Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Комп’ютерний практикум №2

Моделювання систем

Виконав студент групи ІП-15: Плугатирьов Д.В.		Перевірив: Стеценко І.В.
		Дата:
		Оцінка:

Київ 2025

Завдання

1. Реалізувати алгоритм імітації простої моделі обслуговування одним пристроєм з використанням об'єктно-орієнтованого підходу. **5 балів.**
2. Модифікувати алгоритм, додавши обчислення середнього завантаження пристрою. **5 балів.**
3. Створити модель за схемою, представленою на рисунку 1. **30 балів.**
4. Виконати верифікацію моделі, змінюючи значення вхідних змінних та параметрів моделі. Навести результати верифікації у таблиці. **10 балів.**

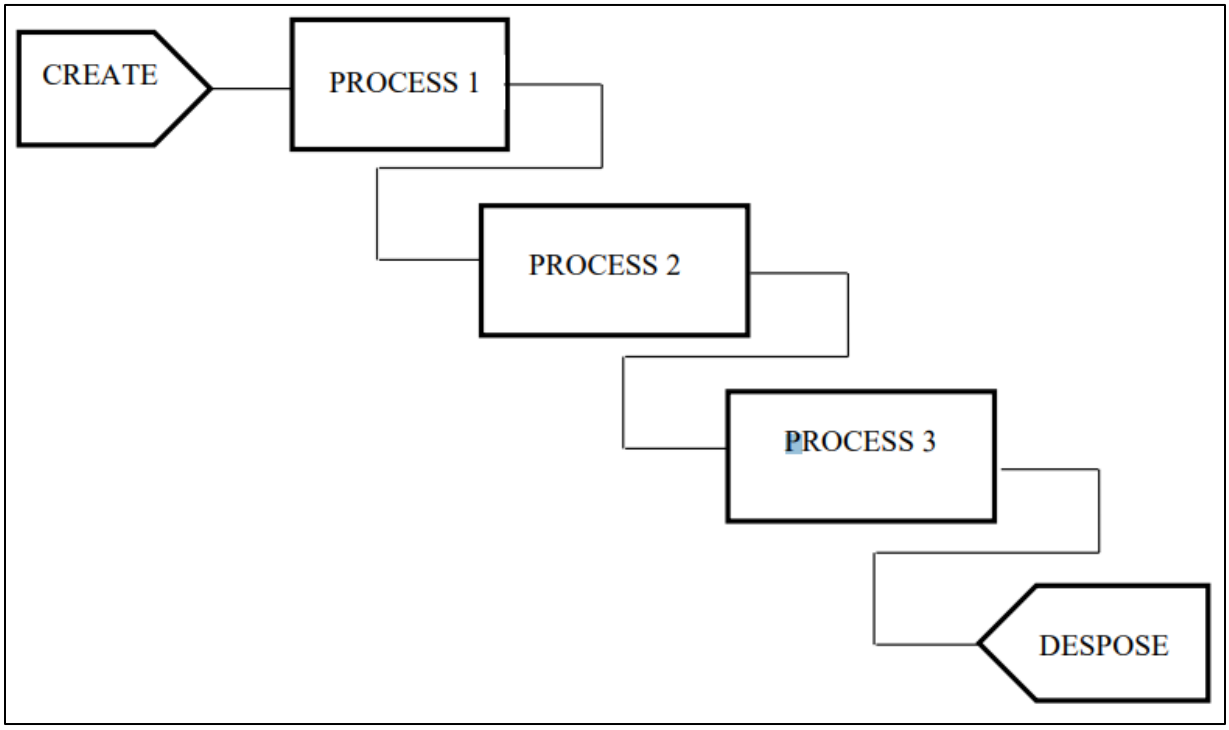

Рисунок 1 – Схема моделі

5. Модифікувати клас PROCESS, щоб можна було його використовувати для моделювання процесу обслуговування кількома ідентичними пристроями. **20 балів.**
6. Модифікувати клас PROCESS, щоб можна було організовувати вихід в два і більше наступних блоків, в тому числі з поверненням у

попередні блоки. **30 балів.**

Структура коду

Програма реалізована у шести файлах. Файл **utils.py** містить допоміжні функції та константи (зокрема, форматування часу, **INF_TIME**, **TIME_EPS**, а також декоратор для підрахунку створених екземплярів).

У файлі **base_element.py** оголошено базовий клас **Element**, який відповідає за основну логіку обробки подій: зберігає час поточної події, генерує час наступної, передає подію далі за заданими імовірностями переходу.

Клас **CreateElement**, що знаходиться у файлі **creator.py**, відповідає за генерацію заявок (подій) з обраним розподілом інтервалів (експоненційним у нашому прикладі). Його метод **end_action()** збільшує лічильник створених заявок та визначає час появи наступної події.

Файл **process.py** містить клас **ProcessElement**. Він реалізує чергу (з обмеженням на максимальний розмір) і підтримує роботу з декількома ідентичними "обробниками" (handlers). У випадку, коли всі обробники зайняті, заявка або потрапляє у чергу, або "втрачається" (збільшується кількість відмов), якщо черга переповнена. Після завершення обслуговування подія може бути скерована в один чи кілька наступних блоків із заданими імовірностями (у прикладі частина заявок повертається з третього процесу назад у перший).

Логікою запуску та "прогону" (simulation run) усієї мережі елементів керує клас **Model** із файлу **simulation_model.py**. Він відстежує глобальний час, знаходить найраніший момент завершення події серед усіх елементів, викликає для них **end_action()** і так далі, аж поки не буде досягнуто кінцевий час моделювання.

Нарешті, у файлі **simulation.py** створюються та з'єднуються екземпляри

CreateElement і кількох **ProcessElement**, встановлюються початкові параметри (наприклад, кількість обслуговуючих пристроїв, інтенсивність надходження заявок тощо), після чого запускається симуляція та формується підсумкова статистика.

Алгоритм роботи програми

На початку **CreateElement** з певним середнім інтервалом часу генерує нову заявку та негайно передає її на вхід першому **ProcessElement**. Процесори (**ProcessElement**) приймають заявку в чергу, якщо всі пристрої вже зайняті; якщо у черзі є вільне місце, заявка чекатиме, інакше система "відмовляє". Коли обслуговування завершується, метод **end_action()** передає заявку далі за визначеними ймовірностями: у прикладі частина заявок може повертатися назад у попередній процес. Клас **Model** синхронізує події: знаходить найближчий час завершення обслуговування серед усіх процесорів та генерує виклик їхніх **end_action()**. По закінченні заданого періоду моделювання збирається статистика: кількість створених або оброблених заявок, середній розмір черги, середній час очікування, середнє завантаження обслуговуючих пристроїв і ймовірність відмови.

Підсумкові статистичні дані моделювання

На рисунку нижче наведено приклад підсумкового виводу для проміжку часу 1000 умовних одиниць, коли перший блок генерує заявки із середнім інтервалом 0.2, а три процесори мають власні характеристики обслуговування і різну кількість паралельних пристроїв.

```
Final statistics:
CreateElement1:
Num created: 4862.

ProcessElement3:
Num processed: 1672. Mean queue size: 0.51602. Mean busy handlers: 1.73395. Mean wait time: 0.30865. Failure probability: 0.50619

ProcessElement2:
Num processed: 3391. Mean queue size: 4.62234. Mean busy handlers: 6.76699. Mean wait time: 1.36324. Failure probability: 0.17657

ProcessElement1:
Num processed: 4122. Mean queue size: 6.11403. Mean busy handlers: 4.88126. Mean wait time: 1.48340. Failure probability: 0.17867
```

Рисунок 2 – Підсумкові статистичні дані моделювання

Для більшої наочності можна звести результати у Таблицю 1.

Таблиця 1 – Результати верифікації

Елемент	**Кількість оброблених (або створених) заявок**	**Середній розмір черги**	**Середнє завантаження пристроїв**	**Середній час очікування**	**Ймовірність відмови**
CreateElement1	4862	-	-	-	-
ProcessElement1	4122	6.11403	4.88126	1.48340	0.17867
ProcessElement2	3391	4.62234	6.76699	1.36324	0.17657
ProcessElement3	1672	0.51602	1.73395	0.30865	0.50619

Зі збільшенням числа обслуговуючих пристроїв зменшується середня довжина черги, оскільки одночасно можуть виконуватися більше операцій. Однак якщо кількість пристроїв надто велика, їхнє сумарне завантаження стане низьким (а це означає надмірне витрачання ресурсів). Високі значення середнього розміру черги та часу очікування найчастіше вказують на недостатню кількість паралельних пристроїв або на високе навантаження (стрімкі надходження заявок). З іншого боку, надто мала черга призводить до високої імовірності відмов, коли заявки не можуть потрапити на обслуговування.

Посилання на репозиторій з кодом: https://github.com/I-delver-I/system-modelling.

Висновок

У ході виконання лабораторної роботи було розроблено та реалізовано

імітаційну модель процесу обслуговування заявок із використанням об'єктно-орієнтованого підходу. Вдалося створити систему, що включає генератор заявок (**CreateElement**), оброблювачі (**ProcessElement**) із чергами та обмеженнями на кількість пристроїв, а також модуль моделювання (**Model**), який керує подіями в часі.

Проведене моделювання дозволило оцінити характеристики системи при різних параметрах, включаючи середній розмір черги, середнє завантаження пристроїв, час очікування та ймовірність відмов. Отримані результати підтвердили, що збільшення кількості обслуговуючих пристроїв призводить до зменшення середнього часу очікування та розміру черги, але при цьому надмірна кількість пристроїв може знизити ефективність використання ресурсів.

Виконані експерименти показали, що правильний вибір параметрів (кількість пристроїв, обмеження на чергу, швидкість обробки) дозволяє досягти балансу між продуктивністю та ефективністю роботи системи.