WOJSKOWA AKADEMIA TECHNICZNA

SPRAWOZDANIE Z PROJEKTU

Programowanie współbieżne

Mateusz Grabowski

Grupa WCY21IY4S1

Indeks: 76787

Data: 30.01.2023

1. Treść zadania

Nr zadania: 3/2022
Język implementacji: **C#**
Środowisko implementacyjne: VS2017 lub nowszy
Termin wykonania: **ostatnie zajęcia laboratoryjne**

Podstawowe wymagania:

- Warunkiem koniecznym zaliczenia jest poprawna identyfikacja zasobów dzielonych i synchronizacja dostępu do sekcji krytycznej z wykorzystaniem określonego w zadaniu mechanizmu synchronizacji. Program musi spełniać własność bezpieczeństwa i własność żywotności.
- Rozwiązanie powinno się cechować prostotą, ale jednocześnie elastycznością i skalowalnością.
- Liczba procesów sekwencyjnych musi być dobrana „z wyczuciem", tak, aby zachować czytelność interfejsu, a równocześnie umożliwić wizualizację rozwiązania postawionego problemu z uwzględnieniem ograniczonej w końcu zdolności percepcji osoby oceniającej.
- Kod źródłowy programu powinien być tak skonstruowany, aby istniał prosty sposób modyfikacji liczby procesów sekwencyjnych (za wyjątkiem zadań o ściśle określonej liczbie procesów).
- W przypadku niezaliczenia projektu w podejściu zasadniczym, istnieje możliwość zaliczenia w 2 podejściach poprawkowych (jeden przed wakacjami i jeden po wakacjach). Terminy podejść poprawkowych zostaną ustalone i podane przez prowadzącego.

Sprawozdanie (w formie elektronicznej) powinno zawierać następujące elementy:
1) stronę tytułową,
2) treść zadania,
3) krótki opis problemu, w szczególności przyjęte założenia,
4) wykaz zasobów dzielonych,
5) wykaz wyróżnionych sekcji krytycznych,
6) wykaz obiektów synchronizacji,
7) wykaz procesów sekwencyjnych,
8) wnioski

Problem do rozwiązania:

Jednokierunkowa droga przecina rzekę, na której nie ma mostu. Przeprawę na drugi brzeg obsługuje prom. Obowiązują pewne zasady dotyczące kursowania promu. Prom w jedną stronę przewozi samochody, a z powrotem wraca pusty. Odpływa z lewego brzegu, gdy liczba samochodów będących już na pokładzie osiągnie pewien ustalony limit (pojemność promu) lub gdy upłynął ustalony czas oczekiwania na samochody (próg cierpliwości). W drugim przypadku prom może odpłynąć tylko wtedy, gdy na pokładzie jest przynajmniej jeden samochód. Jeśli jest nadal pusty, to czas oczekiwania liczy się od nowa. Samochody wjeżdżają i zjeżdżają z promu pojedynczo.

2.

Klasa samochód tworzy obiekt samochodu, który widać na ekranie. Zaimplementowano w niej metody i atrybuty związane z stworzeniem obiektu samochodu oraz przedstawieniem go na graficznym interfejsie.

Klasa Lewy_brzeg zawiera w sobie kolejkę samochodów, które stanowią zasób współdzielony. Uwzględnia się w niej metody dodawania samochodów na brzeg i usuwania ich, gdy te wjeżdżają na statek.

W klasie Prom istnieją dwie metody obsługiwane przez wątki. Metoda Watek_Miejsce jest obsługiwana przez 3 wątki, które odpowiadają za pobieranie samochodów z lewego brzegu i ustawienie ich na miejscach na statku w losowej kolejności. Lock() zapewnia, że samochody są ładowane pojedynczo. Metodę Przemieszczanie() obsługuje czwarty wątek, który odpowiada za przemieszczanie się statkiem. W celu zsynchronizowania procesów, zastosowano zamek lock(). Gdy samochody wjeżdżają na statek, statek nie może odpłynąć. Tak samo gdy statek płynie, 3 pierwsze wątki nie mogą zabierać samochodów z brzegu i wstawiać je na statek. W metodzie Watek_Miejsce() oraz Przemieszczanie() zastosowano również monitor z funkcją Wait() oraz Pulse(). Gdy jeden samochód dostanie się na statek, zwalnia się lock, po czym uaktywnia się wątek związany z przemieszczaniem. Jednak zanim to nastąpi, statek czeka jeszcze 3 sekundy na pojawienie się kolejnych samochodów. Metodą Wait() blokuje własny wątek i odblokowuje zamek. Jeżeli są jakieś samochody, są one dodawane na statek i sygnalizują metodą Pulse() o zmianie. Lock jest znowu blokowany i kontynuuje się wątek statku.

Założono, że statek nie odpływa, gdy jest pusty i czeka na lewym brzegu. Samochody są dodawane za pomocą przycisku metodą dodajPojazd_Click() w pliku MainWindow.xaml.cs. Sam statek jest inicjalizowany w tym samym pliku w konstruktorze klasy MainWindow.

3.

Zasoby dzielone stanowią samochody umieszczone w kolejce tworzonej w klasie Lewy_brzeg. Są przedstawione za pomocą zielonych kwadratów:

```
public List<Samochod> _queue = new List<Samochod>();
```

4.

Sekcją krytyczną jest załadowywanie statku oraz przemieszczanie się statku. Dzięki zastosowaniu lock(), samochody są ładowane pojedynczo oraz nie w momencie, gdy statek jest w ruchu. Jeśli dodano dwa samochody na brzegu i po załadowaniu pierwszego samochodu, wątek wywołujący metodę przemieszczania zacznie się wykonywać zanim inny wątek zdąży załadować drugi pojazd, wtedy za pomocą klasy Stopwatch oraz monitora statek zanim odpłynie, musi zaczekać, żeby upewnić się czy może wziąć jeszcze jakiś samochód.

```
public void Watek_Miejsce(int i) // pojedyncze miejsce
        {
            Samochod samochod = null;

            while (true)
            {

                Thread.Sleep(5000); // czas na dojazd na prom
                lock (_lock1)
                {
                    samochod = _lewy_Brzeg.Deq(); // proba pobrania samochodu z brzegu
                    if (samochod != null)
                    {
                        samochody[i] = samochod;
                        samochody[i].MoveX(190 + i * 40,0.01f);
                        samochody[i].setY(190);
                        Monitor.Pulse(_lock1);
                    }
                }

            }

        }


public void Przemieszczanie()
        {
```

```
            while (true)
            {

                //Thread.Sleep(3000);
                lock (_lock1)
                {
                    bool b = true;
                    for (int i = 0; i < 3; i++)
                    {
                        if (samochody[i] != null)
                            b = false;
                    }

                    if (b)
                        continue; // jesli wszystkie miejsca sa puste, to statek nie plynie
                    Stopwatch sw = Stopwatch.StartNew();
                    while (sw.ElapsedMilliseconds <= 3000)
                    {
                        bool c = true;
                        for (int i = 0; i < 3; i++)
                        {
                            if (samochody[i] == null)
                                c = false;
                        }

                        if (c)
                            break;// jezeli wszystkie miejsca sa zapelniane, statek plynie od
razu
                        Monitor.Wait(_lock1, TimeSpan.FromMilliseconds(3000 -
sw.ElapsedMilliseconds));
                    }

                    Thread.Sleep(500);
                    for (int i = 0; i < 3; i++)
                    {
                        if (samochody[i] != null)
                            samochody[i].MoveX(500 + 40 * i, 4);

                    }

                    Move(MAX_X, 4);
                    Thread.Sleep(5000);// czas na przeplyniecie w jedna strone

                    for (int i = 0; i < 3; i++)
                    {
                        if (samochody[i] != null)
                            samochody[i].Hide();
                        samochody[i] = null;
                    }

                    Move(MIN_X, 4);
                    Thread.Sleep(4000); // czas na przeplyniecie w druga strone

                }
            }
        }
```

5.

Obiekty synchronizacji:
`public readonly object _lock1 = new object();`

W metodzie Watek_Miejsce(): `Monitor.Pulse(_lock1);`
W metodzie Przemieszczanie(): `Monitor.Wait(_lock1, TimeSpan.FromMilliseconds(3000 - sw.ElapsedMilliseconds));`

6.

Procesami sekwencyjnymi jest procedura załadowywania statku oraz przemieszczanie się statku w sekcjach krytycznych:

```
samochod = _lewy_Brzeg.Deq();
if (samochod != null)
{
      samochody[i] = samochod;
      samochody[i].MoveX(190 + i * 40,0.01f);
      samochody[i].setY(190);
    // …
}


Thread.Sleep(500);
for (int i = 0; i < 3; i++)
{
      if (samochody[i] != null)
      samochody[i].MoveX(500 + 40 * i, 4);

}

Move(MAX_X, 4);
Thread.Sleep(5000);// czas na przeplyniecie w jedna strone

for (int i = 0; i < 3; i++)
{
      if (samochody[i] != null)
      samochody[i].Hide();
      samochody[i] = null;
}

Move(MIN_X, 4);
Thread.Sleep(4000); // czas na przeplyniecie w druga strone
```

7.

Wnioski: Program wizualnie przedstawia etap czekania statku(szary prostokąt) na samochody(zielone kwadraty) oraz etap transportu. Statek od razu odpływa, gdy zostanie zapełniony lub odpływa po dłuższym czasie, gdy nie jest zapełniony w całości. Jeżeli statek jest pusty i jest na lewym brzegu, to cały czas czeka. Nie dochodzi do sytuacji, gdzie więcej niż jeden wątek znajduje się w sekcji krytycznej. Każdy z uruchomionych czekających wątków ostatecznie wchodzi do swojej sekcji krytycznej. Użytkownik może tworzyć przyciskiem wiele samochodów, które będą ustawiane w kolejce i odpowiednio zarządzane przez działające wątki.