Daria Woźnicka *180744*
Adam Wiśniewski *180734*

Programowanie współbieżne
Sprawozdanie 2
Monitory – wariant 5

1. Cel zadania

Celem zadania jest poznanie monitorowego mechanizmu synchronizacji procesów i zasad jego działania.

Wariant piąty zadania przedstawia problem fryzjerów I minikurzystek.

2. Rozwiązanie problemu

Program został napisany w języku C#.

Po wprowadzeniu przez użytkownika liczby fryzerów I manikurzystek pracujących w salonie oraz pojemności poczekalni, tworzone są obiekty fryzjerów, manikurzystek oraz kierownika salonu.

```
    public int liczbaFryzjerow, liczbaManikiurzystek, pojemnoscPoc
zekalni;


    public void TakeNumbers(out int liczbaFryzjerow, out int liczb
aManikiurzystek, out int pojemnoscPoczekalni)
        {
            Console.WriteLine("Podaj liczbę fryzjerów: ");
            liczbaFryzjerow = int.Parse(Console.ReadLine());
            Console.WriteLine("Podaj liczbę manikiurzystek: ");
            liczbaManikiurzystek = int.Parse(Console.ReadLine());
            Console.WriteLine("Podaj pojemność poczekalni: ");
            pojemnoscPoczekalni = int.Parse(Console.ReadLine());
        }
```

```
  Logic logic = new Logic();
  logic.TakeNumbers(out logic.liczbaFryzjerow, out logic.liczbaMan
ikiurzystek, out logic.pojemnoscPoczekalni);
  Console.WriteLine("Licbza fryzjerów: " + logic.liczbaFryzjerow);
  Console.WriteLine("Liczba manikiurzystek: " + logic.liczbaManiki
urzystek);
  Console.WriteLine("Pojemność poczekalni: " + logic.pojemnoscPocz
ekalni);
```

```
Boss boss = new Boss(logic.liczbaFryzjerow, logic.liczbaManikiurzy
stek, logic.pojemnoscPoczekalni);
```

Daria Woźnicka *180744*
Adam Wiśniewski *180734*

```
public Boss(int barbersCount, int manicuristsCount, int waitingRoo
mCapacity)
        {
            for (int i = 0; i < barbersCount; i++)
            {
                barbers.Add(new Barber(i));
            }

            for (int i = 0; i < manicuristsCount; i++)
            {
                manicurists.Add(new Manicurist(i));
            }

            waiting_room = new List<Client>(waitingRoomCapacity);
        }
```

W naszym programie klienci wchodzą do salonu pojedynczo, co 10 minut (w skali programu – 1 sekundę).

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

        static void HandleTimerElapsed(Boss boss, int counter)
        {
            CreateClient(boss, counter);
        }

        private static void CreateClient(Boss boss, int counter)
        {
            Client client = new Client(counter);
            Console.WriteLine("Przybył klient " + client.name);
            boss.TakeClient(client);
        }
```

Po stworzeniu klienta wykonywana jest akcja kierownika salonu (TakeClient(client)), która przekierowuje klienta do poczekalni. Jeśli jest w niej miejsce, klient zajmuje ostatnie miejsce w poczekalni, jeśli nie, czeka na to miejsce losową ilość czasu (od 0 do 20 minut – w skali programu do 2 sekund).

```
   public void TakeClient(Client client)
        {
            bool wantManicure = client.manicure_wanted;
            bool entered = false;

            entered = CheckWaitingRoom(client);
          [...]
```

Daria Woźnicka *180744*
Adam Wiśniewski *180734*

Następnie, jeśli klientowi skończyła się cierpliwość, klient wychodzi (jego obiekt zostaje zniszczony).
Następnie kierownik sprawdza zapotrzebowanie kolejnych klientów salonu na fryzjerów, a następnie manikurzystki (Klienci którzy są aktualnie obsługiwani przez pracownika salonu mają pierwszeństwo, kolejni są klienci czekający w poczekalni).

```
          […]
          if (!entered)
            {
                Console.WriteLine("Klient " + client.name + " wych
odzi.");
                ClientLeaves(out client);
            }
            CheckClientsForManicure();
            CheckClientsForBarber();
        }
```

Metody CheckClientsForBarber() i CheckClientsForManicure() iterują kolejno po klientach obecnie obsługiwanych przez pracowników, a następnie klientach czekających w poczekalni.

Metoda CheckClientsForBarber():
Jeśli znajdzie klienta, który oczekuje na strzyżenie (w naszym programie zgodnie z treścią polecenia każdy klient żąda strzyżenia, a niektórzy również manicure'u), wysyła go do fryzjerów. Klient próbuje uzyskać usługę kolejnych fryzjerów, jeśli jednak monitor nie pozwala na uzyskanie dostępu do fryzjera, klient próbuje uzyskać dostęp u następnego. Jeśli znajdzie dostępnego fryzjera, fryzjer ten zajmuje monitor I rozpoczyna się strzyżenie (trwające między 10 a 30 minut – w skali programu między 1 a 3 sekundy). Jeśli nie znajdzie żadnego wolnego fryzjera (wszyscy fryzjerzy zajmują monitor) – klient wraca na swoje miejsce.

```
void CheckClientsForBarber(){
   Console.WriteLine("Kierownik sprawdza czy przekierować któregoś
z klientów do fryzjera.");
   if(clients_at_manicurist.Count > 0){
     foreach(Client c in clients_at_manicurist){
        if(!c.haircut_done && !c.haircut_in_progress){
          Console.WriteLine("Klient " + c.name + " jest w trakcie
manicure i potrzebuje strzyżenia.");
          CheckBarbers(c);
          break;}
      }
   }
   if (waiting_room.Count > 0){
     foreach (Client c in waiting_room){
       if (!c.haircut_done){
        Console.WriteLine("Klient " + c.name + " jest w poczekalni
```

```
i potrzebuje strzyżenia.");
       CheckBarbers(c);
       break;}
      }
   }
}
```

```
void CheckBarbers(Client client){
  bool freeBarber = false;
  Console.WriteLine("Szukam fryzjera dla klienta " + client.name);
  foreach(Barber b in barbers){
    Console.WriteLine("Sprawdzam fryzjera " + b.name);
    if (Monitor.TryEnter(b)){
      Console.WriteLine("Fryzjer " + b.name + " wolny. Wchodzi do
monitora");
      waiting_room.Remove(client);
      clients_at_barber.Add(client);
      freeBarber = true;
      try{
        client.Haircut(b);
        Monitor.Pulse(b);}
      finally{
        Monitor.Exit(b);
        Console.WriteLine("Fryzjer " + b.name + " opuszcza monitor
.");
        clients_at_barber.Remove(client);
        if(!client.manicure_wanted){
          Console.WriteLine("Klient " + client.name + " wychodzi."
);
          ClientLeaves(out client);}
        else if(client.manicure_wanted && !client.manicure_done){
           CheckWaitingRoom(client);}
        CheckClientsForBarber();}
      break;
    }
    else{
      Console.WriteLine("Fryzjer " + b.name + " zajęty.");}
  }
  if (!freeBarber){
     Console.WriteLine("Brak wolnych fryzjerów");}
}
```

```
public void Haircut(Barber barber){
  Console.WriteLine("Fryzjer " + barber.name + " rozpoczyna strzyz
enie klienta " + this.name);
  Random rand = new Random();
  int time = rand.Next(1000, 3001);
```

```
  Console.WriteLine("Strzyzenie bedzie trwac " + time / 100 + " mi
nut");
  haircut_in_progress = true;
  Thread.Sleep(time);
  Console.WriteLine("Fryzjer " + this.name + " koniec strzyżenia."
);
  haircut_done = true;
}
```

Analogicznie dzieje się w przypadku manicure (strzyżenie I manicure dzieje się naprzemiennie nie wykluczając się na wzajem – robienie manikiuru klientowi (strzyżenia) może rozpocząć się przed rozpoczęciem strzyżenia (robienia manikiuru), po rozpoczęciu strzyżenia (robienia manikiuru) lub też po całkowitym zakończeniu strzeżenia (robienia manikiuru)).

```
void CheckClientsForManicure(){
  Console.WriteLine("Kierownik sprawdza czy przekierować któregoś
z klientów do manikurzystki.");
  if (clients_at_barber.Count > 0){
    foreach(Client c in clients_at_barber){
      if(c.manicure_wanted && !c.manicure_done && !
c.manicure_in_progress){
        Console.WriteLine("Klient " + c.name + " jest w trakcie st
rzyżenia i chce otrzymać manicure.");
        CheckManicurist(c);
        break;}
    }
  }
  if(waiting_room.Count > 0){
     foreach (Client c in waiting_room){
       if (c.manicure_wanted && !c.manicure_done){
          Console.WriteLine("Klient " + c.name + " jest w poczekal
ni i chce otrzymać manicure.");
          CheckManicurist(c);
          break;}
     }
  }
}
```

```
void CheckManicurist(Client client){
   bool freeManicurist = false;
   Console.WriteLine("Szukam manikurzystki dla klienta " + client.
name);
   foreach (Manicurist m in manicurists){
      if (Monitor.TryEnter(m)){
        Console.WriteLine("Manikurzystka " + m.name + " wolna. Wch
odzi do monitora");
```

Daria Woźnicka *180744*
Adam Wiśniewski *180734*

```
        clients_at_manicurist.Add(client);
        waiting_room.Remove(client);
        freeManicurist = true;
        try{
          client.Manicure(m);
          Monitor.Pulse(m);}
        finally{
          Monitor.Exit(m);
          Console.WriteLine("Manikurzystka " + m.name + " opuszcza
monitor.");
          clients_at_manicurist.Remove(client);
          if (client.haircut_done){
             Console.WriteLine("Klient " + client.name + " wychodz
i.");
             ClientLeaves(out client);}
          else{
             CheckWaitingRoom(client);}
          CheckClientsForManicure();}
        break;}
       }
       if (!freeManicurist){
          Console.WriteLine("Brak wolnych manikurzystek");}
}
```

```
public void Manicure(Manicurist manicurist){
   Console.WriteLine("Manikurzystka " + manicurist.name + " rozpoc
zyna manicure klienta " + this.name);
   Console.WriteLine("Manicure trwa 15 minut.");
   manicure_in_progress = true;
   Thread.Sleep(1500);
   Console.WriteLine("Manikurzystka " + this.name + "koniec manicu
re.");
   manicure_done = true;
}
```

Daria Woźnicka *180744*
Adam Wiśniewski *180734*

Prezentacja działania programu: