НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі №3
«Моделювання систем»
«Побудова імітаційної моделі системи з використанням формалізму моделі масового обслуговування»

Студент: Галько М.В.
Група: ІП-01

Київ, 2023

ЗМІСТ

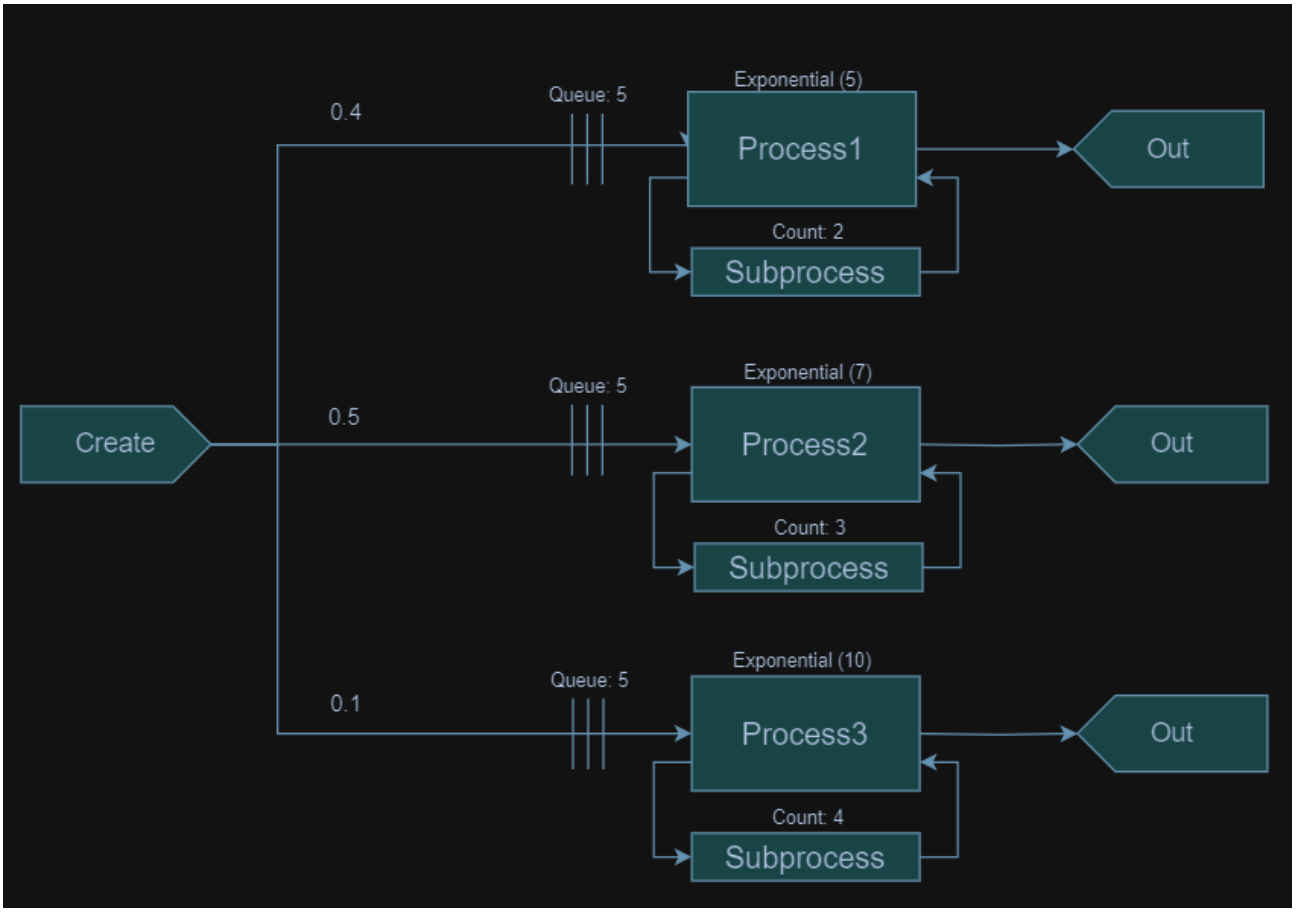
Завдання 1

Завдання

Реалізувати універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю. **30 балів.**

Діаграма структури

Реалізуємо систему із Create та 3-а Process елементами:



Код побудови системи

```
public class Task1
{
    public Model Model;
    private Create _create = new();
    private Process _process1 = new(5, 2, maxQueue: 5);
    private Process _process2 = new(7, 3, maxQueue: 5);
    private Process _process3 = new(10, 4, maxQueue: 5);

    public Task1()
    {
        var container = new NextElementsContainerByProbability();
        container.AddNextElement(_process1, 40);
```

```
        container.AddNextElement(_process2, 50);
        container.AddNextElement(_process3, 10);
        _create.NextElementsContainer = container;

        Model = new Model(new List<Element> { _create, _process1, _process2,
_process3 });
    }
}
```

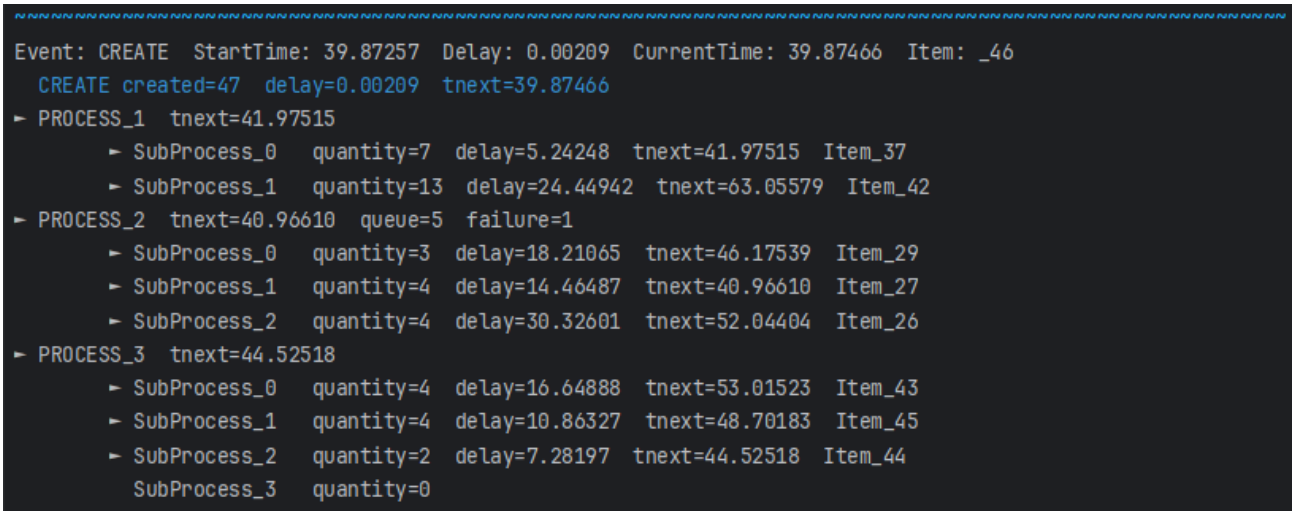
Вивід процесу

Запустимо симуляцію на час 100. Отже маємо наступний вивід:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: CREATE  StartTime: 0.00000  Delay: 0.52189  CurrentTime: 0.52189  Item: _0
  CREATE created=1  delay=0.52189  tnext=0.52189
  PROCESS_1  tnext=∞
          SubProcess_0   quantity=0
          SubProcess_1   quantity=0
► PROCESS_2  tnext=18.45625
        ► SubProcess_0   quantity=1  delay=18.45625  tnext=18.45625  Item_0
          SubProcess_1   quantity=0
          SubProcess_2   quantity=0
  PROCESS_3  tnext=∞
          SubProcess_0   quantity=0
          SubProcess_1   quantity=0
          SubProcess_2   quantity=0
          SubProcess_3   quantity=0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: CREATE  StartTime: 0.52189  Delay: 1.88841  CurrentTime: 2.41030  Item: _1
  CREATE created=2  delay=1.88841  tnext=2.41030
► PROCESS_1  tnext=1.17383
        ► SubProcess_0   quantity=1  delay=0.65194  tnext=1.17383  Item_1
          SubProcess_1   quantity=0
► PROCESS_2  tnext=18.45625
        ► SubProcess_0   quantity=1  delay=18.45625  tnext=18.45625  Item_0
          SubProcess_1   quantity=0
          SubProcess_2   quantity=0
  PROCESS_3  tnext=∞
          SubProcess_0   quantity=0
          SubProcess_1   quantity=0
          SubProcess_2   quantity=0
          SubProcess_3   quantity=0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: PROCESS_1  StartTime: 1.17383  Delay: 0.65194  CurrentTime: ∞  Item: _1
  CREATE created=2  delay=1.88841  tnext=2.41030
  PROCESS_1  tnext=∞
          SubProcess_0   quantity=1
          SubProcess_1   quantity=0
► PROCESS_2  tnext=18.45625
        ► SubProcess_0   quantity=1  delay=18.45625  tnext=18.45625  Item_0
          SubProcess_1   quantity=0
          SubProcess_2   quantity=0
  PROCESS_3  tnext=∞
          SubProcess_0   quantity=0
          SubProcess_1   quantity=0
          SubProcess_2   quantity=0
          SubProcess_3   quantity=0
```

У перших 2-х кроках створюються об’єкти. Item_0 надходить у Process_2, а Item_1 у Process_1. У 3-ому кроці Process_1 завершує обробку Item_1 у SubProcess_0.

Приклад отримання помилки:



Оскільки максимальний розмір черги 5, то при створенні об’єкту, який потрапив до Process_2, він не може потрапити у чергу (повністю зайнята).

Результат

CREATE: Quantity = 104

PROCESS_1: Quantity = 42 WorkTime = 0.895 Mean queue = 0.704 Failure prob. = 0 SubProcess_0: Quantity = 23 WorkTime = 75.605 SubProcess_1: Quantity = 18 WorkTime = 75.302	PROCESS_2: Quantity = 45 WorkTime = 1 Mean queue = 1.982 Failure prob. = 0.178 SubProcess_0: Quantity = 9 WorkTime = 93.325 SubProcess_1: Quantity = 12 WorkTime = 93.164 SubProcess_2: Quantity = 12 WorkTime = 81.19	PROCESS_3: Quantity = 17 WorkTime = 0.832 Mean queue = 0 Failure prob. = 0 SubProcess_0: Quantity = 5 WorkTime = 78.997 SubProcess_1: Quantity = 9 WorkTime = 48.767 SubProcess_2: Quantity = 3 WorkTime = 18.122 SubProcess_3: Quantity = 0 WorkTime = 0.00000

Як видно з результатів, перехід за ймовірністю працює коректно. Найбільш завантажений Process_2(50%), потім Process_1(40%) та Process_3(10%). Середня черга максимальна у найзавантаженішого Process_2, проте досягає лише 2 об’єктів. Помилки трапляються лише у ньому. Для покращення роботи моделі необхідно знизити завантаженість елемента, передавши його роботу найнезавантаженим. В нашому випадку таким елементом є Process_3, в якому SubProcess_2 та Subprocess_3 майже не працюють. Можна зменшити ймовірність переходу до Process_2 до 45% і збільшити у Process_3 до 20%, а Process_1 до 35%. Результати:

```
------------RESULTS------------
CREATE:
        Quantity = 109
PROCESS_1:
        Quantity = 37
        WorkTime = 0.9771692039144457
        Mean length of queue = 0.790220491633973
        Failure probability = 0
        SubProcess_0:
                Quantity = 14
                WorkTime = 96.12551
        SubProcess_1:
                Quantity = 23
                WorkTime = 82.17163
PROCESS_2:
        Quantity = 42
        WorkTime = 0.9590881827181037
        Mean length of queue = 1.6544410005897217
        Failure probability = 0.047619047619047616
        SubProcess_0:
                Quantity = 12
                WorkTime = 94.33738
        SubProcess_1:
                Quantity = 11
                WorkTime = 93.19911
        SubProcess_2:
                Quantity = 15
                WorkTime = 78.43802
```

```
PROCESS_3:
        Quantity = 30
        WorkTime = 0.9582481594436426
        Mean length of queue = 0.8576176286672911
        Failure probability = 0.03333333333333333
        SubProcess_0:
                Quantity = 10
                WorkTime = 93.59177
        SubProcess_1:
                Quantity = 7
                WorkTime = 74.44034
        SubProcess_2:
                Quantity = 8
                WorkTime = 56.49602
        SubProcess_3:
                Quantity = 4
                WorkTime = 43.35118
```

Як видно з результатів завантаженість кожного підпроцесу стала більш рівномірною. Помилки трапляються, оскільки процес створення нових об’єктів є надто частим, але тепер вони рівномірні вцілому.

Завдання 2

Завдання

Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму. **30 балів**:

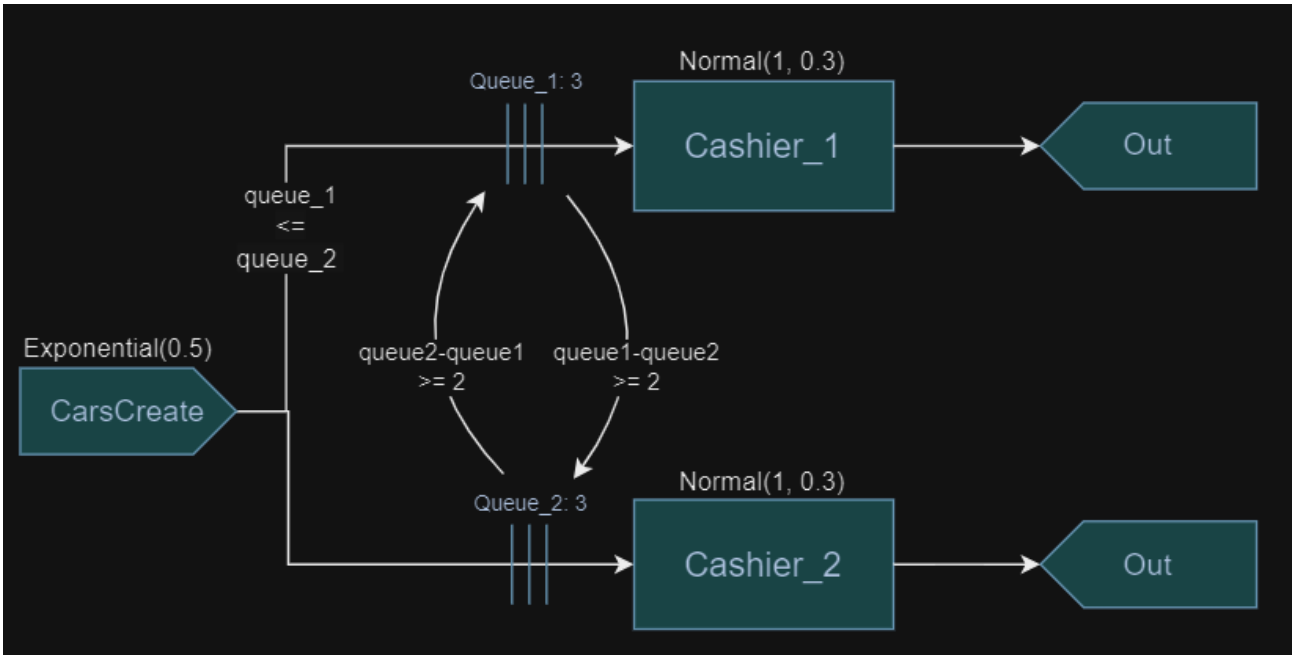
У банку для автомобілістів є два віконця, кожне з яких обслуговується одним касиром і має окрему під'їзну смугу. Обидві смуги розташовані поруч. З попередніх спостережень відомо, що інтервали часу між прибуттям клієнтів у годину пік розподілені експоненційно з математичним очікуванням, рівним 0,5 од. часу. Через те, що банк буває переобтяжений тільки в годину пік, то аналізується тільки цей період. Тривалість обслуговування в обох касирів однакова і розподілена експоненційно з математичним очікуванням, рівним 0,3 од. часу. Відомо також, що при рівній довжині черг, а також при відсутності черг, клієнти віддають перевагу першій смузі. В усіх інших випадках клієнти вибирають більш коротку чергу. Після того, як клієнт в'їхав у банк, він не може залишити його, доки не буде обслугований. Проте він може перемінити чергу, якщо стоїть останнім і різниця в довжині черг при цьому складає не менше двох автомобілів. Через обмежене місце на кожній смузі може знаходитися не більш трьох автомобілів. У банку, таким чином, не може знаходитися більш восьми автомобілів, включаючи автомобілі двох клієнтів, що обслуговуються в поточний момент касиром. Якщо місце перед банком заповнено до границі, то клієнт, що прибув, вважається втраченим, тому що він відразу ж виїжджає.

Початкові умови такі: 1) обидва касири зайняті, тривалість обслуговування для кожного касира нормально розподілена з математичним очікуванням, рівним 1 од. часу, і середньоквадратичним відхиленням, рівним 0,3 од. часу; 2) прибуття першого клієнта заплановано на момент часу 0,1 од. часу; 3) у кожній черзі очікують по два автомобіля.

Визначити величини: 1) середнє завантаження кожного касира; 2) середнє число клієнтів у банку; 3) середній інтервал часу між від'їздами клієнтів від

вікон; 4) середній час перебування клієнта в банку; 5) середнє число клієнтів у кожній черзі; 6) відсоток клієнтів, яким відмовлено в обслуговуванні; 7) число змін під'їзних смуг.

Діаграма структури



Додаткові умови та їх реалізація

Маємо реалізувати:

1) NextElementContainerByQueuePriority для обрання касиру клієнтами за кількістю машин у черзі:

```
public class NextElementsContainerByQueuePriority : NextElementsContainer
{
   private Dictionary<Process, int> _nextElements = new();
   public void AddNextElement(Process process, int ascendingPriority) =>
_nextElements.Add(process, ascendingPriority);

   protected override Element GetNextElement()
   {
       var elementsWithMinQueueLength = GetElementsWithMinQueueLength();
       return elementsWithMinQueueLength.MinBy(e => e.Value).Key;
   }

   private Dictionary<Process, int> GetElementsWithMinQueueLength()
   {
       var minQueueLength = _nextElements.Min(e => e.Key.QueueLength);
       var nextElementsWithMinQueueLength = new Dictionary<Process, int>();
       foreach (var (process, priority) in _nextElements)
```

```
            if (process.QueueLength == minQueueLength)
                nextElementsWithMinQueueLength.Add(process, priority);
        return nextElementsWithMinQueueLength;
    }
}
```

2) Event OnQueueChange та метод, що підписується ChangeQueueIfNecessary для зміни черги автомобіля якщо сусідня черга менша хоча б на 2 автомобілі:

```
private void ChangeQueueIfNecessary()
{
    if (Math.Abs(_cashier1.QueueLength -_cashier2.QueueLength) < 2) return;
    Console.WriteLine($"\nCHANGE_QUEUE: Q1={_cashier1.QueueLength}
Q2={_cashier2.QueueLength}!*!*!*!*!");
    if (_cashier1.QueueLength > _cashier2.QueueLength) {
        Process.TryChangeQueueForLastItem(_cashier1, _cashier2);
        Console.WriteLine("CASHIER1 -> CASHIER2");
    }
    else {
        Process.TryChangeQueueForLastItem(_cashier2, _cashier1);
        Console.WriteLine("CASHIER2 -> CASHIER1");
    }
    Model.QueueChangeCount7++;
}
```

3) Початкові умови:

```
const double startTime = 0.1;
_cars.NextT = startTime;
for (var i = 0; i < 3; i++)
{
    _cashier1.InAct(new Item(startTime));
    _cashier2.InAct(new Item(startTime));
}
```

4) Збір статистики:

```
public class Task2Model : Model
{
    public int QueueChangeCount7;
    public Task2Model(List<Element> elements) : base(elements) {}

    public override void Simulate(double time, double startTime = 0, bool
printSteps = false)
    {
        base.Simulate(time, startTime, printSteps);
        Console.WriteLine();
        Console.WriteLine($"1) Average workTime:"); PrintEachProcessWorkTime_1();
        Console.WriteLine($"2) Average clients count: {AverageItemsCount}");
        Console.WriteLine($"3) Average time between departures:
{AverageBetweenOutActs3}");
```

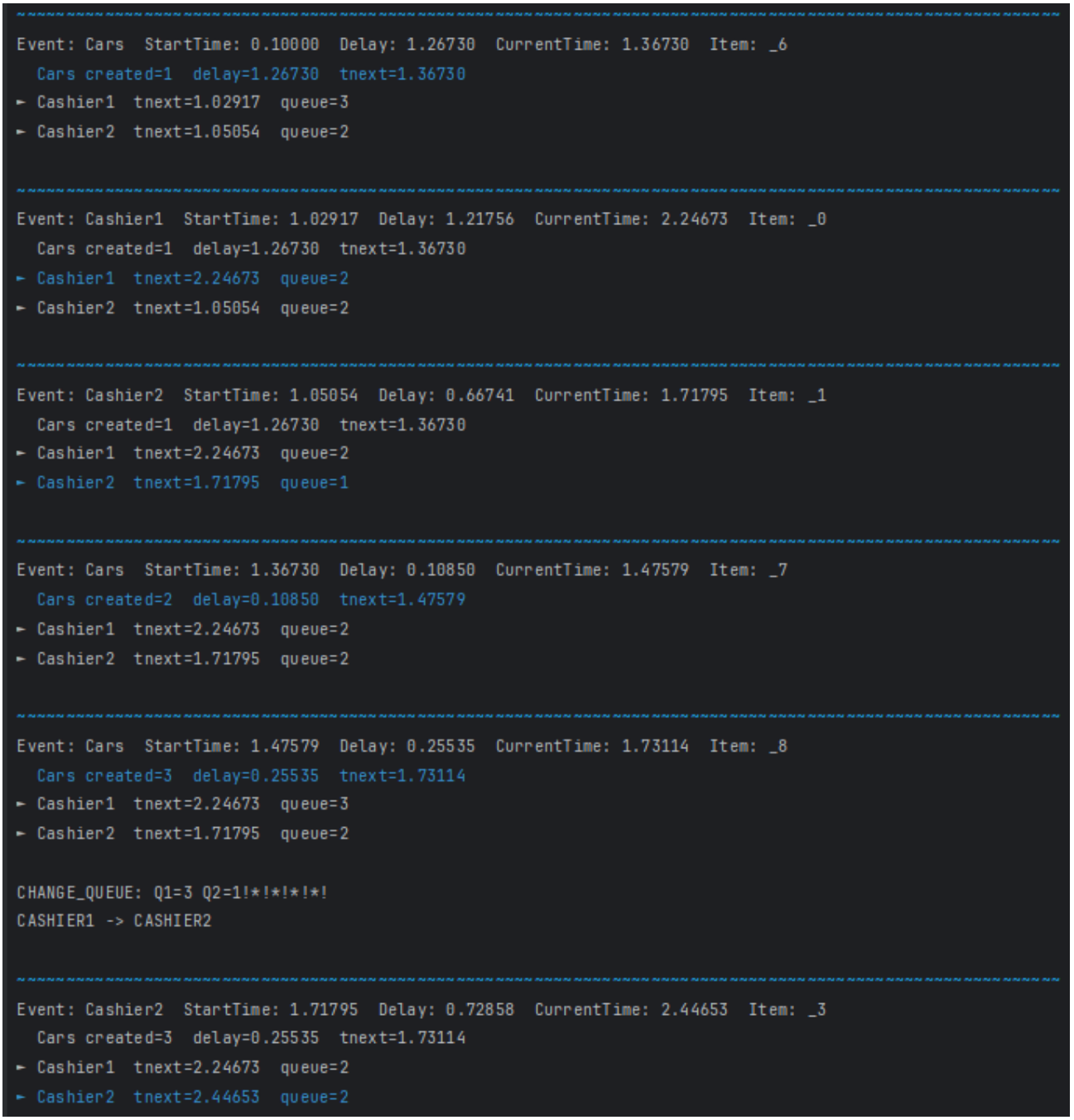
```
        Console.WriteLine($"4) Average time spent by a customer in the bank:
{AverageItemTimeInSystem}");
        Console.WriteLine($"5) Average clients count in each queue:");
PrintEachProcessMeanQueue_5();
        Console.WriteLine($"6) Failure probability: {FailurePercent}%");
        Console.WriteLine($"7) Queue change count: {QueueChangeCount7}");
    }

    private void PrintEachProcessWorkTime_1() => Processes.ForEach(p =>
Console.WriteLine($"\t{p.Name}: {p.WorkTime / AllTime}"));
    private double AverageBetweenOutActs3 => Processes.Sum(p =>
p.TotalTimeBetweenOutActs / p.OutActQuantity) / Processes.Count;
    private void PrintEachProcessMeanQueue_5() => Processes.ForEach(p =>
Console.WriteLine($"\t{p.Name}: {p.MeanQueueAllTime / AllTime}"));
}
```

Код побудови системи

```
public class Task2CarBank
{
    public Task2Model Model;
    private Create _cars = new(0.5, name: "Cars");
    private Process _cashier1 = new(new NormalRandomizer(1, 0.3), maxQueue: 3,
name: "Cashier1");
    private Process _cashier2 = new(new NormalRandomizer(1, 0.3), maxQueue: 3,
name: "Cashier2");

    public Task2CarBank() {
          // Connecting elements
          var cashiers = new NextElementsContainerByQueuePriority();
          cashiers.AddNextElement(_cashier1, 1);
          cashiers.AddNextElement(_cashier2, 2);
          _cars.NextElementsContainer = cashiers;

        // Changing queues condition
        _cashier1.OnQueueChanged += ChangeQueueIfNecessary;
        _cashier2.OnQueueChanged += ChangeQueueIfNecessary;

        // Initial states condition
        const double startTime = 0.1;
        _cars.NextT = startTime;
        for (var i = 0; i < 3; i++)
        {
        _cashier1.InAct(new Item(startTime));
        _cashier2.InAct(new Item(startTime));
        }

        Model = new Task2Model(new List<Element> { _cars, _cashier1, _cashier2 });
}
```

Вивід процесу

Запустимо симуляцію на час 10, щоб продивитися кроки виконання моделі:



Кроки:

1) Через початкові умови маємо зайнятих касирів та черги по дві машини у кожній. Приїжджає перший клієнт у час 0.1, який стає у чергу до першого касира, оскільки черга до 1-го касира при рівних чергах є пріоритетною.
2) Перший касир опрацював клієнта, перейшов до наступного, черга стала 2.
3) Пункт 2 тільки для другого касира, черга стала 1
4) Приїхав клієнт. Оскільки черга до 2-го касира менша займає її.

5) Приїжджає машина, стає у чергу №1. Черги стають 3 та 2.
6) Касир №2 обробив клієнта, взяв наступного. Черги стали 3 та 1. Останній автомобіль черги №1 переїхав до черги №2. Черги стали 2, 2.

Результат

Виконаємо симуляцію на час 300:

```
-------------RESULTS-------------
Cars:
        Quantity = 611
Cashier1:
        WorkTime = 0.9724040811655466
        InActQuantity = 399
        OutActQuantity = 302
        Current queue length = 2
        Mean length of queue = 1.419124406333199
        Failure probability = 0.11278195488721804
        SubProcess_0:
                Quantity = 303
                WorkTime = 0.97240
Cashier2:
        WorkTime = 0.8662566780411117
        InActQuantity = 218
        OutActQuantity = 265
        Current queue length = 1
        Mean length of queue = 1.181107120699592
        Failure probability = 0
        SubProcess_0:
                Quantity = 266
                WorkTime = 0.86626
```

Отже:

1) Під’їхало до банку 611 автомобілів, касири загалом обробили 567 та обробляють ще двох. Отже, 42 клієнта не заїхали на обслуговування;
2) Середня черга майже однакова, через переїзди автомобілів, і сягає 2-х;
3) Перший касир працював майже увесь час, і обробив більше ніж другий;
4) В даній системі Failure probability у другого касира завжди буде 0. Оскільки при чергах 3 та 3, наступний клієнт обере 1-го касира. Побачивши, що черга зайнята, він поїде з території банку.

Статистика:

```
1) Average workTime:
        Cashier1: 0.9724040811655466
        Cashier2: 0.8662566780411117
2) Average clients count: 4.438892286239451
3) Average time between departures: 1.058875971783115
4) Average time spent by a customer in the bank: 1.1252140140229507
5) Average clients count in each queue:
        Cashier1: 1.419124406333199
        Cashier2: 1.181107120699592
6) Failure probability: 7.3649754500818325%
7) Queue change count: 51
```

Завдання 3

Завдання

Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму (**40 балів**):

У лікарню поступають хворі таких трьох типів:

1) хворі, що пройшли попереднє обстеження і направлені на лікування;

2) хворі, що бажають потрапити в лікарню, але не пройшли повністю попереднє обстеження;

3) хворі, які тільки що поступили на попереднє обстеження.

Чисельні характеристики типів хворих наведені в таблиці:

Тип хворого	Відносна частота	Середній час реєстрації, хв
1	0,5	15
2	0,1	40
3	0,4	30

При надходженні в приймальне відділення хворий стає в чергу, якщо обидва чергових лікарі зайняті. Лікар, який звільнився, вибирає в першу чергу хворого типу 1.

Після заповнення різноманітних форм у приймальне відділення хворі 1 типу ідуть прямо в палату, а хворі типів 2 і 3 направляються в лабораторію. Троє супровідних розводять хворих по палатах. Хворим не дозволяється направлятися в палату без супровідного. Якщо всі супровідні зайняті, хворі очікують їхнього звільнення в приймальному відділенні. Як тільки хворий доставлений у палату, він вважається таким, що завершив процес прийому до лікарні. Хворі, що спрямовуються в лабораторію, не потребують супроводу. Після прибуття в лабораторію хворі стають у чергу в реєстратуру. Після реєстрації вони ідуть у кімнату очікування, де чекають виклику до одного з двох

лаборантів. Після здачі аналізів хворі або повертаються в приймальне відділення (хворий типу 2), або залишають лікарню (хворий типу 1). Після повернення в приймальне відділення хворий, що здав аналізи, розглядається як хворий типу 1.

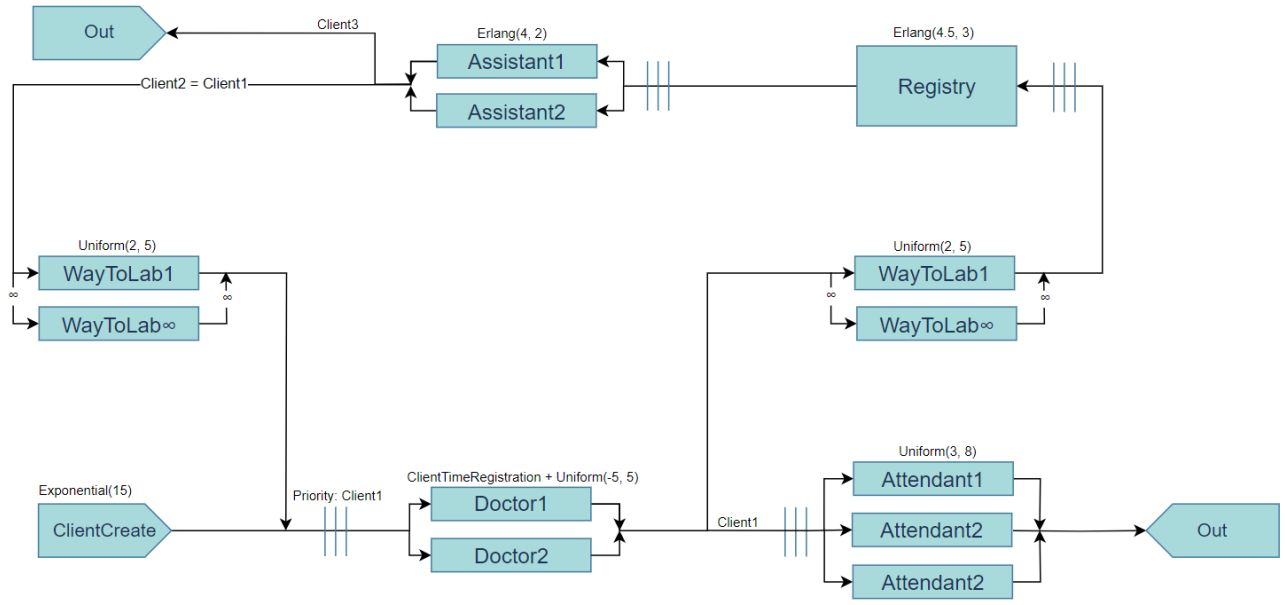
У наступній таблиці приводяться дані по тривалості дій (хв):

Величина	Розподіл
Час між прибуттями в приймальне відділення	Експоненціальний з математичним сподіванням 15
Час слідування в палату	Рівномірне від 3 до 8
Час слідування з приймального відділення в лабораторію і навпаки	Рівномірне від 2 до 5
Час обслуговування в реєстратуру лабораторії	Ерланга з математичним сподіванням 4,5 і k=3
Час проведення аналізу в лабораторії	Ерланга з математичним сподіванням 4 і k=2

Визначити час, проведений хворим у системі, тобто інтервал часу, починаючи з надходження і закінчуючи доставкою в палату (для хворих типу 1 і 2) або виходом із лабораторії (для хворих типу 3).

Визначити також інтервал між прибуттями хворих у лабораторію.

Діаграма структури



Додаткові умови та їх реалізація

Маємо реалізувати:

1) Типи хворих:

```
public class Client : Item
{
    public ClientType ClientType { get; set; }
    public double RegistrationTime { get; set; }

    public Client(string name, ClientType clientType, double registrationTime,
double startTime) : base(startTime)
    {
        Name = name;
        ClientType = clientType;
        RegistrationTime = registrationTime;
    }
}

public enum ClientType {Chamber, NotExamined, Lab}
```

2) Створення хворих із різною ймовірністю:

```
public class CreateClient : Create
{
    public CreateClient(Randomizer rand,string name) : base(rand,name) {}

    protected override Item CreateItem()
    {
        var number = new Random().NextDouble();
        if (number <= 0.5) return new Client("ChamberClient", ClientType.Chamber,
15, CurrT);
        if (number <= 0.6)
```

```
            return new Client("NotExaminedChamberClient",ClientType.NotExamined,
        40, CurrT);
        return new Client("LabClient", ClientType.Lab, 30, CurrT);
    }
}
```

3) Пріоритетну чергу до доктора для хворого типу 1:

```
internal class ClientsQueue : ItemsQueue
{
    public ClientsQueue(int limit) : base(limit) {}
    public override Item GetItem()
    {
        var chamberClient = Queue.Find(x => (x as Client).ClientType ==
ClientType.Chamber);
        if (chamberClient != null)
        {
            Queue.Remove(chamberClient);
            return chamberClient;
        }
        return base.GetItem();
    }
}
```

4) Перехід від доктора до наступних елементів в залежності від типу хворого:

```
public class NextAfterDoctor : NextElementsContainer
{
    private Process NextAttendantProcess { get; }
    private Process NextWayToLab { get; }

    private DoctorProcess _doctorProcess;

    public NextAfterDoctor(DoctorProcess doctorProcess, Process
nextAttendantProcess, Process nextWayToLab)
    {
        _doctorProcess = doctorProcess;
        NextAttendantProcess = nextAttendantProcess;
        NextWayToLab = nextWayToLab;
    }

    protected override Element GetNextElement()
    {
        var client = _doctorProcess.Item as Client;
        if (client.ClientType == ClientType.Chamber)
            return NextAttendantProcess;
        return NextWayToLab;
    }
}
```

5) Клас DoctorProcess:

```
public class DoctorProcess : Process
{
    public DoctorProcess(int doctorsCount, string name, string subProcessName, int
maxQueue = 2147483647)
```

```
        : base(new UniformRandomizer(-5, 5), doctorsCount, name, maxQueue,
subProcessName)=> Queue = new ClientsQueue(maxQueue);

    protected override double GetDelay()=> (Item as Client).RegistrationTime +
Randomizer.GenerateDelay();
}
```

6) Генерація розподілу Ерланга:

```
public class ErlangRandomizer : Randomizer
{
    private double TimeMean { get; }
    private int k { get; }

    public ErlangRandomizer(double timeMean, int k)
    {
        TimeMean = timeMean;
        this.k = k;
    }

    public override double GenerateDelay() => Enumerable.Range(0, k).Select(_ =>
-TimeMean * Math.Log(_random.NextDouble())).Sum();
}
```

7) Зміна типу клієнта після здачі аналізів:

```
public class LabAssistanceProcess : Process
{
    public LabAssistanceProcess(Randomizer randomizer, int assistanceCount, string
name, string subProcessName, int maxQueue = 2147483647) :
        base(randomizer, assistanceCount, name, maxQueue, subProcessName) {}

    protected override void NextElementsContainerSetup()
    {
        if (Item is Client { ClientType: ClientType.NotExamined } client)
        {
            client.ClientType = ClientType.Chamber;
            client.RegistrationTime = 15;
            client.Name = "ChamberClient";
        }
        else NextElementsContainer = null;
    }
}
```

8) Збір статистики:

```
public class Task3HospitalModel : Model
{
    public Task3HospitalModel(List<Element> elements, bool initialStateIsNeeded =
false) : base(elements, initialStateIsNeeded) {}

    public override void Simulate(double time, double startTime = 0, bool
printSteps = false)
    {
        base.Simulate(time, startTime, printSteps);
        Console.WriteLine();
```

```
        Console.WriteLine($"1) Average spent time in system:
{AverageItemTimeInSystem}");
        Console.WriteLine($"2) Average time between arrivals in registry:
{AverageBetweenInActs_2()}");
    }

    private double AverageBetweenInActs_2()
    {
        var lab = Elements.OfType<Process>().First(p => p.Name == "Registry");
        return lab.TotalTimeBetweenInActs / lab.InActQuantity;
    }
}
```

Код побудови системи

```
public class Task3Hospital
{
    public Model Model { get; }

    public Task3Hospital()
    {
        CreateClient patients = new CreateClient(new ExponentialRandomizer(15),
"Patient");
        DoctorProcess doctors = new(2, "Doctors", "Doctor");
        Process attendants = new(new UniformRandomizer(3, 8), subProcessCount: 3,
name: "Attendants", subProcessName: "Attendant");
        Process fromHospitalToLab = new(new UniformRandomizer(2, 5), 25, name:
"WayToLab");
        Process labRegistry = new(new ErlangRandomizer(4.5, 3), name: "Registry");
        LabAssistanceProcess labAssistants = new(new ErlangRandomizer(4, 2),
assistanceCount: 2, "Assistants", subProcessName: "Assistant");
        Process fromLabToHospital = new(new UniformRandomizer(2, 5), 25, name:
"WayToHospital");

        patients.NextElementsContainer = new NextElementContainer(doctors);
        doctors.NextElementsContainer = new NextAfterDoctor(doctors, attendants,
fromHospitalToLab);
        fromHospitalToLab.NextElementsContainer = new
NextElementContainer(labRegistry);
        labRegistry.NextElementsContainer = new
NextElementContainer(labAssistants);
        labAssistants.NextElementsContainer = new
NextElementContainer(fromLabToHospital);
        fromLabToHospital.NextElementsContainer = new
NextElementContainer(doctors);

        Model = new Task3HospitalModel(new List<Element>() { patients, doctors,
attendants, fromHospitalToLab, labRegistry, labAssistants, fromLabToHospital });
    }
}
```

Вивід процесу

Продемонструємо роботу системи для хворих типу 2 та 3 (тип 2 стає типом 1, таким чином можемо побачити поведінку і для нього). Тип 3:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: Patient  StartTime: 0.00000  Delay: 34.79083  CurrentTime: 34.79083  Item: LabClient_0
  Patient created=1  delay=34.79083  tnext=34.79083
► Doctors  tnext=26.19709
        ► Doctor_0      quantity=1  delay=26.19709  tnext=26.19709  LabClient_0
          Doctor_1      quantity=0
  Attendants  tnext=∞
          Attendant_0   quantity=0
          Attendant_1   quantity=0
          Attendant_2   quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0   quantity=0
          Assistant_1   quantity=0
  WayToHospital  tnext=∞
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: Doctors  StartTime: 26.19709  Delay: 0.00000  CurrentTime: ∞  Item: LabClient_0
  Patient created=1  delay=34.79083  tnext=34.79083
  Doctors  tnext=∞
          Doctor_0      quantity=1
          Doctor_1      quantity=0
  Attendants  tnext=∞
          Attendant_0   quantity=0
          Attendant_1   quantity=0
          Attendant_2   quantity=0
► WayToLab  tnext=30.36620  workingSubProcesses count = 1

  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0   quantity=0
          Assistant_1   quantity=0
  WayToHospital  tnext=∞
```

LabClient приходить і потрапляє до доктора №0. Після доктора йде до лабораторії.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: WayToLab  StartTime: 30.36620  Delay: 4.16910  CurrentTime: ∞  Item: LabClient_0
  Patient created=1  delay=34.79083  tnext=34.79083
  Doctors  tnext=∞
          Doctor_0      quantity=1
          Doctor_1      quantity=0
  Attendants  tnext=∞
          Attendant_0   quantity=0
          Attendant_1   quantity=0
          Attendant_2   quantity=0
  WayToLab  tnext=∞
► Registry  tnext=45.32957
  Assistants  tnext=∞
          Assistant_0   quantity=0
          Assistant_1   quantity=0
  WayToHospital  tnext=∞
```

Пацієнта реєструють.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: Registry  StartTime: 45.32957  Delay: 14.96337  CurrentTime: ∞  Item: LabClient_0
  Patient created=2  delay=32.49014  tnext=67.28097
► Doctors  tnext=68.25964
        ► Doctor_0       quantity=2  delay=33.46881  tnext=68.25964  LabClient_1
          Doctor_1       quantity=0
  Attendants  tnext=∞
          Attendant_0    quantity=0
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
► Assistants  tnext=67.51835
        ► Assistant_0    quantity=1  delay=22.18877  tnext=67.51835  LabClient_0
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Пацієнта обстежує асистент №0. За цей час до доктора №0 надійшов наступний пацієнт.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: Assistants  StartTime: 67.51835  Delay: 22.18877  CurrentTime: ∞  Item: LabClient_0
  Patient created=4  delay=0.94616  tnext=68.43939
► Doctors  tnext=68.25964  queue=1
        ► Doctor_0       quantity=2  delay=33.46881  tnext=68.25964  LabClient_1
        ► Doctor_1       quantity=1  delay=31.70600  tnext=98.98696  LabClient_2
  Attendants  tnext=∞
          Attendant_0    quantity=0
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=1
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Асістент закінчив обстеження LabClient_0. Хворий покинув систему. За час обстеження доктор №1 прийняв хворого, і утворилася черга розміром 1 у приймальному відділенні.

Тепер приклад поведінки пацієнта №2 у системі:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: Patient  StartTime: 0.00000  Delay: 15.53273  CurrentTime: 15.53273  Item: NotExaminedChamberClient_0
  Patient created=1  delay=15.53273  tnext=15.53273
► Doctors  tnext=38.37797
        ► Doctor_0       quantity=1  delay=38.37797  tnext=38.37797  NotExaminedChamberClient_0
          Doctor_1       quantity=0
  Attendants  tnext=∞
          Attendant_0    quantity=0
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=0
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Пацієнт №2 (NotExaminedChamberClient) прийшов у приймальне відділення і його прийняв доктор №0.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Event: Doctors  StartTime: 38.37797  Delay: 0.00000  CurrentTime: 53.78247  Item: NotExaminedChamberClient_0
  Patient created=3  delay=32.78054  tnext=68.90681
► Doctors  tnext=53.78247
          Doctor_0       quantity=1
        ► Doctor_1       quantity=2  delay=17.65620  tnext=53.78247  ChamberClient_2
  Attendants  tnext=∞
          Attendant_0    quantity=1
          Attendant_1    quantity=0
          Attendant_2    quantity=0
► WayToLab  tnext=41.92645  workingSubProcesses count = 1

  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=0
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Після доктору наш пацієнт пішов у лабораторію. За час спілкування із лікарем надійшло 2 пацієнти типу 1, 1 з яких був проведений у палату після спілкування із доктором. Пропустимо ідентичні етапи перевірки у лабораторії із хворим типу 3.

```
Event: Assistants  StartTime: 59.32026  Delay: 5.51822  CurrentTime: ∞  Item: ChamberClient_0
  Patient created=3  delay=32.78054  tnext=68.90681
  Doctors  tnext=∞
          Doctor_0       quantity=1
          Doctor_1       quantity=2
► Attendants  tnext=60.26845
        ► Attendant_0    quantity=2  delay=6.48598  tnext=60.26845  ChamberClient_2
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=1
          Assistant_1    quantity=0
► WayToHospital  tnext=61.72353  workingSubProcesses count = 1
```

Ассистент закінчив роботу із нашим пацієнтом і він направився до приймального відділення.

```
Event: WayToHospital  StartTime: 61.72353  Delay: 2.40327  CurrentTime: ∞  Item: ChamberClient_0
  Patient created=3  delay=32.78054  tnext=68.90681
► Doctors  tnext=81.07920
        ► Doctor_0       quantity=2  delay=19.35567  tnext=81.07920  ChamberClient_0
          Doctor_1       quantity=2
  Attendants  tnext=∞
          Attendant_0    quantity=2
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=1
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Хворий прийшов до лікарні і його прийняв лікар як хворого типу 1.

```
Event: Doctors  StartTime: 81.07920  Delay: 0.00000  CurrentTime: 98.06680  Item: ChamberClient_0
  Patient created=4  delay=18.38866  tnext=87.29548
► Doctors  tnext=98.06680
          Doctor_0       quantity=2
        ► Doctor_1       quantity=3  delay=29.15999  tnext=98.06680  LabClient_3
► Attendants  tnext=84.68302
        ► Attendant_0    quantity=3  delay=3.60382  tnext=84.68302  ChamberClient_0
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=1
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Після спілкування із доктором, хворого забирає супровідний і веде до палати.

```
Event: Attendants  StartTime: 84.68302  Delay: 3.60382  CurrentTime: ∞  Item: ChamberClient_0
  Patient created=4  delay=18.38866  tnext=87.29548
► Doctors  tnext=98.06680
          Doctor_0       quantity=2
        ► Doctor_1       quantity=3  delay=29.15999  tnext=98.06680  LabClient_3
  Attendants  tnext=∞
          Attendant_0    quantity=3
          Attendant_1    quantity=0
          Attendant_2    quantity=0
  WayToLab  tnext=∞
  Registry  tnext=∞
  Assistants  tnext=∞
          Assistant_0    quantity=1
          Assistant_1    quantity=0
  WayToHospital  tnext=∞
```

Його провели і він є 3-ім пацієнтом, що надійшов до палати.

Результат

Проведемо симуляцію на час 1000 хвилин. Статистика працівників:

```
-------------RESULTS-------------
Patient:
        Quantity = 64
Doctors:
        WorkTime = 0.9358399052172152
        InActQuantity = 72
        OutActQuantity = 70
        Current queue length = 0
        Mean length of queue = 0.8624
        Failure probability = 0
        Doctor_0:
                Quantity = 38
                WorkTime = 0.81700
        Doctor_1:
                Quantity = 34
                WorkTime = 0.81343
Attendants:
        WorkTime = 0.195636615875831
        InActQuantity = 37
        OutActQuantity = 37
        Current queue length = 0
        Mean length of queue = 0
        Failure probability = 0
        Attendant_0:
                Quantity = 35
                WorkTime = 0.18935
        Attendant_1:
                Quantity = 2
                WorkTime = 0.01039
        Attendant_2:
                Quantity = 0
                WorkTime = 0.00000
```

```
Registry:
        WorkTime = 0.45979638999663475
        InActQuantity = 33
        OutActQuantity = 33
        Current queue length = 0
        Mean length of queue = 0.1433823128874475
        Failure probability = 0
        SubProcess_0:
                Quantity = 33
                WorkTime = 0.45980
```

```
Assistants:
        WorkTime = 0.26216990764026255
        InActQuantity = 33
        OutActQuantity = 33
        Current queue length = 0
        Mean length of queue = 0
        Failure probability = 0
        Assistant_0:
                Quantity = 27
                WorkTime = 0.21825
        Assistant_1:
                Quantity = 6
                WorkTime = 0.06523
```

Статистика переходів:

```
WayToLab:
        WorkTime = 0.11393195730301735
        InActQuantity = 33
```

```
WayToHospital:
        WorkTime = 0.025556158167290555
        InActQuantity = 8
```

Додаткова статистика:

1) Average spent time in system: 43.99584379435704
2) Average time between arrivals in registry: 27.322393179796315

Висновки:

1) Доктори працюють майже увесь час і мають чергу в 1 людину в середньому. Оскільки загальна кількість клієнтів, що прийшла до лікарні 64, а доктори виконали 72 сеанси, то можна стверджувати, що 8 хворих прийшли повторно (тип 2).
2) Супровідний №0 провів майже усіх хворих, черги немає. Для економії грошей лікарня може відмовитись від вакансій Супровідний №1 та №2.
3) Реєстратура працює добре, завантаженість 46%, іноді формуються черги до 2 особ. Для зниження часу хворого у системі можна додати додаткового працівника у реєстратуру.

4) Ассистенти працюють добре, черг немає. Формується ситуація, що ассистент №0 не справиться сам. Але сильного навантаження на №1 немає. На місце ассистента №1 можна брати інтернів.
5) З переходів видно, що на частку тих, хто прийшов до лабораторії, 75% просто здають аналізи. Можливо реалізувати пріоритет у черзі для хворих, що будуть лягати у лікарню на етапі реєстрації. Це має зменшити час хворого у системі. Або виконати пункт 3.

Результати нової моделі

Спробуємо запустити нову модель без супровідних №1 та №2, та додамо нового працівника у реєстатуру:

```
------------RESULTS------------
Patient:
        Quantity = 61
Doctors:
        WorkTime = 0.8691965778317153
        InActQuantity = 66
        OutActQuantity = 64
        Current queue length = 0
        Mean length of queue = 0.33342023410721977
        Failure probability = 0
        Doctor_0:
                Quantity = 38
                WorkTime = 0.82530
        Doctor_1:
                Quantity = 28
                WorkTime = 0.60428
Attendants:
        WorkTime = 0.20363860101271392
        InActQuantity = 37
        OutActQuantity = 36
        Current queue length = 0
        Mean length of queue = 0.008835705293438419
        Failure probability = 0
Registry:
        WorkTime = 0.3474335754139706
        InActQuantity = 27
        OutActQuantity = 25
        Current queue length = 0
        Mean length of queue = 0
        Failure probability = 0
        SubProcess_0:
                Quantity = 22
                WorkTime = 0.30868
        SubProcess_1:
                Quantity = 5
                WorkTime = 0.06947
Assistants:
        WorkTime = 0.16818150190190348
        InActQuantity = 25
        OutActQuantity = 25
        Current queue length = 0
        Mean length of queue = 0
        Failure probability = 0
        Assistant_0:
                Quantity = 22
                WorkTime = 0.14766
        Assistant_1:
                Quantity = 3
                WorkTime = 0.02407
```

1) Average spent time in system: 32.53513388382148
2) Average time between arrivals in registry: 35.49048691635963

Як видно з результатів, середній час пацієнта у системи знизився з 44 хвилин до 33. Супровідник справляється сам. Завантаженість працівників реєстратури подібна до ассистентської. Різниця полягає лише у часі процесів. В реєстратурі час роботи завеликий. Отже, по можливості, для налагодження системи необхідно скоротити час самої реєстрації. Це призведе до більш швидкої здачі аналізів, а це може призвести до збільшення черги на прийом до лікаря. В такому разі необхідно буде додати нову вакансію на посаду лікар №3.