

Programowanie współbieżne
Sprawozdanie 4
Podstawowe problemy programowania współbieżnego – wariant 5

1. Cel zadania

Celem zadania jest zapoznanie się z podstawowymi problemami programowania współbieżnego.

Wariant piąty zadania przedstawia problem palaczy i dostawców.

2. Rozwiązanie problemu

Program został napisany w języku C#. Problem rozwiązaliśmy za pomocą mechanizmu monitorów.

Wariant zadania zakłada, że istnieje trzech dostawców, przy czym każdy z nich dostarcza inną parę spośród trzech produktów (tytoń, bibułka, zapalniczka). Po uruchomieniu programu zostają więc stworzone trzy różne obiekty klasy Provider.

[Funkcja Main klasy Program:]

```
Provider providerTP = new Provider(true, true, false, "TP");
Provider providerTM = new Provider(true, false, true, "TM");
Provider providerPM = new Provider(false, true, true, "PM");

List<Provider> providers = new List<Provider>();
providers.Add(providerTP);
providers.Add(providerTM);
providers.Add(providerPM);

providerTP.Start();
providerTM.Start();
providerPM.Start();
```

[...]

[Klasa Provider:]

```
class Provider : BaseThread{
    public string name;
    public bool tobacco = false;
    public bool papers = false;
    public bool matches = false;

    public Provider(bool _tobacco, bool _papers, bool _matches, string _name){
        tobacco = _tobacco;
        papers = _papers;
        matches = _matches;
        name = _name;
    }
    [...]
}
```

Aby łatwiej rozróżnić te obiekty nadaliśmy im nazwy odpowiadające produktom, które dostarczają:

providerTP dostarcza tytoń i bibułki (Tobacco & Papers), providerTM – tytoń i zapałki (Tobacco & Matches), a providerPM – bibułki i zapałki.

Każdy z dostawców jest osobnym wątkiem uruchamianym zaraz po ich stworzeniu.

Następnie w programie co sekundę tworzony jest obiekt jednego z trzech rodzajów palaczy (posiadających bibułki lub posiadających tytoń lub posiadających zapałki) – rodzaj palacza, czyli posiadana przez niego rzecz jest losowana w konstruktorze Palacza.

[funkcja Main klasy Program:]

```
int counter = -1;

System.Timers.Timer timer = new System.Timers.Timer();
timer.Elapsed += (sender, e) => { counter++; HandleTimerElapsed(counter
, providers); };
timer.Interval = 1000;
timer.Enabled = true;

if (Console.Read() == 'q')
    timer.Enabled = false;
[...]
```

[klasa Program]

```
static void HandleTimerElapsed(int counter, List<Provider> providers){
    CreateClient(counter, providers);
}

private static void CreateClient(int counter, List<Provider> providers)
{
    Smoker smoker = new Smoker(counter, providers);
    if (smoker.tobacco)
        Console.WriteLine("Przybył palacz " + smoker.name + " i posiada
tytoń");
    else if(smoker.papers)
        Console.WriteLine("Przybył palacz " + smoker.name + " i posiada
bibułki");
    else
        Console.WriteLine("Przybył palacz " + smoker.name + " i posiada
zapałki");

    smoker.CheckProviders();
}
```

[konstruktor obiektu palacza:]

```
class Smoker{
    public int name;
    public bool tobacco = false;
    public bool papers = false;
    public bool matches = false;

    List<Provider> providers;

    public Smoker(int _name, List<Provider> _providers){
        name = _name;
        providers = _providers;

        Random random = new Random();
        int rand = random.Next(0, 3);
        switch(rand){
            case 0:
                tobacco = true;
                break;
            case 1:
                papers = true;
                break;
            case 2:
                matches = true;
                break;
            default:
                break;
        }
    }
}
```

Po stworzeniu palacza wywoływana jest jego metoda **CheckProviders()**, w której palacz przeszukuje podaną mu listę wszystkich trzech dostawców w celu znalezienia tego, który nie dostarcza produktu posiadanego przez palacza (czyli notabene tego, który dostarcza produkty, których palacz potrzebuje).

[metoda CheckProviders() klasy Smoker:]

```
public void CheckProviders(){
    foreach(Provider p in providers){
        if(tobacco && !p.tobacco){
            Console.WriteLine("Palacz " + name + " znalazł dostawcę " + p.name);
            Transaction(p);
        }
        else if(papers && !p.papers){
            Console.WriteLine("Palacz " + name + " znalazł dostawcę " + p.name);
            Transaction(p);
        }
        else if(matches && !p.matches){
            Console.WriteLine("Palacz " + name + " znalazł dostawcę " + p.name);
            Transaction(p);
        }
    }
}
```

Po znalezieniu odpowiedniego dostawcy palacz chce rozpocząć z nim transakcję wywołując własną metodę Transaction(Provider p).

[metoda Transaction klasy Smoker:]

```
public void Transaction(Provider p){
    Console.WriteLine("Palacz " + name + " przychodzi do dostawcy " + p.name);
    Monitor.Enter(p);
    try{
        Console.WriteLine("Palacz " + name + " rozpoczyna transakcję z " + p.name);
        p.Transaction(this);
    }
    finally{
        Console.WriteLine("Palacz " + name + " kończy transakcję z " + p.name);
        Monitor.Exit(p);
        p.SmokerLeaves(this);
    }
}
```

Palacz próbuje uzyskać dostęp do transakcji z wybranym dostawcą (metody Transaction klasy Provider), jeśli jednak monitor zabrania dostępu do niego (dostawca może być zajęty transakcją z innym palaczem) jest ustawiany w kolejce procesów oczekujących na dostęp do dostawcy.

Kiedy palacz uzyska dostęp do dostawcy, monitor automatycznie blokuje dostęp do tego dostawcy innym procesom (wątkom). Po zakończeniu transakcji dostawca opuszcza monitor, a obiekt palacza jest niszczone.

Transakcja dostawcy z palaczem trwa losową ilość czasu (między 1.5 a 3 sekundy).

[Metoda Transaction klasy Provider:]

```
public void Transaction(Smoker s){
    Thread transaction = Thread.CurrentThread;

    Random random = new Random();
    int time = random.Next(1500, 3001);
    Console.WriteLine("Transakcja dostawcy " + name + " z palaczem " + s.name + " będzie trwała " + time / 1000.00f + " sekund");
    Thread.Sleep(time);
    Console.WriteLine("Koniec transakcji dostawcy " + name + " z palaczem " + s.name);
}
```

Użycie monitora zapewnia, że tylko jeden proces/obiekt może brać udział w transakcji z konkretnym dostawcą w tym samym czasie.

Przedstawienie działającego programu:

```
file:///E:/semestr VII/Programowanie Współbieżne/wspolbiezne/Palacze/Palacze/bin/Debug/Palacze.EXE
Przybył palacz 0 i posiada zapałki
Palacz 0 znalazł dostawcę TP
Palacz 0 przychodzi do dostawcy TP
Palacz 0 rozpoczyna transakcję z TP
Transakcja dostawcy TP z palaczem 0 będzie trwała 2,936 sekund
Przybył palacz 1 i posiada tytoń
Palacz 1 znalazł dostawcę PM
Palacz 1 przychodzi do dostawcy PM
Palacz 1 rozpoczyna transakcję z PM
Transakcja dostawcy PM z palaczem 1 będzie trwała 1,828 sekund
Przybył palacz 2 i posiada bibułki
Palacz 2 znalazł dostawcę TM
Palacz 2 przychodzi do dostawcy TM
Palacz 2 rozpoczyna transakcję z TM
Transakcja dostawcy TM z palaczem 2 będzie trwała 2,466 sekund
Koniec transakcji dostawcy PM z palaczem 1
Palacz 1 kończy transakcję z PM
Palacz 1 wychodzi
Koniec transakcji dostawcy TP z palaczem 0
Palacz 0 kończy transakcję z TP
Palacz 0 wychodzi
Przybył palacz 3 i posiada bibułki
Palacz 3 znalazł dostawcę TM
Palacz 3 przychodzi do dostawcy TM
Przybył palacz 4 i posiada zapałki
Palacz 4 znalazł dostawcę TP
Palacz 4 przychodzi do dostawcy TP
Palacz 4 rozpoczyna transakcję z TP
Transakcja dostawcy TP z palaczem 4 będzie trwała 2,535 sekund
Koniec transakcji dostawcy TM z palaczem 2
Palacz 2 kończy transakcję z TM
Palacz 2 wychodzi
Palacz 3 rozpoczyna transakcję z TM
Transakcja dostawcy TM z palaczem 3 będzie trwała 2,807 sekund
Przybył palacz 5 i posiada bibułki
Palacz 5 znalazł dostawcę TM
Palacz 5 przychodzi do dostawcy TM
Przybył palacz 6 i posiada tytoń
Palacz 6 znalazł dostawcę PM
Palacz 6 przychodzi do dostawcy PM
Palacz 6 rozpoczyna transakcję z PM
Transakcja dostawcy PM z palaczem 6 będzie trwała 1,887 sekund
Koniec transakcji dostawcy TP z palaczem 4
Palacz 4 kończy transakcję z TP
Palacz 4 wychodzi
Przybył palacz 7 i posiada bibułki
Palacz 7 znalazł dostawcę TM
Palacz 7 przychodzi do dostawcy TM
Koniec transakcji dostawcy TM z palaczem 3
Palacz 3 kończy transakcję z TM
Palacz 3 wychodzi
Palacz 5 rozpoczyna transakcję z TM
Transakcja dostawcy TM z palaczem 5 będzie trwała 1,934 sekund
Koniec transakcji dostawcy PM z palaczem 6
Palacz 6 kończy transakcję z PM
Palacz 6 wychodzi
Przybył palacz 8 i posiada zapałki
Palacz 8 znalazł dostawcę TP
Palacz 8 przychodzi do dostawcy TP
Palacz 8 rozpoczyna transakcję z TP
Transakcja dostawcy TP z palaczem 8 będzie trwała 2,918 sekund
Przybył palacz 9 i posiada zapałki
Palacz 9 znalazł dostawcę TP
Palacz 9 przychodzi do dostawcy TP
Koniec transakcji dostawcy TM z palaczem 5
Palacz 5 kończy transakcję z TM
Palacz 5 wychodzi
Palacz 7 rozpoczyna transakcję z TM
Transakcja dostawcy TM z palaczem 7 będzie trwała 1,841 sekund
Przybył palacz 10 i posiada bibułki
Palacz 10 znalazł dostawcę TM
Palacz 10 przychodzi do dostawcy TM
Koniec transakcji dostawcy TP z palaczem 8
Palacz 8 kończy transakcję z TP
Palacz 8 wychodzi
Palacz 9 rozpoczyna transakcję z TP
Transakcja dostawcy TP z palaczem 9 będzie trwała 2,208 sekund
Koniec transakcji dostawcy TM z palaczem 7
Palacz 7 kończy transakcję z TM
Palacz 7 wychodzi
Palacz 10 rozpoczyna transakcję z TM
```

3. Wnioski

Powyższa implementacja zapewnia uniknięcie problemu blokady, ze względu na monitor, który pozwala na operacje tylko między jednym palaczem i jednym dostawcą jednocześnie. Unikamy również problemu zagłodzenia - nasza implementacja powoduje, że każdy palacz zostanie w końcu dopuszczony do odpowiedniego dostawcy.

Przy rozwiązywaniu powyższego problemu palaczy papierosów musieliśmy wykluczyć użycie mechanizmu spotkań – palacz musi wiedzieć, do którego konkretnego dostawcy musi się udać oraz jaką jego metodę wywołać.