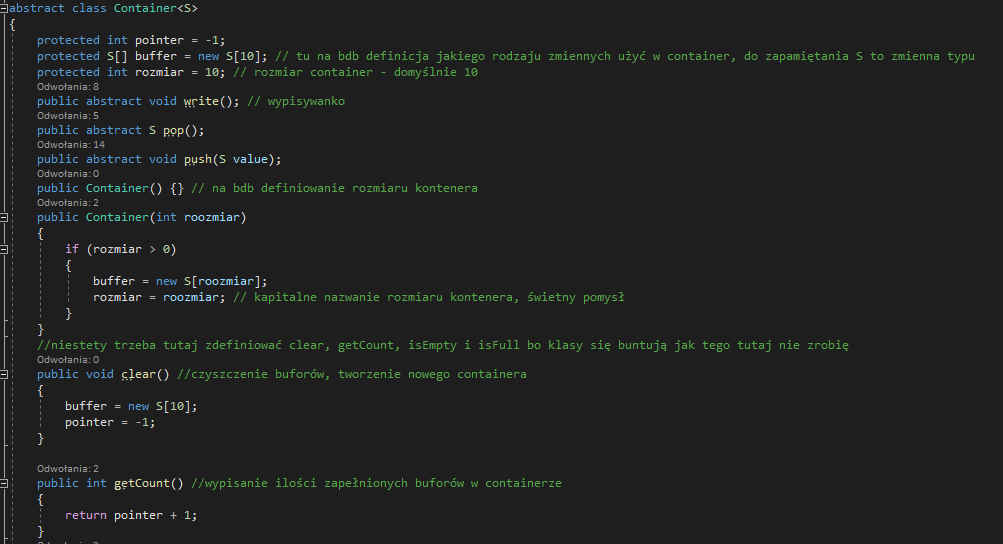
|  |  |
| --- | --- |
| **SPRAWOZDANIE** | |
| **Przedmiot** | **Temat** |
| Niezawodność systemów informatycznych | Stos i kolejka – C# - NSI1 |
| **Imię i nazwisko** | **Data wykonania zadania** |
| Konrad Kulka | 19.10.2022 r. |

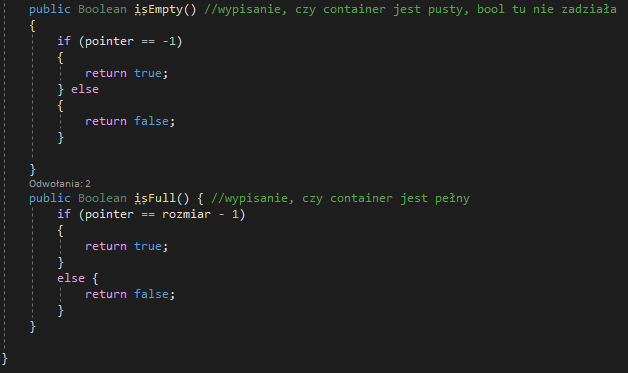
Zadanie:



Rozwiązanie:

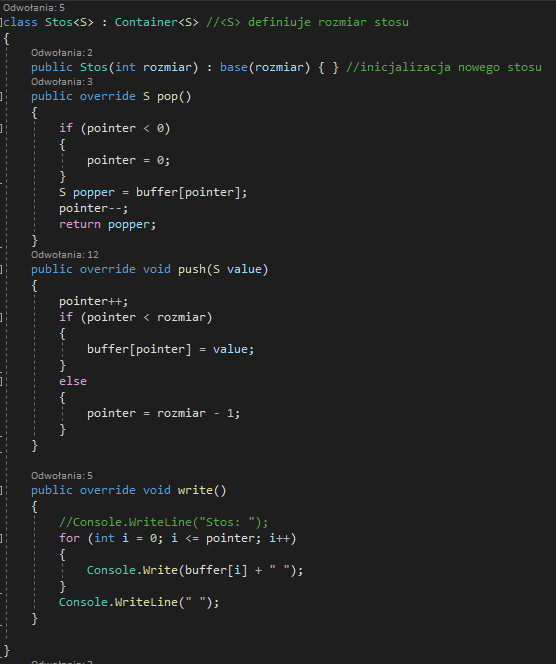
Na początku zdefiniowany został podstawowy interfejs polimorficzny, wykorzystany zostanie do rozwiązania zadania. Znajduje się w nim: definicja wskaźnika, dynamiczny rozmiar, wypisanie i przeliczenie zawartości, wypychanie, usuwanie i czyszczenie containera, sprawdzenie czy jest pusty i pełny:



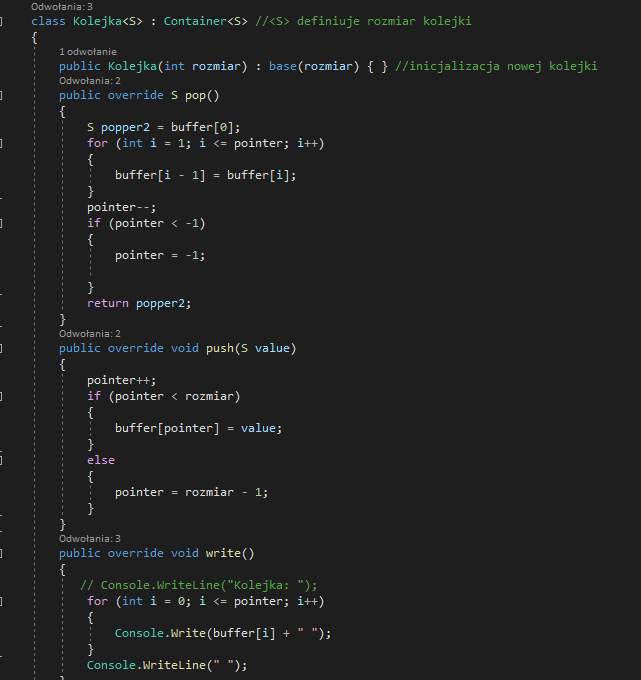


Aby zoptymalizować działanie programu, funkcje write(), clear(), getCount, isEmpty(), isFull() zostały zdefiniowane w głównej klasie Container, przez co nie zachodzi potrzeba definiowania ich przy definiowaniu klas dziedziczących.

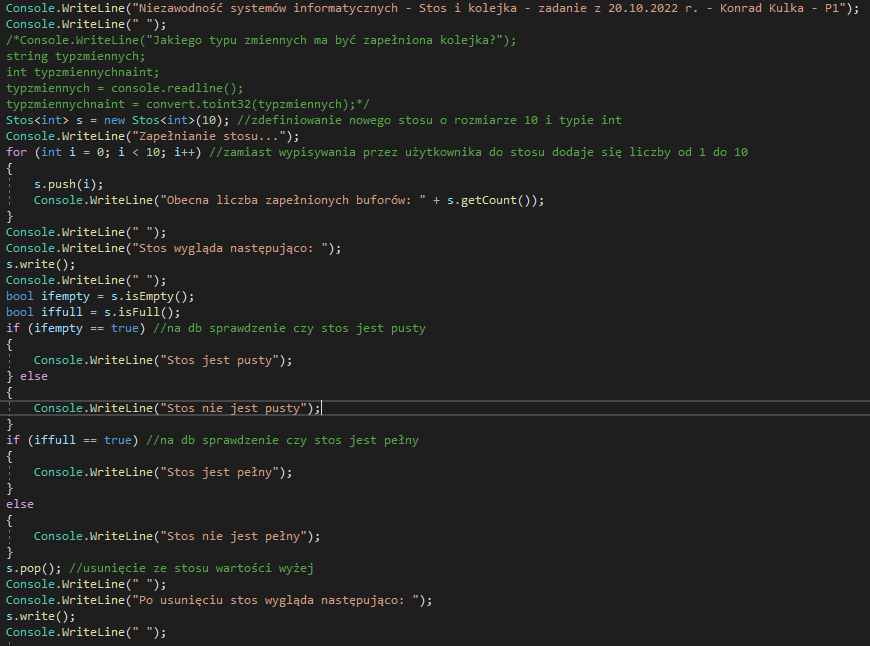
Następnie zdefiniowano klasę Stos, dziedzicząca z klasy Container właściwości. Dzięki temu będzie można nadpisywać przykładowo zmienną przeliczenia ilości zapełnionych buforów w stosie:



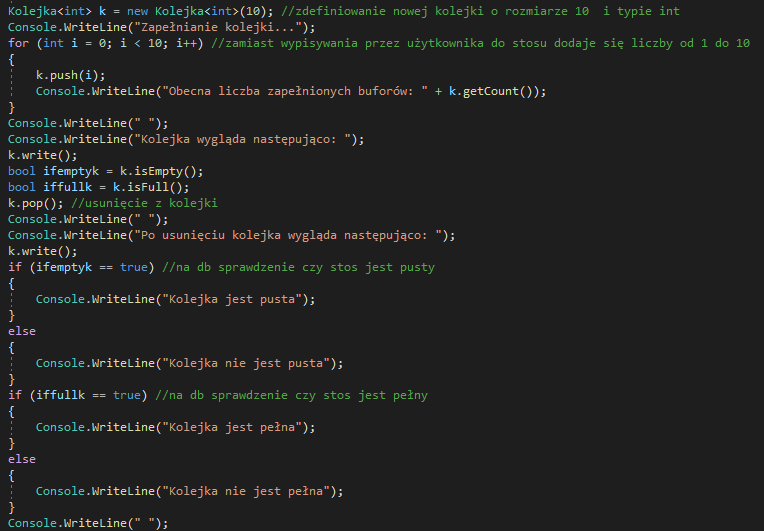
Następnie zdefiniowano klasę Kolejka, która tak jak klasa Stos również dziedziczy z klasy Container:



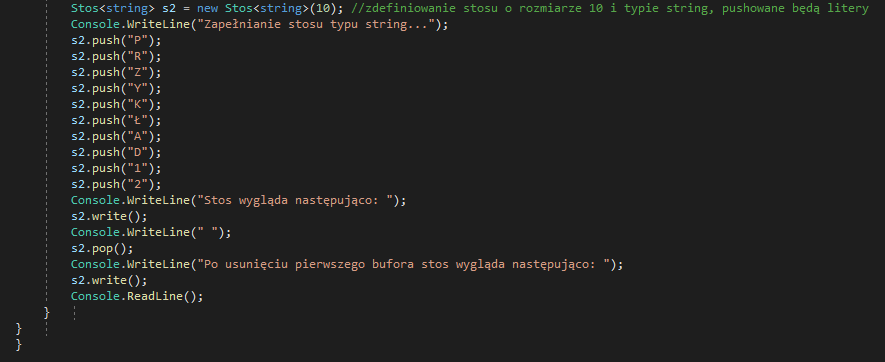
Po zdefiniowaniu klas Stos i Kolejka, można już zdefiniować klasę Program, w którym odbędą się operacje na klasach. Działa to w następującej kolejności: zdefiniowanie nowego stosu o rozmiarze 10, zapełnienie go liczbami od 0 do 9 - push, wypisywanie w trakcie zapełniania liczby zapełnionych buforów - getCount, wypisanie gotowego stosu - write, sprawdzenie, czy stos jest pusty i pełny - i isFull, usunięcie ze stosu - pop, ponowne wypisanie gotowego stosu w celu sprawdzenia, czy usunięcie przebiegło pomyślnie:



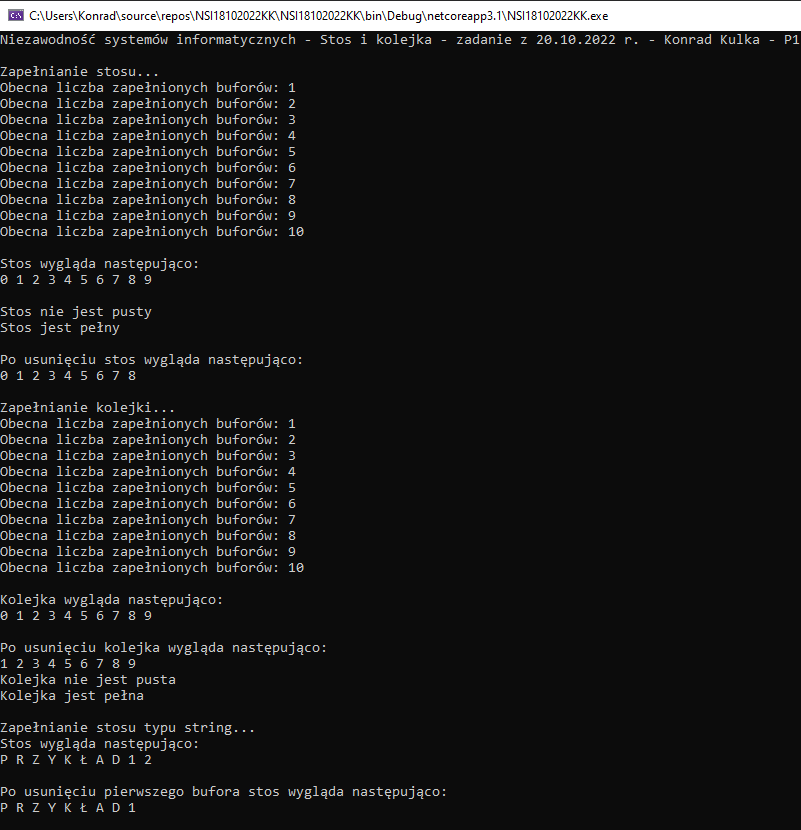
Podobnie jak w klasie Stos, następne działania zostaną podjęte w następującej kolejności: zdefiniowanie nowej kolejki o rozmiarze 10, zapełnienie jej liczbami od 0 do 9 - push, wypisywanie w trakcie zapełniania liczby zapełnionych buforów - getCount, wypisanie gotowej kolejki - write, sprawdzenie, czy kolejka jest pusta i pełna – isFull, isEmpty, usunięcie z kolejki - pop, ponowne wypisanie gotowego stosu w celu sprawdzenia, czy usunięcie przebiegło pomyślnie:



W celu sprawdzenia możliwości definicji integeru danych w stosie, zdefiniowano nowy stos wykorzystujący string i dokonano modyfikacji. Zamiast generowania liczb wypychane będą litery „na sztywno” - push,wypisanie – write, usunięcie ze stosu – pop, ponowne wypisanie:



Wynik:



Wnioski:

Wykonanie zadania, pomimo paru problemów natury technicznej (konfiguracja środowiska Visual Studio – na początku nie był w stanie skompilować programu, konsola samoistnie się wyłączała – rozwiązano przy pomocy Console.ReadLine() ) udało się pomyślnie rozwiązać zadanie. W celu ułatwienia rozwiązania i powtórki wiedzy z zakresu języka C#, a dokładniej z tematu polimorfizmu i interfejsów, skorzystano z dokumentacji zawartej na stronie w3schools.com. Zdobyta wiedza będzie przydatna w rozwiązywaniu kolejnych zadań z przedmiotu Niezawodność systemów informatycznych.