Н. С. Бакуменко, І. О. Трофимова, М. О. Хайленко

ІМІТАЦІЙНО-ПОДІЙНЕ МОДЕЛЮВАННЯ З ВИКОРИСТАННЯМ СИСТЕМ GPSS І ANYLOGIC

2017

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Н. С. Бакуменко, І. О. Трофимова, М. О. Хайленко

ІМІТАЦІЙНО-ПОДІЙНЕ МОДЕЛЮВАННЯ

З ВИКОРИСТАННЯМ СИСТЕМ GPSS І ANYLOGIC

Навчальний посібник

Харків «ХАІ» 2017

УДК 004.942:519.872(075.8)
Б19

Рецензенти: д-р фіз.-мат. наук, проф. С. П. Кунцевич,
д-р техн. наук, проф. Р. М. Тріщ

Бакуменко, Н.С.
Б19 Імітаційно-подійне моделювання з використанням систем GPSS та AnyLogic [Текст]: навч. посіб. / Н. С. Бакуменко, І. О. Трофимова, М. О. Хайленко. – Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2017. – 41 с.

Розглянуто основи імітаційно-подійного моделювання у середовищах GPSS і AnyLogic. Наведено принципи моделювання одноканальних і багатоканальних систем масового обслуговування. Матеріал проілюстровано прикладами розв'язання практичних задач. Запропоновано завдання для самостійної роботи.

Для студентів, що навчаються за спеціальностями «Прикладна математика», «Комп'ютерні науки» в рамках курсу «Імітаційно-подійне моделювання».

Іл. 25. Табл. 2. Бібліогр.: 6 назв.

УДК 004.942:519.872(075.8)

ВСТУП

Моделювання є одним із способів пізнання дійсності. Коли аналіз досліджуваної системи є складною або дорогою процедурою, вдаються до моделювання. Найчастіше моделювання є єдиним способом дослідження об'єкта, що цікавить, або процесу. Наприклад, перед створенням відділення банку логістичні компанії на основі даних про щільність заселення району, наявність компаній-конкурентів обчислюють очікувану кількість клієнтів і моделюють роботу відділення, намагаючись оптимізувати кількість касирів.

Виділяють два види моделювання: математичне, основане на строгих математичних моделях, та імітаційне. Імітаційне моделювання полягає в розробленні та виконанні програм на комп'ютері. Імітаційне моделювання – обширна галузь, в якій можна виділити чотири основні напрями: моделювання динамічних систем, дискретно-подійне моделювання, системна динаміка і агентне моделювання.

Системи масового обслуговування становлять дуже важливий сектор економіки і людської діяльності, а необдумана експлуатація їх може не тільки значно зменшувати очікуваний прибуток, але й призводити до їхньої збитковості. Існують два підходи до моделювання систем масового обслуговування: розв’язання складної системи рівнянь Колмогорова–Чепмена і комп'ютерний експеримент з використанням методу дискретно-подійного моделювання.

Багато економічних завдань пов'язано із системами масового обслуговування, тобто такими системами, в яких, з одного боку, виникають масові запити (заявки) на виконання будь-яких послуг, а з іншого, – відбувається задоволення цих запитів. Система масового обслуговування містить такі елементи: джерело вимог, вхідний потік вимог, черга, обслуговуючі пристрої (канали обслуговування), вихідний потік вимог. Такі системи досліджують з використанням теорії масового обслуговування.

Методами теорії масового обслуговування може бути вирішено багато завдань дослідження процесів, що відбуваються в економіці. Так, в організації торгівлі ці методи дозволяють визначити оптимальну кількість торгових точок одного профілю, кількість продавців, частоту завезення товарів та ін. Однією з характерних ознак систем масового обслуговування можуть бути склади або бази постачальницько-збутових організацій. Завдання теорії масового обслуговування в цьому випадку зводиться до того, щоб установити оптимальне співвідношення між кількістю вимог до обслуговування, що надходять на базу, і кількістю обслуговуючих пристроїв, при використанні яких сумарні витрати на обслуговування і збитки від простою транспорту були б мінімальними. Теорію масового обслуговування можна застосовувати і при розрахунку площі складських приміщень; при цьому складську площу розглядають як обслуговуючий пристрій, а надходження транспортних засобів під вивантаження – як

вимогу. Моделі теорії масового обслуговування застосовують також при вирішенні ряду завдань організації та нормування праці, інших соціально-економічних проблем.

Процес моделювання складається з двох етапів: розроблення моделі й аналізу отриманої моделі.

Для розроблення імітаційних моделей протягом останніх чотирьох десятиліть було створено безліч інструментів. Залежно від парадигми моделювання можна виділити один або кілька програмних комплексів, що значно спрощують процес створення моделей. Наприклад, при розробленні безперервних систем світовим лідером є програмний продукт Matlab / Simulink, але його дуже погано пристосовано до роботи з дискретними системами. Для створення таких моделей зручно використовувати програмні продукти GPSS або AnyLogic.

1 МОДЕЛЮВАННЯ СИСТЕМ З ОДНИМ ПРИСТРОЄМ І ЧЕРГОЮ

1.1 Моделювання систем з одним пристроєм і чергою в GPSS

GPSS (General Purpose Simulation System – система імітаційного моделювання загального призначення) – це середовище моделювання, яке дозволяє моделювати дискретні й безперервні процеси та проводити експерименти на ЕОМ. В основу системи GPSS покладено мову імітаційного моделювання GPSS – мову декларативного типу, основними елементами якої є транзакти і блоки.

Транзакт GPSS являє собою об'єкт з набором атрибутів. При створенні транзакти нумерують послідовно, починаючи з 1, тобто кожному транзакту в системі надається унікальний номер. У процесі моделювання транзакти переходять від одного блока до іншого, при цьому на них впливають інші об’єкти GPSS.

Блоки мови GPSS є підпрограмами, що виконують певний набір функцій. Кожен блок має набір параметрів (операндів), які визначають його роботу.

Процес моделювання в системі GPSS запускається за допомогою команди ***START***. Формат команди:

START A, B, C, D,

де А – лічильник завершення, обов'язковий операнд, значення якого може бути цілим додатним числом. Лічильник завершення являє собою комірку пам'яті з ім'ям TG1, яка зберігає додатне ціле число. Це число застосовують для керування тривалістю процесу моделювання та записують в комірку TG1 командою START на початку процесу моделювання;

B – операнд виведення стандартних даних. Значення за замовчуванням – Null, що відповідає стандартному звіту, значення NP ("no printout") означає, що виведення даних не здійснюватиметься;

C – не використовується, підтримується для забезпечення сумісності версій;

D – виведення списків; значення 1 означає, що списки поточних і майбутніх подій будуть наведені в стандартному звіті, Null – не будуть.

Для внесення транзактів у модель використовують блок ***GENERATE***. Інтервал часу між послідовними появами транзактів з блока ***GENERATE*** називають інтервалом надходження. Формат блока:

GENERATE [A], [B], [C], [D], [E],

де A – середнє значення інтервалу надходження (значення за замовчуванням – 0);

B – величина розкиду можливих значень щодо середнього значення. Якщо операнд B не задано, то інтервал часу надходження – детермінована величина (тобто значення за замовчуванням дорівнює 0);

C – момент часу, в який у блоці ***GENERATE*** має з'явитися перший транзакт (значення за замовчуванням – 0);

D – обмежувач загальної кількості транзактів, яка може увійти в модель через даний блок ***GENERATE*** протягом часу моделювання (значення за замовчуванням – ∞);

E – рівень пріоритету кожного з транзактів, що може приймати значення від 0 до 127. Чим більше число, тим вище пріоритет (значення за замовчуванням – 0).

Для видалення транзактів з моделі використовують блок ***TERMINATE***. Формат блока:

TERMINATE [A].

Операнд A є величиною зменшення лічильника завершення. Цей операнд задає величину, яка віднімається від лічильника кожен раз, коли транзакт входить у блок ***TERMINATE***. За замовчуванням A = 0.

Блоки ***SEIZE*** і ***RELEASE*** використовують для моделювання роботи одноканального пристрою. Блок ***SEIZE*** дозволяє транзактам зайняти пристрій. Якщо пристрій є вільним, транзакт негайно отримує право зайняти його, входить до блока ***SEIZE*** і намагається перейти до наступного блока. Якщо пристрій вже зайнято, транзакт записується у список затримки пристрою і не входить у блок ***SEIZE***. Формат блока:

SEIZE Ім'яПристрою.

Блок ***RELEASE*** звільняє зайнятий пристрій і виключає транзакт зі змагання за пристрій. Формат блока:

RELEASE Ім'яПристрою.

Для подання й збору статистики про використання пристрою в стандартному звіті GPSS наводяться такі характеристики:

- ENTRIES – кількість входів у пристрій;
- UTIL. – коефіцієнт використання;
- AVE. TIME – середній час перебування транзакту в пристрої;
- AVAIL. – стан готовності;
- OWNER – номер останнього транзакту, що зайняв пристрій;
- PEND – кількість перерваних у пристрої транзактів;
- INTER – кількість транзактів, що переривають пристрій;
- RETRY – кількість транзактів, які очікують спеціальних умов;
- DELAY – кількість транзактів, які очікують зайняття пристрою.

Для часової затримки в системі GPSS використовують блок ***ADVANCE***. Він затримує просування транзакту на заданий проміжок модельного часу. Формат команди:

ADVANCE A, [B].

Операнд А визначає середнє значення інтервалу затримки, операнд В – величину розкиду можливих значень щодо середнього значення (за замовчуванням – 0). Інтервал часу, на який затримується транзакт у блоці ***ADVANCE***, може обчислюватися різними способами залежно від того, скільки параметрів задано у блоці. Якщо визначено лише операнд А, то затримка здійснюється рівно на А одиниць модельного часу. Якщо задано обидва параметри А і В, то як інтервал затримки використовується випадкове число, яке лежить в діапазоні від А - В до А + В.

Блоки ***QUEUE*** і ***DEPART*** служать для збору статистики, що зберігається в об'єкті, названому *чергою*.

Коли транзакт входить до блока ***QUEUE***, вміст черги збільшується, а коли до блока ***DEPART***, – зменшується.

Формат команд:

QUEUE A [,B];

DEPART A [,B],

де А – ім'я або номер черги;

В – кількість елементів, на яку збільшується (зменшується) довжина черги (за замовчуванням – 1).

При вході транзакту в блок ***QUEUE*** лічильник входів для даної черги збільшується на В, довжина черги (лічильник поточного вмісту) збільшується на В, значення поточної довжини черги зберігається в стандартному числовому атрибуті Q$<ім'я черги>, і транзакт приєднується до черги с запам'ятовуванням її імені і значення поточного модельного часу.

Транзакт перестає бути елементом черги тільки після того, як переходить у блок ***DEPART*** відповідної черги. При його виході довжина

черги зменшується на В. Після цього визначається, чи є час, проведений транзактом у черзі, нульовим і, якщо так, то такий транзакт є транзактом з нульовим перебуванням у черзі і змінюється лічильник нульових входжень.

Для збору статистики про використання пристрою в стандартному звіті GPSS наводяться такі характеристики:

– QUEUE – номер або ім'я черги;
– ENTRY – загальна кількість входів;
– CONT. – поточна довжина черги;
– AVE, TIME – середній час перебування транзакту в черзі;
– ENTRY (0) – кількість «нульових» входів;
– AVE.CONT – середня довжина черги;
– AVE.(0) – середній час перебування транзакту в черзі без урахування «нульових» входів;
– RETRY – кількість транзактів, які очікують спеціальних умов.

Приклад 1.1. Розглянемо роботу приміської залізничної каси. Час приходу пасажирів розподілено рівномірно на інтервалі 5±2 хвилини. Час обслуговування пасажира також розподілено рівномірно на інтервалі 4±2 хвилини. Промоделювати роботу каси протягом восьми годин. Проаналізувати статистику використання каси і черги.

Програму мовою GPSS, що моделює роботу залізничної каси, показано на рисунку 1.1.

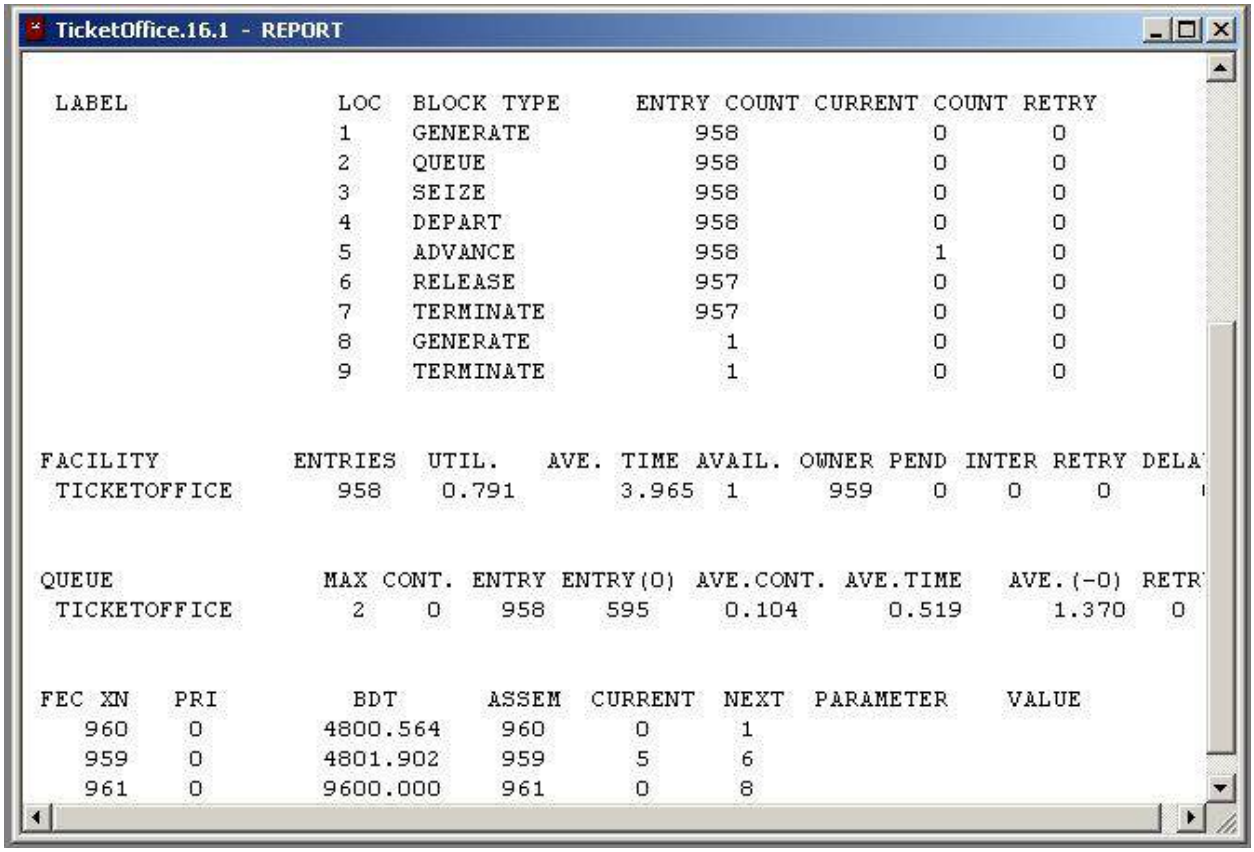
TicketOffice.gps

```
GENERATE 5,2 ;прибытие пассажиров с интервалом от 3 до 7 секунд
QUEUE TICKETOFFICE; вход в очередь;
SEIZE TICKETOFFICE; занятие устройства "железнодорожная касса"
DEPART TICKETOFFICE;  выход из очереди
ADVANCE 4,2; обслуживание в кассе;
RELEASE TICKETOFFICE; освобождение устройства
TERMINATE ;уничтожение транзакта

GENERATE 4800; таймер-сегмент
TERMINATE 1
```

Рисунок 1.1 – Текст програми, що моделює роботу залізничної каси

Фрагмент звіту за результатами моделювання зображено на рисунку 1.2. Із звіту про використання пристрою TICKETOFFICE видно, що за час роботи каси було обслуговано 958 осіб, коефіцієнт використання пристрою – 0.79, середній час обслуговування – 3.965 хвилин. У черзі до каси знаходилося 595 осіб, середній час їх очікування – 0.519 хвилин.



TicketOffice.16.1 - REPORT

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	958	0	0
	2	QUEUE	958	0	0
	3	SEIZE	958	0	0
	4	DEPART	958	0	0
	5	ADVANCE	958	1	0
	6	RELEASE	957	0	0
	7	TERMINATE	957	0	0
	8	GENERATE	1	0	0
	9	TERMINATE	1	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELA
TICKETOFFICE	958	0.791	3.965	1	959	0	0	0	

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETR
TICKETOFFICE	2	0	958	595	0.104	0.519	1.370	0

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
960	0	4800.564	960	0	1		
959	0	4801.902	959	5	6		
961	0	9600.000	961	0	8		

Рисунок 1.2 – Фрагмент звіту за результатами моделювання

1.2 Моделювання систем з одним пристроєм і чергою в AnyLogic

1.2.1 Основи моделювання в системі AnyLogic

AnyLogic – продукт нового покоління, що дозволяє створювати моделі будь-якої вибраної парадигми, безперервної або дискретної, або навіть поєднувати їх.

Система AnyLogic має спеціальну бібліотеку під назвою Enterprise Library, призначену для дискретно-подійного моделювання. Ця бібліотека містить всі необхідні компоненти, які відображатимуть типові вузли систем масового обслуговування, наприклад, джерела заявок, черги, пристрої обслуговування та ін.

Процес створення найпростішої моделі полягає в складанні схеми системи масового обслуговування з наявних компонентів (використовуючи технологію drag'n'drop), їх з'єднанні і параметризації. Для створення більш складних систем необхідно додавати фрагменти коду мовою програмування Java, що реалізують логіку роботи системи.

За своєю функціональністю компоненти бібліотеки можна поділити на шість категорій: потік заявок, робота з вмістом заявки, оброблення, робота з ресурсами, транспортування, транспортування по мережі.

При побудові блок-схем системи масового обслуговування (СМО) важливо розуміти, як заявки переміщуються між компонентами. Заявки надходять у компонент і залишають його через порти. Виділяють вхідний і вихідний порти. Вхідний порт можна поєднати тільки з вихідним портом. При надходженні заявки або її виході компонент генерує подію, оброблення якої не є обов'язковим, але в деяких випадках допомагає в проектуванні моделі.

Для з'єднання компонентів використовують **З'єднувач (Соединитель),** який знаходиться на вкладці **Головна** (**Основная)** палітри інструментів.

Усі компоненти бібліотеки Enterprise Library знаходяться на вкладці **Enterprise Library**.

Заявками в системі є екземпляри вбудованого класу **ENTITY** або його підкласів.

Параметром компонента може бути будь-який коректний код Java, який повертає значення необхідного типу.

1.2.2 Генерація заявок у системі AnyLogic

Для генерації заявок використовують компонент Source. Його властивості зображено на рисунку 1.3.

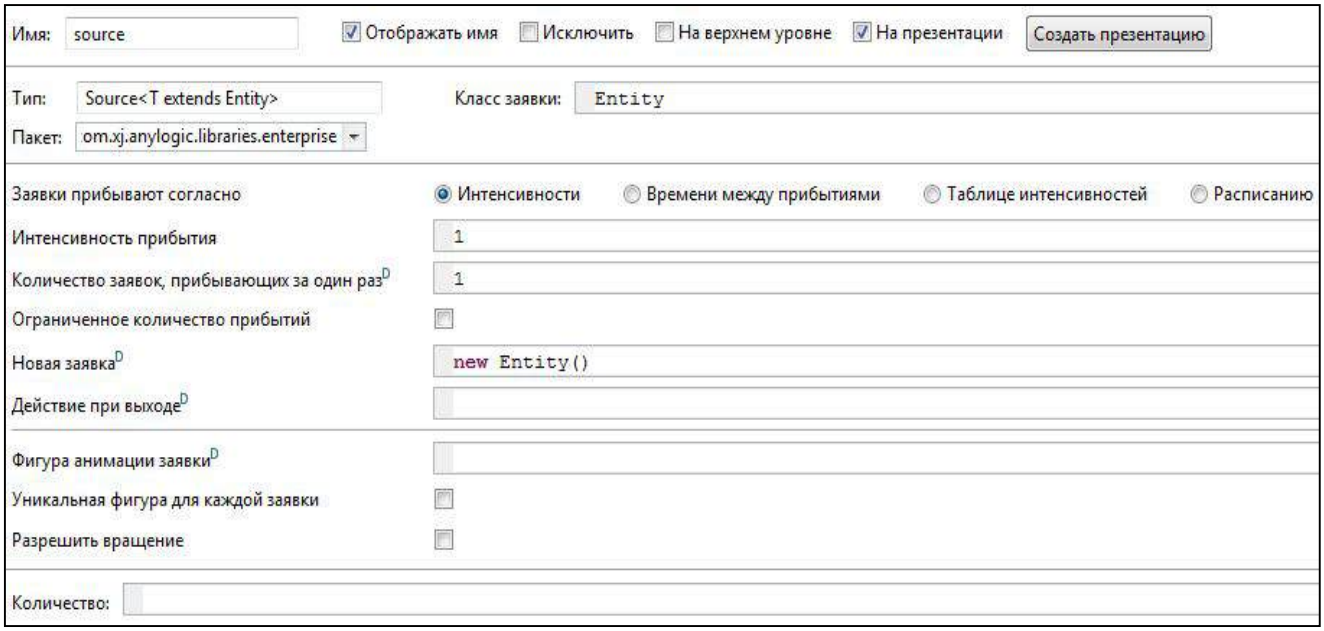


Рисунок 1.3 – Властивості компонента *Source*

Розглянемо основні властивості компонента *Source*:

1) *Ім'я (Имя)* – ідентифікатор об’єкта в системі;
2) *Клас заявки (Класс заявки)* вказує, екземпляри якого класу генерує компонент. Це має бути клас Entity або його підкласи;

3) *Заявки надходять згідно з (Заявки прибывают согласно)* – визначає, чи будуть заявки прибувати згідно з такими варіантами:
   - *Інтенсивністю (Интенсивности)* – заявки надходять згідно з заданою інтенсивністю прибуття (еквівалентною експоненціально розподіленому часу між прибуттям із середнім значенням, що дорівнює 1 / інтенсивність);
   - *Часом між прибуттями* (Времени между прибытиями) – часом між двома послідовними прибуттями, що визначається заданим виразом (цю опцію можна використовувати для періодичної генерації заявок або генерації заявок з інтервалами часу, що не розподілені за експоненціальним законом);
   - *Таблицею інтенсивностей (Таблице интенсивностей)* – заявки генеруються згідно з таблицею інтенсивностей – табличною функцією, що визначає, як інтенсивність прибуття заявок змінюється з часом;
   - *Розкладом (Расписанию)* – заявки генеруються відповідно до розкладу – табличної функції, в якій задано часи прибуття заявок і кількість заявок, що надходять в кожний зазначений у таблиці момент часу;
   - *Викликами методу inject() (Вызовам метода inject())* – заявки створюються не автоматично, а тільки під час виклику методу inject();
4) *Інтенсивність прибуття (Интенсивность прибытия)* – інтенсивність надходження заявок. При виборі варіанту *Часом між прибуттями* (*Времени между прибытиями)* поле *Інтенсивність прибуття (Интенсивность прибытия)* зникає та з'являється поле *Час між прибуттями (Время между прибытиями)*, в якому вказується час, через який надходять заявки, або закон розподілу. В таблиці 1.1 наведено деякі закони розподілу.

Таблиця 1.1 – Функції для визначення законів розподілу випадкових величин в AnyLogic

Закон розподілу	**Функція в AnyLogic**
Експоненціальний	exponential(double lambda, double min)
Рівномірний	uniform(double min, double max)
Рівномірний дискретний	uniform_discr(int min, int max)
Трикутний	triangular(double min, double max, double mode)
Нормальний	uniform(double min, double max)

Наприклад, якщо час надходження заявки розподілено рівномірно на інтервалі 10 – 30 секунд, то в полі *Час між прибуттями (Время между прибытиями)* необхідно записати uniform_discr(10, 30);

5) *Кількість заявок, що надходять за один раз (Количество заявок, прибывающих за один раз)* – це постійне ціле значення або необхідний закон розподілу;
6) *Обмежена кількість прибуттів* (*Ограниченное количество прибытий)* – це перемикач, при активації якого з'являється поле *Максимальна кількість прибуттів* (*Максимальное количество прибытий),* в якому вказується ціле значення;
7) *Нова заявка (Новая заявка)* – це код, який виконується при генерації заявки;
8) *Дія при виході (Действие при выходе)* – код, який виконується при виході заявки з компонента.

Компонент також містить змінну *entity* типу *Клас заявки* (*Класс заявки)*, що вказує на поточну заявку.

Функція **count()** повертає кількість заявок, створених компонентом, функція **inject(int n)** генерує n заявок.

1.2.3 Оброблення заявок у системі AnyLogic

Для оброблення заявок в AnyLogic використовується компонент Delay. Його властивості зображено на рисунку 1.4.

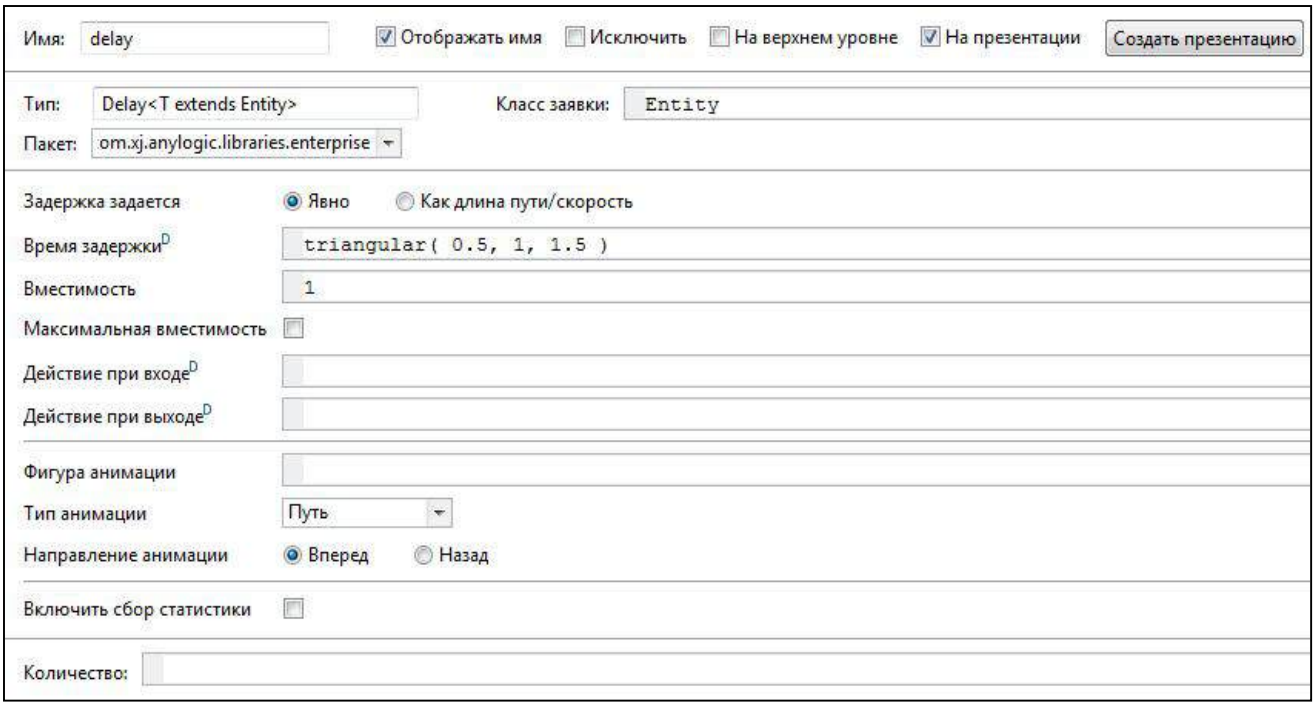


Рисунок 1.4 – Властивості компонента *Delay*

Розглянемо основні властивості компонента *Delay:*

1) *Ім'я (Имя)* – ідентифікатор об'єкта в системі;
2) *Клас заявки (Класс заявки)* вказує екземпляри якого класу генерує компонент, має належати класу *Entity* або його підкласам;

3) *Час затримки (Время задержки)* – час оброблення запиту, константне значення або функція розподілу;
4) *Місткість (Вместимость)* – кількість заявок, що можуть оброблятися одночасно;
5) *Максимальна місткість* (*Максимальная вместимость)* – це перемикач, який вказує, що компонент може оброблювати нескінчену кількість заявок;
6) *Дія при вході (Действие при входе)* – код, що виконується при надходженні заявки;
7) *Дія при виході (Действие при выходе)* – код, що виконується при виході заявки;
8) *Включити збір статистики* (*Включить сбор статистики)* – це перемикач, який вказує, що треба виконати збір статистики під час роботи моделі. Найбільший інтерес при ввімкненій опції має значення коефіцієнта використання компонента. Статистика відображається при виборі компонента в режимі роботи моделі.

Функції компонента *Delay*:

size() – повертає кількість заявок, що обробляються у даний момент;

get(int i) – повертає *i*-ту заявку, яка є об'єктом класу *Клас заявки (Класс заявки)*.

Кожну заявку, що надходить в модель, має бути вилучено компонентом Sink. Цей компонент має такі властивості: *Ім'я* (*Имя)*, *Клас заявки* (*Класс заявки)* та *Дія при вході (Действие при входе)*.

1.2.4 Оброблення черг у системі AnyLogic

При надходженні заявки на компонент *Delay*, якщо він обробляє в цей момент максимальну кількість заявок, генерується виняткова ситуація, яка зупиняє моделювання. Для запобігання цієї проблеми використовують компонент Queue, властивості якого зображено на рисунку 1.5.

Розглянемо основні властивості компонента *Queue*:

1) *Ім'я* (*Имя)*, *Клас заявки* (*Класс заявки)*, *Дія при вході (Действие при входе)*, *Дія при виході* (*Действие при выходе)* – аналогічні властивостям розглянутих компонентів Source і Delay;
2) *Місткість (Вместимость)* – кількість заявок, які може зберігати компонент. Якщо перемикач *Максимальна місткість* (*Максимальная вместимость)* є активованим, то кількість заявок необмежена. При перевищенні кількості заявок генерується виняткова ситуація;
3) *Дозволити вихід за тайм-аутом* (*Разрешить уход по тайм-ауту)* – це перемикач, який задає автоматичне видалення заявок через час, указаний в полі *Тайм-аут*, яке з'являється при активації перемикача;
4) *Дозволити витіснення (Разрешить вытеснение)* – це перемикач, який замінює тип черги FIFO (перший прийшов, перший вийшов, тип черги за

замовчуванням) на чергу з пріоритетом. Пріоритет заявки, що надійшла, вказується в полі *Пріоритет заявки* (*Приоритет заявки)*, яке з'являється при активації перемикача.

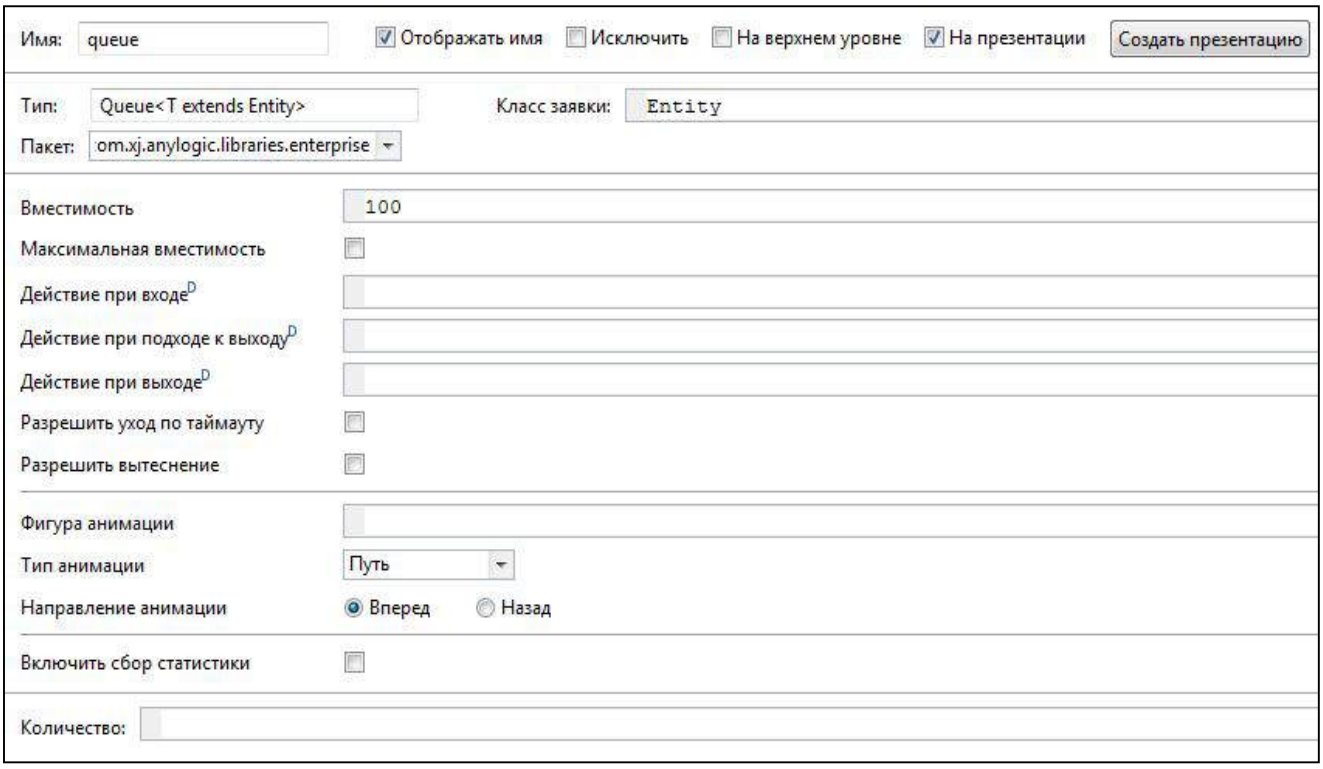


Рисунок 1.5 – Властивості компонента *Queue*

Поля *Тайм-аут*, *Пріоритет заявки* (*Приоритет заявки)* містять локальну змінну *entity*, що вказує на заявку, яка покидає компонент через перевищення часу перебування, або на заявку, що надходить з черги з пріоритетом.

При активації перемикача *Включити збір статистики* (*Включить сбор статистики)* виконується збір статистики під час роботи моделі. Найбільш цікавою є властивість *Довжина черги*.

Функції компонента *Queue*:

size() – повертає кількість заявок у черзі;

get(int i) – повертає *i*-ту заявку, яка є об'єктом класу *Клас заявки (Класс заявки)*.

Дуже часто під час вирішення нетривіальних завдань зручно конкретизувати заявку, додавши в клас *Entity* нові властивості (поля). Для цього необхідно створити підклас класу *Entity*.

Для створення Java класу в середовищі AnyLogic необхідно вибрати пункт меню *Файл->Створити->Java клас (Файл->Создать->Java класс)* і в діалоговому вікні (рисунок 1.6) ввести ім'я і суперклас.

Під час натискання на кнопку *Далі (Далее)* з'являється вікно, в якому можна додати необхідні поля (рисунок 1.7).

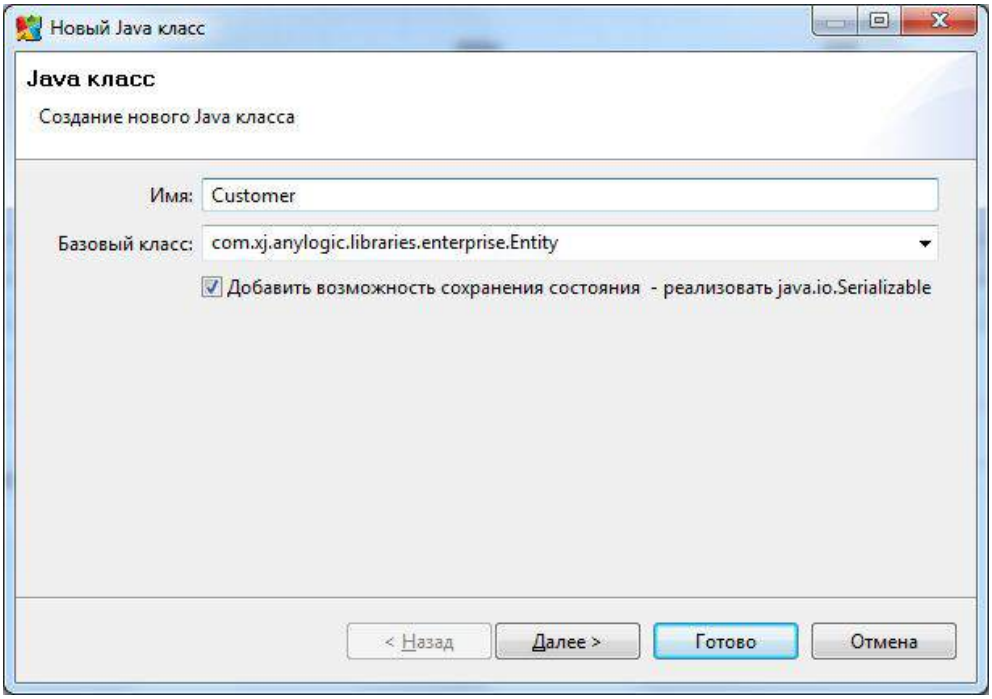


Рисунок 1.6 – Діалогове вікно створення Java класу

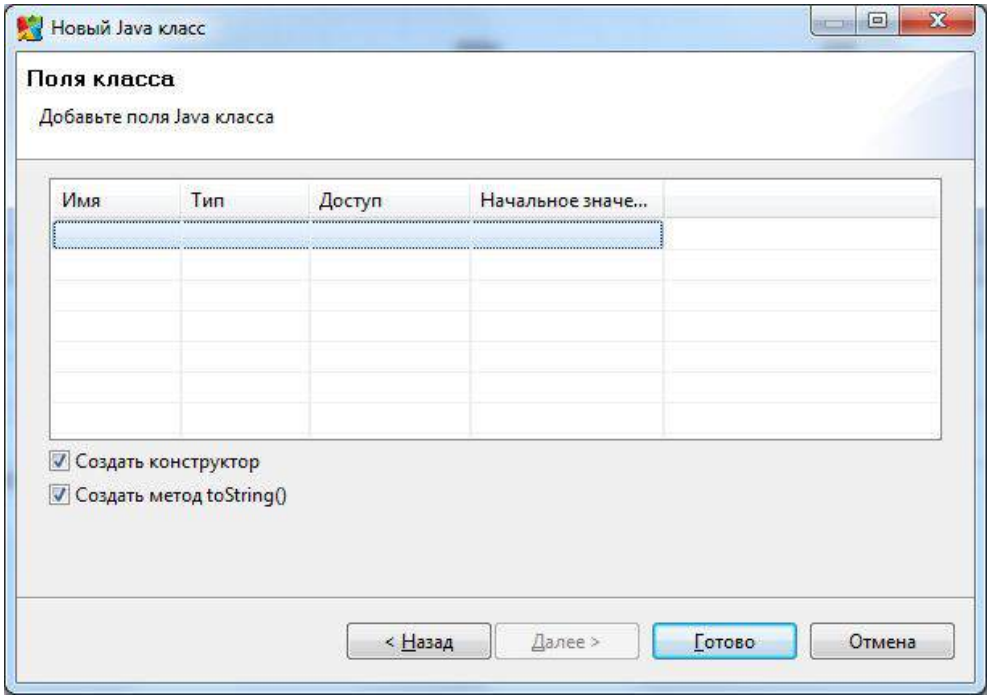


Рисунок 1.7 – Вікно додавання полів до класу

Під час активації перемикача *Створити конструктор (Создать конструктор)* створюється конструктор, в який передаються всі додані поля.

Під час активації перемикача *Створити метод toString() (Создать метод toString().)* автоматично генерується метод *toString().*

Приклад 1.2. Розглянемо роботу приміської залізничної каси. Час приходу пасажирів розподілено рівномірно на інтервалі 5±3 хвилини. Час обслуговування пасажира також розподілено рівномірно на інтервалі 2±1 хвилини. Провести моделювання роботи каси протягом восьми годин.

Після запуску на екрані з'явиться головне вікно програми з відкритою початковою сторінкою, яку можна закрити, натиснувши на хрестик поруч з написом *Початкова сторінка (Начальная страница).* Після закриття початкової сторінки на екрані з'явиться вікно, зображене на рисунку 1.8.

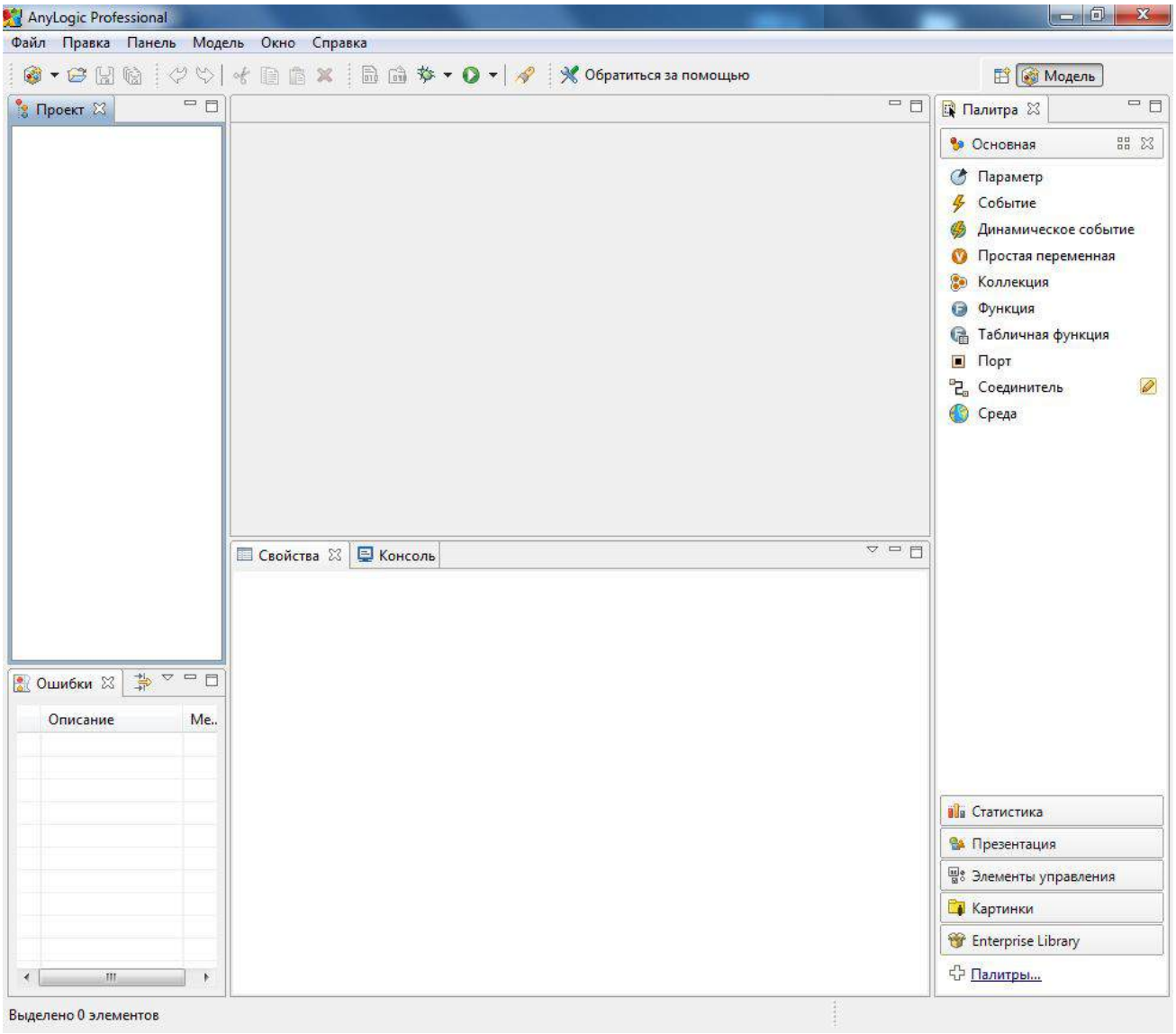


Рисунок 1.8 – Головне вікно системи AnyLogic

Під час вибору елемента меню *Файл->Створити->Модель* (*Файл->Создать->Модель*) на екрані з'явиться майстер створення моделей. У першому вікні майстра необхідно ввести ім'я і місце розташування створюваної моделі. У другому вікні майстра вибирають шаблон створюваної моделі (рисунок 1.9). Необхідно вибрати *Почати створення моделі "з нуля" (Начать создание модели "с нуля")*.

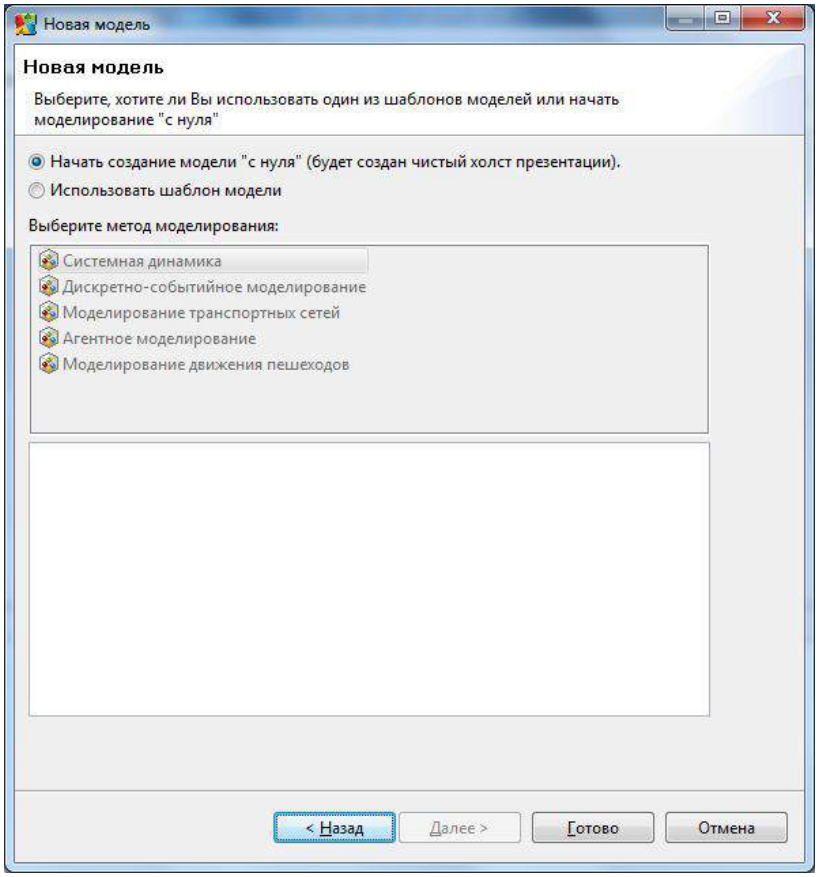


Рисунок 1.9 – Вибір шаблону моделі

Після натискання на кнопку *Готово* створюється модель.
Головне вікно програми приймає вигляд, зображений на рисунку 1.10.

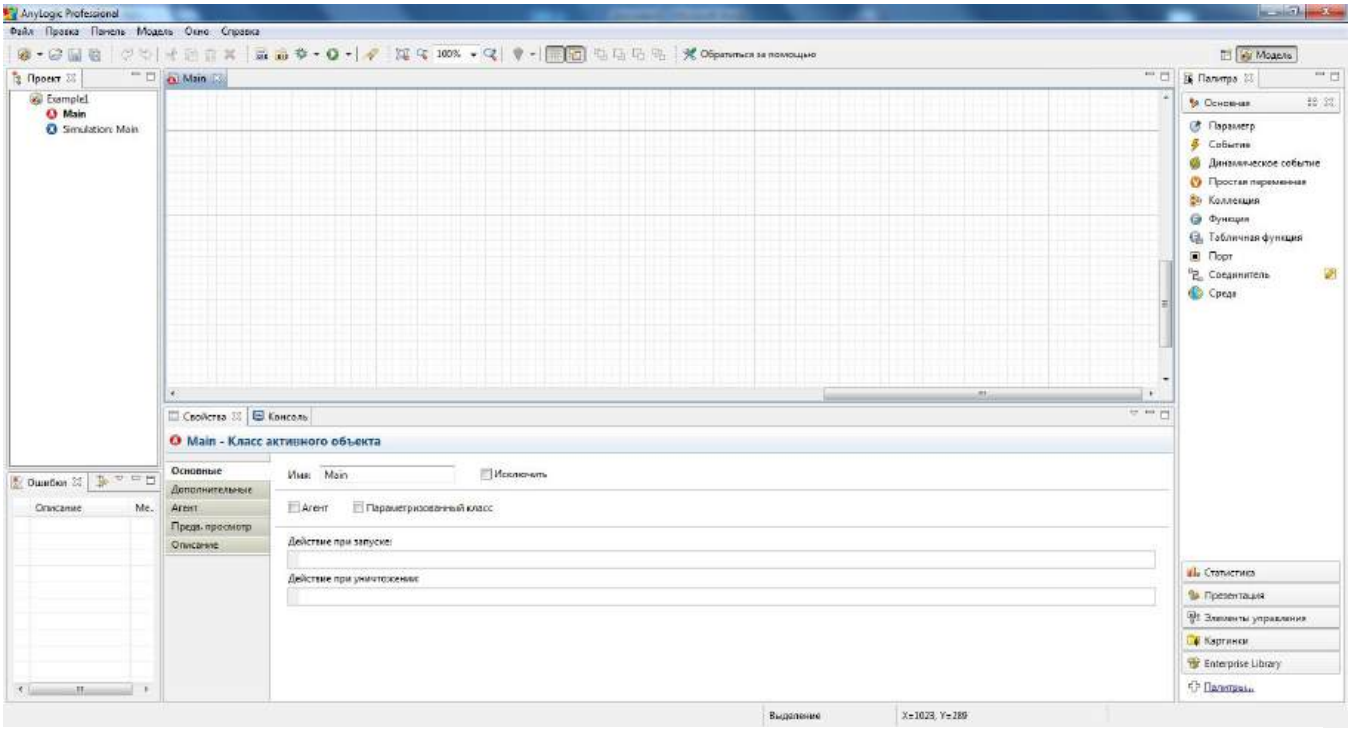


Рисунок 1.10 – Головне вікно після створення моделі

У завданні зазначено, що час прибуття пасажирів розподілено рівномірно на інтервалі 5±3 хвилини. Прибуття пасажира – це заявка в системі. Для створення заявок використовується об'єкт *Source*. Коли приходить пасажир, а каса зайнята, пасажир стає в чергу, отже, необхідним є об'єкт *Queue*. Для обслуговування використовується об'єкт *Delay*. Час обслуговування згідно із завданням – 2 ± 1 хвилини. Після обслуговування пасажир йде з системи, тобто заявка переміщується в об'єкт *Sink*.

На рисунку 1.11 зображено робочу область моделі після додавання і зв'язування необхідних об'єктів.

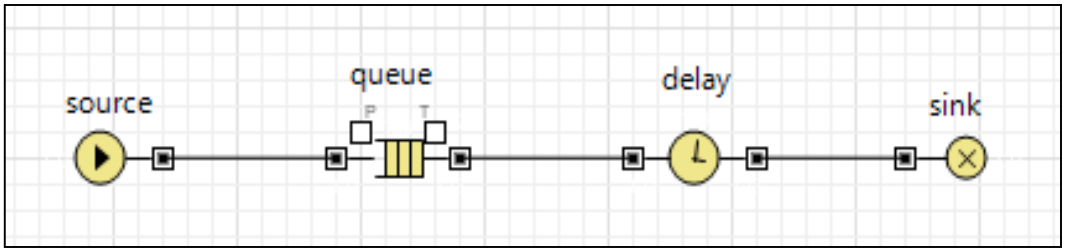


Рисунок 1.11 – Робоча область моделі

Наступний етап – параметризація моделі.

У властивості *Заявки прибувають згідно з/ Часом між прибуттями (Заявки прибывают согласно/ Времени между прибытиями)* об'єкта *Source* вказується рівномірний закон розподілу з мінімальним значенням 2, максимальним – 8: **uniform_discr (2, 8)**. У властивості *Час затримки (Время задержки)* об'єкта *Delay* вказується рівномірний закон розподілу зі значеннями 1 і 3: **uniform_discr (1, 3)**. Для збирання статистики компонентів *Queue* і *Delay* необхідно ввімкнути перемикач *Включити збір статистики (Включить сбор статистики)*.

Перш ніж запускати процес моделювання необхідно створити експеримент і налаштувати його параметри. Середовище AnyLogic підтримує такі типи експериментів: простий, оптимізація, варіювання параметрів, порівняння "прогонів", Монте-Карло, аналіз чутливості, калібрування, нестандартний. Для даної задачі достатньо простого експерименту, який автоматично створюється при створенні моделі.

В елементі головного вікна "Проект" необхідно вибрати створений системою AnyLogic простий експеримент *Simulation: Main*. Потім в елементі *Властивості (Свойства)* перейти на вкладку *Модельний час (Модельное время)* і встановити кінцевий час – 480 хвилин (рисунок 1.12). Для запуску моделі необхідно натиснути на кнопку *Запуск* панелі інструментів.

У наступному вікні слід натиснути кнопку *Запустити модель і відкрити презентацію класу Main (Запустить модель и открыть презентацию класса Main)*. На екрані з'явиться вікно, зображене на рисунку 1.13.

За допомогою кнопок на панелі інструментів можна запустити модель, зробити крок моделювання, призупинити, зупинити, уповільнити, прискорити і перемотати час до кінця відповідно.

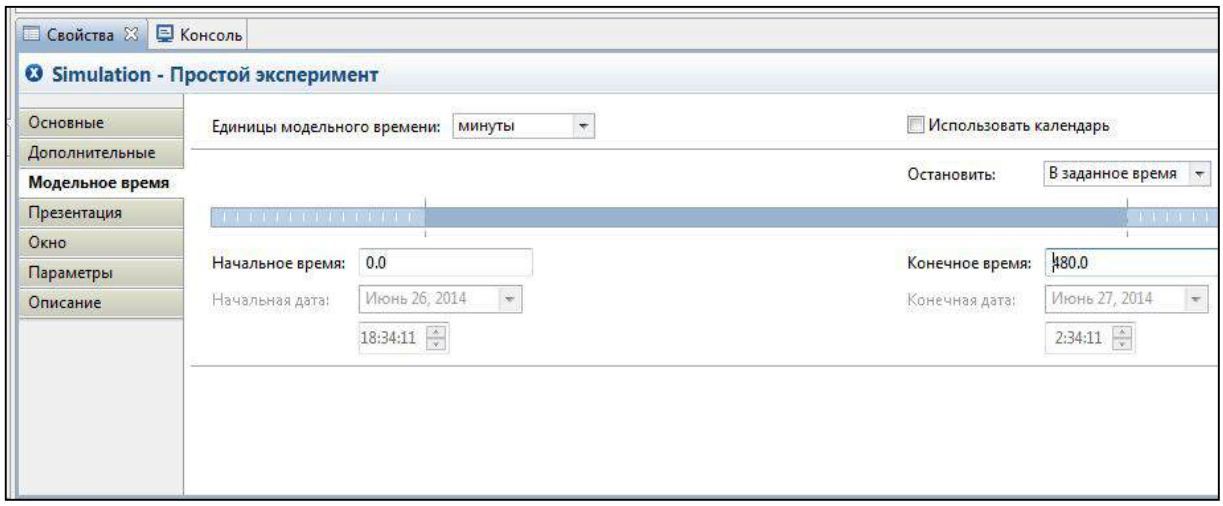


Рисунок 1.12 – Властивості експерименту

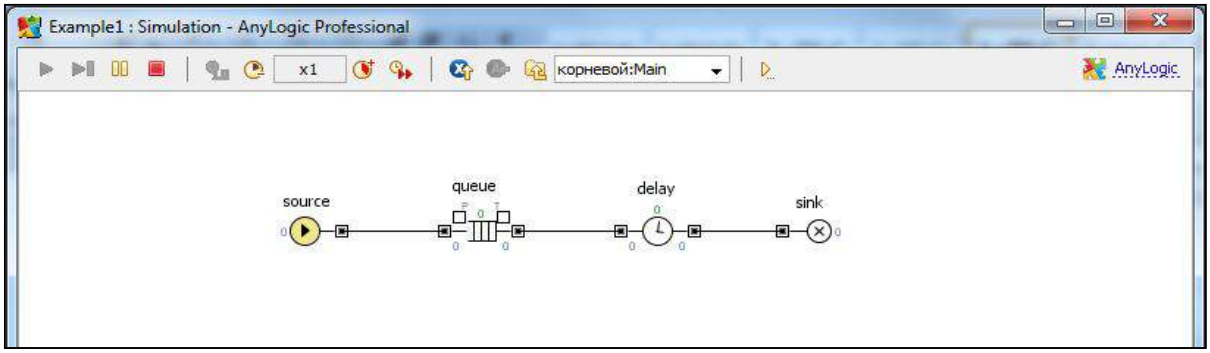


Рисунок 1.13 – Вікно моделювання

Натиснувши на елементи *Queue* і *Delay*, можна побачити їх статистику (рисунок 1.14).

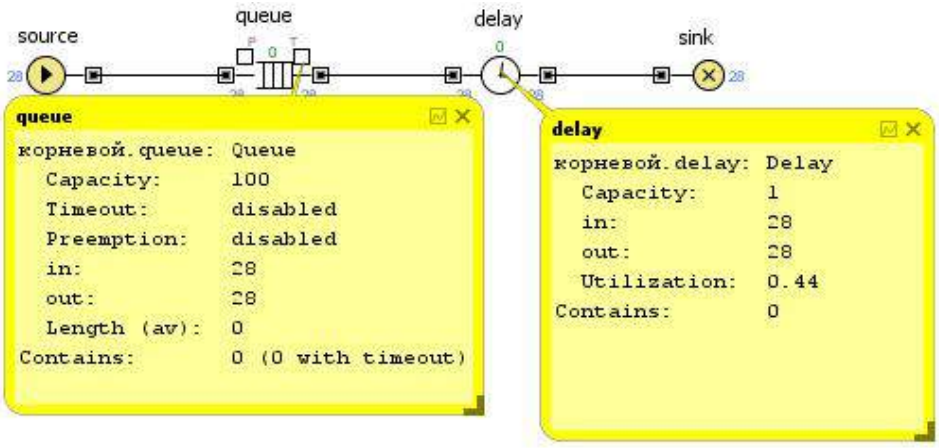


Рисунок 1.14 – Статистика об'єктів *Queue* і *Delay*

У рядку стану знаходиться інформація про поточний час у моделі, час моделювання, кількість зайнятої пам'яті та ін.

1.3 Варіанти завдань для самостійної роботи

1. Змоделювати роботу лікаря-терапевта. Прихід пацієнтів розподілено рівномірно на інтервалі *a*. Час прийому *b* також розподілено рівномірно. Пацієнтів приймають у порядку «першим прийшов – першим обслужили». Модель роботи лікаря має забезпечити збирання статистики про чергу протягом *c* годин.

2. Змоделювати роботу бібліотекаря. Прихід читачів розподілено рівномірно на інтервалі *a*. Час роботи *b* з читачами також розподілено рівномірно. Читачів обслуговують в порядку «першим прийшов – першим обслужили». Модель роботи бібліотекаря має забезпечити збирання статистики про чергу протягом *c* годин.

3. Булочна «Гарячий хліб» має одного контролера-касира. Протягом години приходять в середньому 54 покупця. Середня вартість однієї покупки дорівнює семи гривням. Середній час обслуговування контролером-касиром одного покупця становить одну хвилину. Визначити виручку від продажу, характеристики СМО та проаналізувати роботу булочної.

4. Інтенсивність потоку автомобілів на АЗС до колонки за бензином АІ-92 становить 30 автомобілів за одну годину, а середній час заправки дорівнює п’яти хвилинам. Провести аналіз роботи системи масового обслуговування АЗС.

5. У перукарні працює тільки один чоловічий майстер. Середній час стрижки одного клієнта становить 20 хвилин. Клієнти в середньому приходять кожні 25 хвилин. Середня вартість стрижки – 60 грн. Як в першу зміну – з 9 до 15, так і в другу – з 15 до 21 – працює один майстер. Провести аналіз роботи системи обслуговування.

6. У порту є один причал для розвантаження суден. Інтенсивність потоку суден дорівнює 0,4 (суден на добу). Середній час розвантаження одного судна становить дві доби. Передбачається, що черга може бути необмеженої довжини. Знайти показники ефективності роботи причалу.

2 МОДЕЛЮВАННЯ СИСТЕМ З ОДНИМ ПРИСТРОЄМ І ЧЕРГОЮ ДЛЯ ОБСЛУГОВУВАННЯ З ПРІОРИТЕТОМ

2.1 Моделювання систем з одним пристроєм і чергою для обслуговування з пріоритетом у системі GPSS

Для зміни або присвоювання пріоритету транзакту в GPSS користуються блоком **PRIORITY**, який змінює пріоритет активного

транзакту. Якщо пріоритет транзакта був заданий у блоці **GENERATE**, блок **PRIORITY** змінює його.

Формат блока:

PRIORITY A[,B],

де A – нове значення пріоритету (ціле число);

B – режим BUFFER, який розміщує активний транзакт у списку поточних подій позаду транзактів, що дорівнюють йому за пріоритетом. Допустимі значення – BU або NULL.

У випадку, коли необхідно організовувати обслуговування з перериваннями та припиненням обслуговування більш пріоритетним транзактом менш пріоритетного транзакту, тобто вилученням його з пристрою, необхідно використовувати пару блоків **PREEMPT (ЗАХОПИТИ) – RETURN (ПОВЕРНУТИ)** так само, як для звичайного пристрою без переривань застосовувати блоки **SEIZE – RELEASE**.

Блок **PREEMPT** дозволяє транзакту залежно від умов, заданих в операндах блока, зайняти пристрій. Блок **PREEMPT** може затримати транзакт на вході.

Формат блока:

PREEMPT A,[B],[C],[D],[E],

де A – ім'я пристрою (числове або символьне), в якому генерується переривання;

B – пріоритетний режим. Якщо задано значення PR, то буде реалізовано режим переривання;

C – номер або ім'я блока, в який переходить перерваний транзакт;

D – номер параметра перерваного транзакту, в який записується час, що залишився, якщо обслуговуваний транзакт видаляється зі списку майбутніх подій;

E – режим видалення. Видаляє вилучений транзакт зі змагання за пристрій.

Операнд B задає пріоритетний режим (якщо B = PR) або режим переривання (якщо цей операнд вилучено). Під час роботи в пріоритетному режимі транзакт, який вже займає пристрій або генерує на ньому переривання, може бути перерваний тільки транзактом, пріоритет якого вище пріоритету даного транзакту. Перервані транзакти претендують на додаткове використання пристрою, коли транзакт, що їх перервав, увійде до відповідного блока **RETURN**.

Операнд C задає номер або ім'я блока, до якого в цей же момент має спробувати увійти перерваний транзакт. Такий транзакт втрачає керування пристроєм, але претендує на право його використання, якщо тільки не

задано аргумент операнда E. В пріоритетному режимі роботи бажано задавати операнд C, якщо транзакт, що перериває, має більш високий пріоритет, ніж той, що переривався.

Операнд D задає номер параметра, пов'язаного з перерваним транзактом. Якщо такий транзакт у момент переривання переходить у список майбутніх подій, тоді залишок часу записується в заданий параметр. Якщо такий параметр не існує, то він створюється. В пріоритетному режимі роботи операнд D задають тільки в тому випадку, якщо транзакт, що перериває, має більш високий пріоритет, ніж транзакт, що переривався.

Операнд E задає або не задає режим видалення RE. В цьому режимі перерваний транзакт більш не претендує на використання пристрою і намагається увійти в блок, заданий операндом C (якщо в операнді E знаходиться значення RE, то має бути визначено і операнд C). У пріоритетному режимі роботи режим RE використовується тільки в тому випадку, якщо пріоритет транзакту, що перериває, більше пріоритету транзакту, що переривається. При використанні RE перерваний транзакт не має входити в блоки **RELEASE** і **RETURN**, з'єднані пристроєм, в якому обслуговувався перерваний транзакт. Якщо режим RE не задано (операнд E вилучено), то перерваний транзакт після повернення в список поточних подій буде знову намагатися зайняти пристрій.

Блок **RETURN** є парним блоку **PREEMPT**, також як блок **RELEASE** є парним блоку **SEIZE.** Його призначено для звільнення раніше захопленого пристрою.

Формат блока:

RETURN A,

де A – ім'я пристрою (числове або символьне).

Приклад 2.1. Розглянемо роботу залізничної каси, в якій продають квитки на потяги на сьогодні та здійснюють попередній продаж квитків. Час звернення в касу пасажирів за квитками на потяги на сьогодні розподілено рівномірно на інтервалі 10±3 хвилини, а час обслуговування – 3±1 хвилини. Час приходу пасажирів, які купують квитки завчасно, розподілено рівномірно на інтервалі 20±5 хвилин, а час обслуговування – 10±5 хвилин. Змоделювати роботу системи для випадку, коли пасажирів, які купують квитки на сьогодні, обслуговують в пріоритетному порядку.

Програму мовою GPSS, що моделює роботу залізничної каси, показано на рисунку 2.1.

На рисунку 2.2 зображено фрагмент звіту за результатами моделювання.

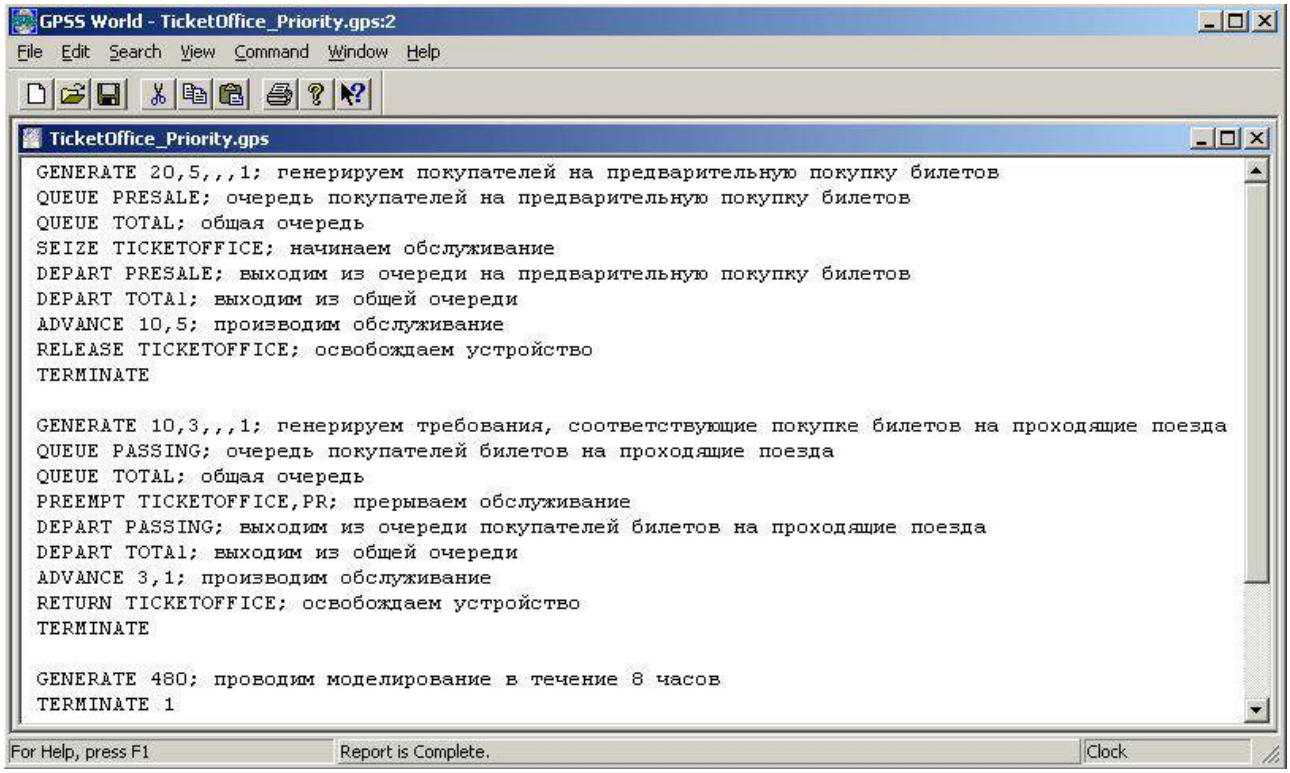


Рисунок 2.1 – Текст програми, що моделює роботу залізничної каси

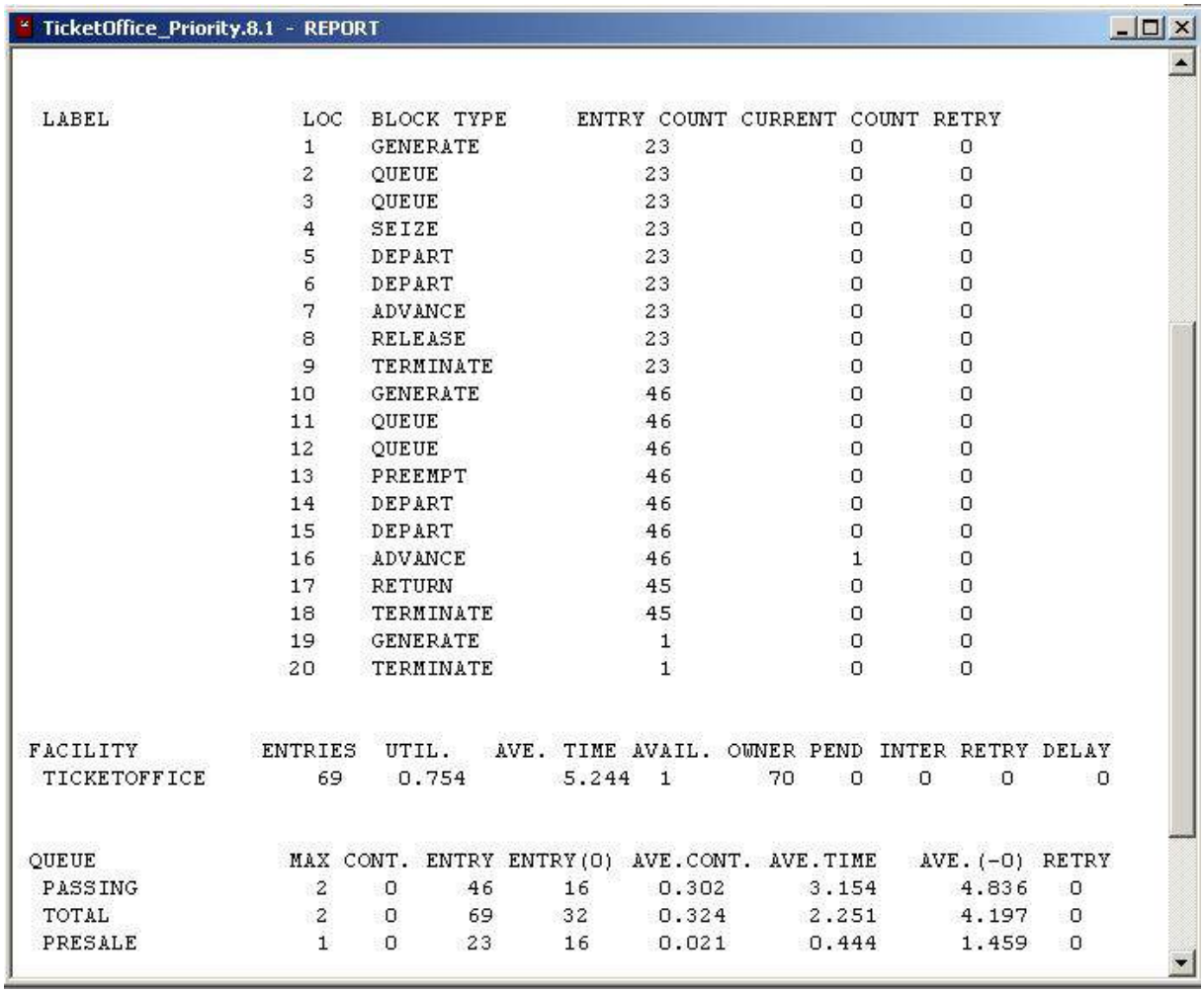


Рисунок 2.2 – Фрагмент звіту за результатами моделювання

2.2 Моделювання систем з одним пристроєм і чергою для обслуговування з пріоритетом у системі AnyLogic

Моделювання обслуговування з пріоритетом в системі AnyLogic розглянемо на прикладі.

Приклад 2.2. Проаналізуємо роботу залізничної каси, в якій продають квитки на потяги на сьогодні та здійснюють попередній продаж квитків. Час звернення в касу пасажирів за квитками на потяги на сьогодні розподілено рівномірно на інтервалі 10±3 хвилини, а час їх обслуговування – 3±1 хвилини. Час приходу пасажирів, які купують квитки завчасно, розподілено рівномірно на інтервалі 20±5 хвилин, а час їх обслуговування – 7±3 хвилини. Змоделювати роботу системи для випадку, коли пасажирів, які купують квитки на сьогодні, обслуговують у пріоритетному порядку.

У цій системі є два типи заявок: пасажири, які купують квитки на сьогодні та пасажири, які купують квитки завчасно. Обидва типи заявок мають однакові властивості (пріоритет і час оброблення заявки), які не містить бібліотечний клас *Entity*.

Для конкретизації цих заявок необхідно створити підклас класу *Entity* з такими додатковими полями: *minTime*, *maxTime* і *priority* типу *int*, модифікатор доступу – *public*. При активованому перемикачі *Створити конструктор (Создать конструктор)* середовище згенерує весь необхідний код.

На рисунку 2.3 зображено схему системи масового обслуговування.

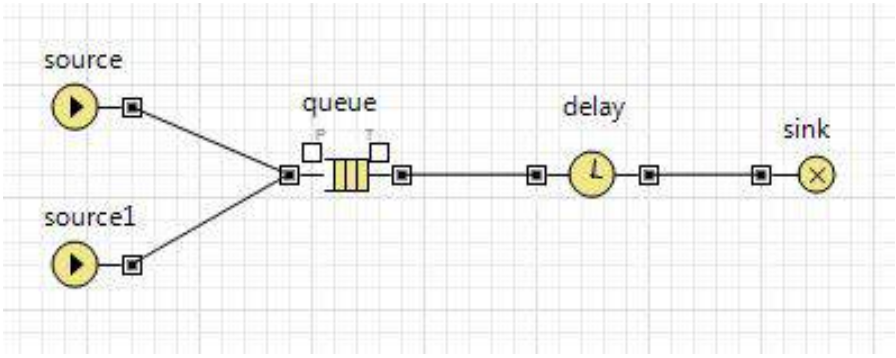


Рисунок 2.3 – Схема системи масового обслуговування

Компоненти схеми мають такі властивості:

1. *Source*:
   1) *Клас заявки (Класс заявки)* – Customer;
   2) *Заявки прибувають згідно з (Заявки прибывают в соответствии с)* – *Часом між прибуттями (Временем между прибытиями)*;
   3) *Час між прибуттями (Время между прибытиями)* – uniform_discr(7, 13);

4) *Нова заявка (Новая заявка)* – new Customer(2, 4, 5), де 2, 4 – мінімальний і максимальний час обслуговування, 5 – пріоритет.

2. *Source1*:
   1) *Клас заявки (Класс заявки)* – Customer;
   2) *Заявки надходять згідно з (Заявки прибывают в соответствии с)* – *Часом між прибуттями (Временем между прибытиями)*;
   3) *Час між прибуттями (Время между прибытиями)* – uniform_discr(15, 25);
   4) *Нова заявка (Новая заявка)* – new Customer(4, 10, 1), де 4, 10 – мінімальний і максимальний час обслуговування, 1 – пріоритет.
3. *Queue*:
   1) *Клас заявки (Класс заявки)* – Customer;
   2) *Дозволити витіснення (Разрешить вытеснение)* активовано;
   3) *Пріоритет заявки (Приоритет заявки)* – entity.priority;
   4) *Включити збір статистики (Включить сбор статистики) активовано.*
4. *Delay*:
   1) *Клас заявки (Класс заявки)* – Customer;
   2) *Час затримки (Время задержки)* – uniform(entity.minValue, entity.maxValue);
   3) *Включити збір статистики (Включить сбор статистики)* активовано.

На рисунку 2.4 зображено стан СМО після восьми годин роботи.

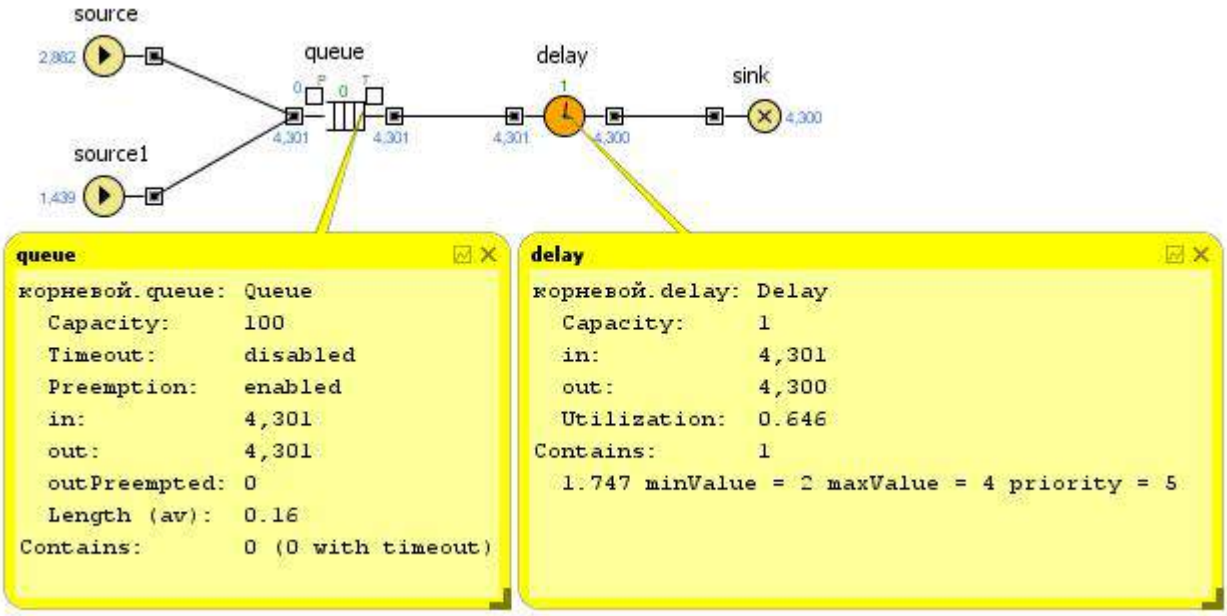


Рисунок 2.4 – Стан СМО

Для розв'язання цієї задачі можна скористатися також технологією блокування заявок.

Для блокування заявок використовується компонент  Hold, який блокує всі заявки, що проходять через нього. Для активації і дезактивації блокування використовується метод **setBlocked(boolean isBlocked).**

2.3 Варіанти завдань для самостійної роботи

1. На прийом до лікаря-терапевта приходять пацієнти двох типів: 1) ті, що мають медичну картку на руках, і час їхнього приходу розподілено рівномірно на інтервалі 12±7 хвилин; 2) ті, що прийшли на прийом у перший раз (не мають медичної картки), час їхнього приходу розподілено рівномірно на інтервалі 30±3 хвилини. Час прийому пацієнтів першого типу – 15±3 хвилин, а другого типу – 26±9 хвилин. Необхідно змоделювати роботу лікаря протягом чотирьох годин.

2. У бібліотеку приходять читачі двох типів: ті, що прийшли в бібліотеку в перший раз і повторно. Прихід читачів першого типу розподілено рівномірно на інтервалі 25±3 хвилини, другого – 35±15 хвилин. Час роботи з читачами першого типу – 20±10 хвилин, другого типу – 13±8 хвилин. Модель роботи бібліотекаря має забезпечити збирання статистики про черги. Необхідно змоделювати роботу бібліотекаря протягом шести годин.

3. У квиткову касу аеропорту приходять пасажири двох типів: першого типу – купують авіаквитки, другого типу – міняють наявні авіаквитки. Прихід пасажирів першого типу розподілено рівномірно на інтервалі 5±3 хвилини, прихід пасажирів другого типу так само розподілено рівномірно на інтервалі 25±20 хвилин. Час обслуговування пасажирів першого типу – 7±5 хвилин, а другого – 14±6 хвилин. Модель роботи квиткової каси аеропорту має забезпечити збирання статистики про черги. Необхідно змоделювати роботу каси протягом восьми годин.

4. У пункт обміну валюти приходять клієнти двох типів: 1) ті, що бажають купити валюту; прихід таких клієнтів розподілено рівномірно на інтервалі 15±8 хвилин; 2) ті, що бажають здати одну валюту і купити іншу; їх прихід розподілено рівномірно на інтервалі 55±25 хвилин. Час обслуговування клієнтів першого типу – 10±3 хвилини, другого типу –15±8 хвилин. Модель роботи обмінного пункту має забезпечити збирання статистики про черги. Необхідно змоделювати роботу пункту протягом чотирьох годин.

5. На пошту з одним вікном для прийому телеграм приходять клієнти двох типів: 1) дати телеграму в межах країни, прихід таких клієнтів розподілено рівномірно на інтервалі 5±4 хвилини; 2) дати телеграму за кордон, прихід таких клієнтів розподілено рівномірно на інтервалі 55±25 хвилин. Час прийому телеграм у клієнтів першого типу – 5±3 хвилини, другого типу – 10±6 хвилин. Модель роботи вікна прийому телеграм має забезпечити збирання статистики про черги. Необхідно змоделювати роботу вікна прийому телеграм протягом 12 годин.

3 МОДЕЛЮВАННЯ БАГАТОКАНАЛЬНИХ ПРИСТРОЇВ

3.1 Моделювання багатоканальних пристроїв у GPSS

У системі GPSS є можливість моделювати однорідні паралельні пристрої за допомогою спеціальних засобів, які називаються багатоканальними пристроями (БКП) або пам'ятями. Для введення в програму пам'яті використовується блок **STORAGE**, а зміна стану пам'яті проводиться блоками **ENTER** і **LEAVE**.

Формат команди для визначення пам'яті:

Name STORAGE A,

де Name – мітка об'єкта;

A – загальна ємність.

Пам'ять можна перевизначити введенням нової команди **STORAGE** з тим же ім'ям.

Для того, щоб зайняти певну кількість блоків пам'яті, в GPSS використовується блок **ENTER**. Коли транзакт намагається увійти в блок **ENTER**, його потреба в пам'яті порівнюється з доступним об'ємом пам'яті, і транзакт або займає блок, або чекає звільнення певної кількості елементів пам'яті. Формат команди:

ENTER A[, B],

де A – ім'я або номер пам'яті;

B – кількість елементів, на які зменшується доступна ємність пам'яті (за замовчуванням дорівнює 1).

Коли транзакт входить у блок **ENTER**, операнд A обчислюється і використовується для пошуку пам'яті з таким номером. Якщо така пам'ять не існує, то відбувається зупинка за помилкою. Операнд B використовується для визначення необхідного обсягу пам'яті.

Якщо пам'ять знаходиться в доступному стані та існує достатня кількість елементів для задоволення потреби транзакту, то йому дозволяється увійти в блок **ENTER**, і його потреба задовольняється зменшенням поточної кількості вільних елементів на число необхідних. В іншому випадку транзакт поміщається для очікування в список затримки пам'яті в порядку пріоритету.

Коли транзакт входить у блок **ENTER**, інтерпретатор виконує такі дії:

1) збільшує лічильник входів БКП на значення операнда B;
2) збільшує поточний вміст БКП на значення операнда B;
3) зменшує доступну ємність БКП на значення операнда B.

Для звільнення елементів пам'яті в GPSS використовується блок **LEAVE**. Формат команди:

LEAVE A,B,

де A – ім'я або номер пам'яті;
B – кількість елементів пам'яті.

Коли транзакт входить у блок **LEAVE**, інтерпретатор виконує зворотні дії:

1) зменшує поточний зміст БКП на значення операнда B;
2) збільшує доступну ємність БКП на значення операнда B.

Для збирання статистики про використання БКП в стандартному звіті GPSS виводяться такі характеристики:

– STORAGES – номер або ім'я БКП;
– CAP. – ємність БКП, обумовлена командою STORAGE;
– REMAIN – кількість одиниць вільної ємності БКП у кінці періоду моделювання;
– MIN – мінімальна кількість використовуваної ємності БКП за період моделювання;
– MAX – максимальна кількість використовуваної ємності БКП за період моделювання;
– ENTRIES – кількість входів у БКП за період моделювання;
– AVEL. – середнє значення зайнятої ємності за період моделювання;
– UTIL – середній коефіцієнт використання всіх пристроїв БКП;
– DELAY – кількість транзактів, які очікують можливості входу в блок **ENTER**.

Приклад 3.1. На овочеву базу прибувають два типи вантажівок – постачальників і споживачів. Для розвантаження вантажівки постачальників необхідно мати три вантажника і 60±10 хвилин. Вантажівки постачальників прибувають кожні 40±20 хвилин. Для завантаження вантажівки споживачів необхідно мати одного вантажника і 3±1 хвилини. Вантажівки споживачів прибувають кожні 35±5 хвилин. Отримання та відвантаження овочів здійснюється на різних пунктах. На овочевій базі працюють всього шість вантажників. Промоделювати роботу овочевої бази протягом доби.

Програму мовою GPSS, що моделює роботу овочевої бази, показано на рисунку 3.1, фрагмент звіту за результатами моделювання – на рисунку 3.2.

Зі звіту про використання пам’яті LOADER видно, що ємність пам’яті дорівнює шести, на момент закінчення моделювання три елементи пам’яті були вільними, протягом терміну моделювання середнє значення зайнятої ємності дорівнювало 4,444, коефіцієнт використання пристроїв пам’яті – 0,741, а транзактів, що очікують звільнення пристроїв пам’яті за час моделювання, не було. За допомогою рисунку 3.2 можна ознайомитись зі статистикою перебування транзактів у черзі на розвантаження вантажівок споживачів (CONSUMER) і постачальників (SHIPPER).

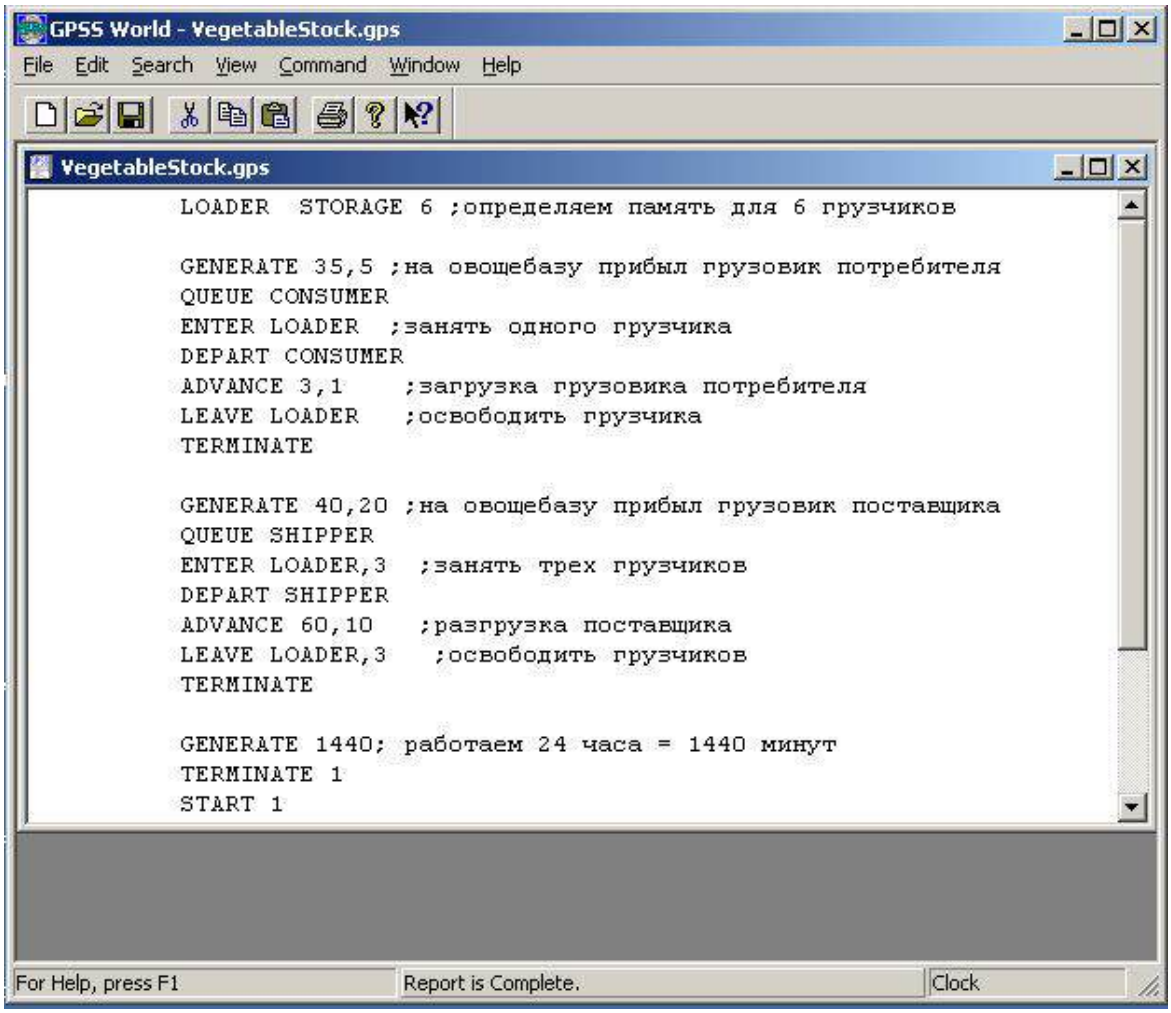


Рисунок 3.1 – Текст програми, що моделює роботу овочевої бази

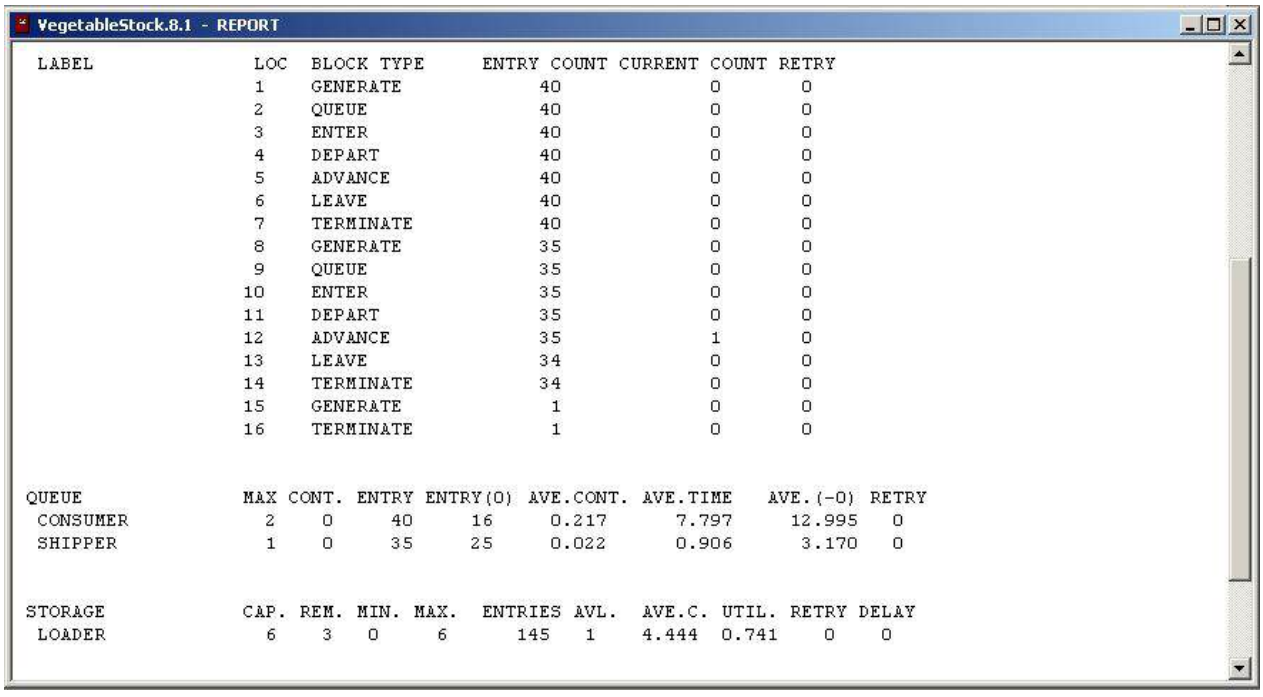


Рисунок 3.2 – Фрагмент звіту за результатами моделювання

3.2 Моделювання багатоканальних пристроїв, що використовують ресурси, в системі AnyLogic

У реальних системах часто зустрічаються ситуації, коли для оброблення заявки, що надійшла, необхідними є ресурси певного типу. Такими ресурсами можуть бути як об’єкти, що відновлюються, так і об’єкти, що не відновлюються. Ресурсами, що відновлюються, можуть бути, наприклад, вантажники на складі, комірки пам’яті в запам’ятовуючому пристрої. Ресурси, що не відновлюються, використовуються один раз для забезпечення однієї певної заявки, наприклад, напівфабрикати на конвеєрній лінії.

Для моделювання таких систем у бібліотеці **EnterpriseLibrary** середовища AnyLogic є об’єкти *ResourcePool*, *Seize*, *Release* та *Service*.

Об’єкт *ResourcePool* зберігає задану кількість об'єктів певного типу. Кількість об'єктів можна задавати безпосередньо *(Directly)* або за часом *(Bytableovertime)*.

Об’єкт *Seize* при отриманні заявки захоплює задану кількість ресурсів *(Resourcequantity)* і передає заявку далі.

Об’єкт *Release* звільнює задану кількість ресурсів при отриманні заявки.

Якщо між захопленням і звільненням ресурсів необхідно відпрацювати заявку, то ці об’єкти необхідно пов’язати з об’єктом *Delay* і, за необхідності, з об’єктом *Queue*. Ці чотири об’єкти (*Seize*, *Queue*, *Delay* і *Release*) можна замінити одним – *Service*. Цей об’єкт при отриманні заявки захоплює необхідну кількість ресурсів, потім обробляє заявку заданий час, потім звільняє ресурси і передає заявку далі. При цьому об’єкт *Service* має вбудовану чергу заданої місткості. Для ресурсів, що не відновлюються, цей об’єкт не використовується.

Приклад 3.2. На овочеву базу прибувають два типи вантажівок – постачальників і споживачів. Для розвантаження вантажівки постачальників необхідно мати три вантажника і 60±10 хвилин. Вантажівки постачальників прибувають кожні 60±20 хвилин. Для завантаження вантажівки споживачів необхідно мати одного вантажника і 3±1 хвилини. Вантажівки споживачів прибувають кожні 35±5 хвилин. Отримання та відвантаження овочів здійснюється на різних пунктах. На овочевій базі працюють всього шість вантажників. Необхідно промоделювати роботу овочевої бази протягом доби.

У системі є два типи заявок – постачальників і споживачів. Оскільки оброблення заявок здійснюється в різних пунктах, необхідно створити два генератори заявок і два механізми, які їх обробляють.

Модель буде мати такий вигляд, як показано на рисунку 3.3.

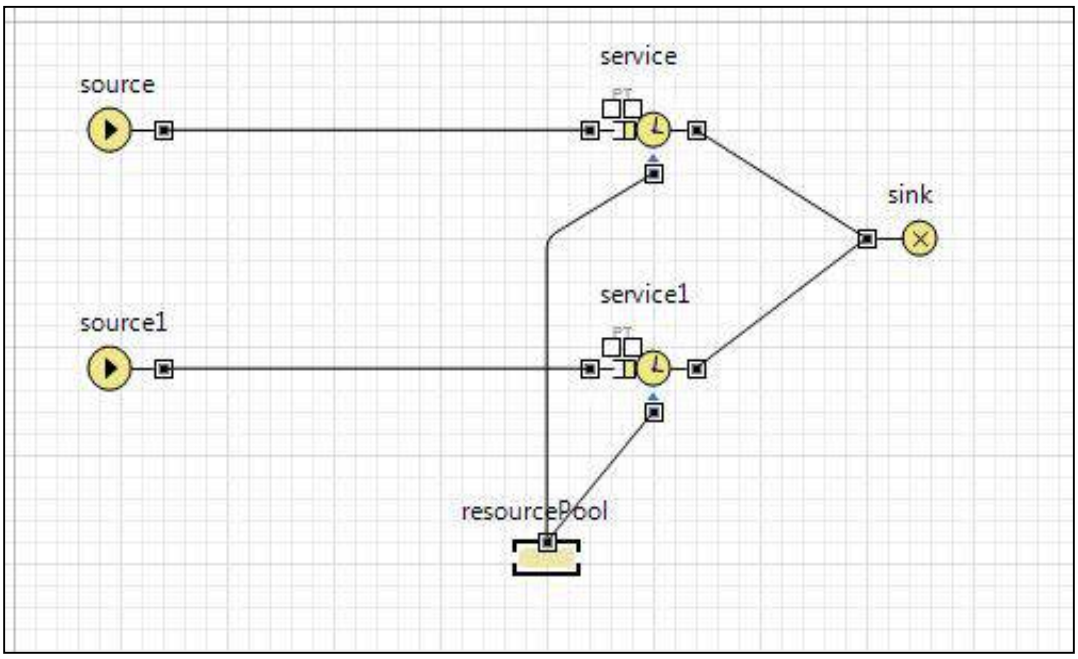


Рисунок 3.3 – Схема моделі

Властивості об’єктів (в англійській версії AnyLogic) є такими:

1. *Source:*
   1) *Заявки надходять згідно з (Arrivals defined by)*: *Часом між прибуттями (Interval time);*
   2) *Час між прибуттями (Interval time)*: uniform_discr(40, 80);
2. *Source1*:
   1) *Заявки надходять згідно з (Arrivals defined by): Часом між прибуттями (Interval time);*
   2) *Час між прибуттями (Interval time):* uniform_discr(30, 40).
3. *Resourcepool*:
   1) *Місткість (Capacity): 6.*
4. *Service:*
   1) *Кількість ресурсів (resource quantity)*: 3;
   2) *Час затримки (Delay time)*: uniform_discr(50, 70);
   3) *ResourcePool object*: recourcePool.
5. *Service1:*
   1) *Кількість ресурсів (resource quantity)*: 3;
   2) *Час затримки (Delay time)*: uniform_discr(20, 40);
   3) *ResourcePool object*: recourcePool.

Властивість **StopTime** на вкладці *Модельний час* (*ModelTime)* об’єкта *Моделювання: Main (Simulation*: *Main)* необхідно встановити такою, що дорівнює 86400, а **Timeunits** – *seconds*.

Після роботи моделі в статистиці об’єктів можна побачити коефіцієнт їхнього використання (попередньо встановивши властивість *Показувати статистику* (*Enablestatistic*) у true) (рисунок 3.4).

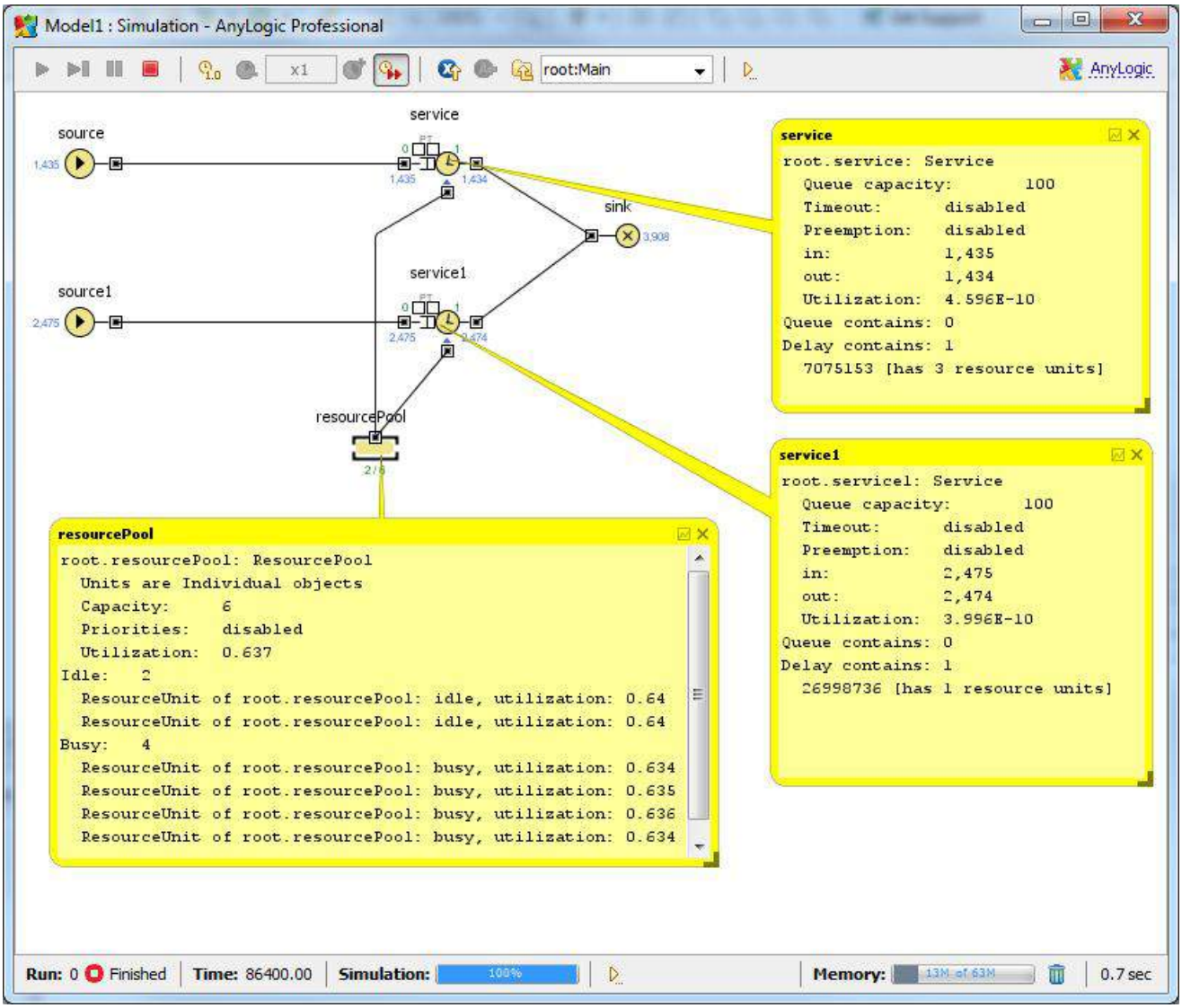


Рисунок 3.4 – Статистика об’єктів після моделювання

3.3 Варіанти завдань для самостійної роботи

1. Відвідувачі приходять до закусочної кожні 10±5 секунд. У закусочній є два сектори для обслуговування відвідувачів – в одному пропонують гарячі обіди, а в другому – бутерброди. Сектор для видачі гарячих обідів має шість точок, а для видачі бутербродів – тільки одну. Видача гарячої страви триває одну хвилину, а бутерброда – 30 секунд. Відвідувачі заходять до закусочної за гарячою стравою кожні 5±2 хвилини, а за бутербродами – кожні 40±10 секунд. Потрібно зібрати статистику про черги до обох секторів для 1000 відвідувачів.

2. У салоні краси є чоловічий і жіночий зали. Чоловічий зал обладнано трьома кріслами, а жіночий – шістьма. Час обслуговування в чоловічому залі становить 30±20 хвилин, а в жіночому – 80±20 хвилин. Інтервал між відвідуваннями салону жінками становить 30±10 хвилин, а чоловіками – 10±5 хвилин. Необхідно промоделювати роботу салону краси протягом восьми годин.

3. Повідомлення генеруються зі швидкістю 7±3 секунди. Вони мають бути відправлені спочатку однією головною лінією до центру комутації, який потім передає їх на кінцевий пункт призначення за допомогою п'яти окремих ліній. Тільки одне повідомлення може бути на будь-якій лінії в певний момент часу. Довжину повідомлення рівномірно розподілено на інтервалі від 10 до 100 символів. Пересилання даних основним каналом триває 100±10 секунд, допоміжними каналами – 50±10 секунд. Необхідно імітувати передачу 1000 повідомлень і оцінити статистику про черги на обслуговування.

4. У суперкомп'ютері є 500 процесорів. Суперкомп'ютер призначено для розв'язання тільки двох задач – визначення прогнозу погоди на основі інформації з гідрометцентрів та обчислення макроекономічних показників. Для розв'язання першої задачі комп'ютеру необхідно мати 215 процесорів і 30 секунд. Для розв'язання другої – 365 процесорів і 48 секунд. Прогноз погоди необхідно обчислювати кожні 30±10 секунд, а макроекономічні показники – кожні 360±40 секунд. Промоделювати роботу суперкомп'ютера протягом тижня.

5. На заводі вісім конвеєрів виробляють запчастини двох типів – А і Б. Для виготовлення запчастин типу А необхідно 50 секунд і шість конвеєрів, а для виготовлення запчастин типу Б – 15 секунд і три конвеєри. Запчастини типу А виготовляються кожні 60±10 секунд, а типу Б – 20±5 секунд. Промоделювати роботу заводу протягом зміни.

6. У лікарні працюють два лікаря і п'ять медсестер. Під час прийому пацієнтів з кожним лікарем мають працювати три медсестри. До першого лікаря пацієнти приходять кожні 10±1 хвилин, їх обслуговують 5±2 хвилини, до другого – кожні 25±5 хвилин, їх обслуговують 8±2 хвилини. Промоделювати роботу лікарів протягом чотирьох годин.

4 ЗМІНА МАРШРУТУ РУХУ ВИМОГ У МОДЕЛІ

4.1 Зміна маршруту руху вимог у GPSS за допомогою блоків GATE, TRANSFER, TEST

Блок ***TRANSFER*** забезпечує перехід транзакту до нового блока.

Формат блока:

TRANSFER A,B,C,D.

Блок ***TRANSFER*** може функціонувати в одному з дев'яти режимів, кожний з яких має різні характеристики. Коли транзакт входить у блок ***TRANSFER***, операнд А використовується для визначення режиму.

Якщо операнд А не задано, блок ***TRANSFER*** працює в режимі безумовної передачі

TRANSFER , B.

Тут В – номер або мітка блока, до якого має перейти транзакт.

У режимі статистичної передачі блок ***TRANSFER*** використовується у такому форматі:

TRANSFER A,[B],C,

де A – ймовірність передачі транзакту до блока C. Ймовірність може бути задано числом з десятковою крапкою або цілим додатнім числом, що інтерпретується як долі від тисячі;

B – позиція блока, до якого має перейти транзакт (з ймовірністю 1 – A);

C – позиція блока, до якого має перейти транзакт (з імовірністю A).

Якщо операнд B не задано, то транзакт переходить з імовірністю 1 – A до наступного блока.

Якщо в операнді A вказано зарезервоване слово **BOTH**, то блок ***TRANSFER*** працює в режимі **BOTH**.

У цьому режимі транзакт, що входить, спочатку намагається перейти до блока, вказаного в операнді B. Якщо це не вдається, транзакт намагається перейти до блока, вказаного в операнді C. Якщо транзакт не зможе перейти в жоден блок, він залишається в ***TRANSFER*** і при кожному перегляді списку поточних подій буде повторювати в тому ж порядку спроби переходу, доки не зможе вийти із цього блока. Формат блока:

TRANSFER BOTH, B, C.

Якщо в операнді A вказано зарезервоване слово **ALL**, то блок ***TRANSFER*** працює в режимі **ALL**. Формат блока:

TRANSFER ALL, B[, C[, D].

Якщо цей блок не може прийняти активний транзакт, то послідовно перевіряються всі блоки, доки не буде досягнуто блока, що задано в операнді C, або один із перевірених блоків не прийме транзакт. Місцезнаходження кожного блока, що послідовно перевіряється, обчислюється шляхом додавання операнда D до місцезнаходження раніше перевіреного блока. Якщо операнд D не використовується, перевіряється кожний блок між блоками, вказаними в операндах B і C. Якщо не використовується операнд C, перевіряється тільки один блок. Якщо жоден із блоків не може прийняти транзакт, він залишається в ***TRANSFER***, доки не зможе увійти до одного з блоків.

Якщо в операнді A вказано зарезервоване слово **PICK**, то блок ***TRANSFER*** працює в режимі **PICK**. Формат блока:

TRANSFER PICK, B, C.

Коли транзакт входить у цей блок, місцезнаходження нового блока вибирається випадково в числовому інтервалі між B і C.

Якщо в операнді B вказано ім’я функції, блок ***TRANSFER*** працює у функціональному режимі, при цьому місце призначення активного

транзакту вибирають шляхом обчислення функції, вказаної в другому операнді, з додаванням до цього значення необов'язкового прирощення, вказаного в операнді С. Формат блока:

TRANSFER PICK, FunctName [,C].

Блок ***GATE*** змінює маршрут руху транзактів залежно від стану деякого об'єкта. Формат блока:

GATE O A,B,

де O – умовний оператор;
A – ім'я об'єкта, що перевіряється;
B – номер блока призначення у випадку невиконання умови тесту.

Блок ***GATE*** працює або в «режимі відмови» (при цьому операнд B не використовується), або в режимі «альтернативного виходу» (операнд B задано). Коли транзакт намагається увійти до блока ***GATE***, що працює в режимі відмови, і умова тесту не виконується, транзакт блокується.

Якщо тест проходить успішно, активний транзакт входить до блока ***GATE***, а потім переходить до наступного блока. Заблоковані транзакти розміщуються у списку повторних спроб об'єкта, що перевіряється. Коли стан будь-якого з об'єктів змінюється, заблокований транзакт знову активізується, повторюється тест, заданий блоком ***GATE***, і, якщо умова виконується, транзакту дозволяється увійти до блока ***GATE***.

Якщо блок ***GATE*** працює в режимі альтернативного виходу (задано операнд B) і транзакт намагається увійти до блока, а умова тесту не виконується, транзакт прямує до альтернативного блока призначення, заданого операндом B.

Якщо тест успішний, активний транзакт входить до блока ***GATE***, а потім переходить до наступного блока.

Умовні оператори блока ***GATE*** наведено в таблиці 4.1.

Таблиця 4.1 – Умовні оператори блока ***GATE***

Умовний оператор	***Опис умови виконання тесту***
FNV	Пристрій, указаний в операнді A, має бути недоступним
FV	Пристрій має бути доступним
NI	Пристрій, указаний в операнді A, не має бути перерваним у цей момент
I	Пристрій має бути перерваним у цей момент
NU	Пристрій не використовується
U	Пристрій використовується
SE	Пам'ять, що вказана в операнді A, має бути порожньою

Продовження таблиці 4.1

Умовний оператор	***Опис умови виконання тесту***
SF	Пам'ять має бути заповненою
SNE	Пам'ять не має бути порожньою
SNF	Пам'ять не має бути заповненою
SNV	Пам'ять не має бути доступною
SV	Пам'ять має бути доступною
LS	Логічний ключ, заданий в операнді А, встановлено у положення «увімкнено»
LR	Логічний ключ, заданий в операнді А, встановлено у положення «вимкнено»
M	Блок MATCH, заданий в операнді А, має містити транзакт, що очікує умови синхронізації
NM	Блок MATCH, заданий в операнді А, не має містити транзакт, що очікує умови синхронізації

Блок ***TEST*** порівнює значення і керує місцем призначення активного транзакту на основі результату порівняння. ***TEST*** змінює маршрут руху транзактів залежно від стану деякого об’єкта. Формат блока:

TEST O A, B, C,

де O – оператор відношення;
A – значення, що перевіряється;
B – контрольне значення;
C – номер блока призначення.

Блок ***TEST*** функціонує в режимі відмови і в режимі альтернативного виходу. В будь-якому випадку операнди А і В обчислюються і порівнюються. Якщо операнд С не використовується, блок ***TEST*** функціонує в режимі відмови, інакше – в режимі альтернативного виходу.

Якщо транзакт намагається увійти до блока ***TEST***, що працює в режимі відмови, і вказана умова не виконується, транзакт блокується, тобто йому не дозволено увійти до блока, доки умову не буде виконано. Після виконання заданої умови активний транзакт входить до блока ***TEST*** і потім переходить до наступного блока.

Якщо операнд С використовується, блок ***TEST*** функціонує в режимі альтернативного виходу. Коли транзакт намагається увійти до блока і умова не виконується, йому призначається альтернативний блок, вказаний в операнді С. Якщо умова виконується, активний транзакт входить до блока ***TEST*** і потім переходить до наступного блока.

Оператори відношення блока ***TEST*** наведено в таблиці 4.2.

Таблиця 4.2 – Оператори відношення блока ***TEST***

Оператор відношення	***Опис умови виконання оператора***
E	Операнд А має дорівнювати операнду В
G	Операнд А має бути більше операнда В
GE	Операнд А має бути більше або дорівнювати операнду В
L	Операнд А має бути менше операнда В
LE	Операнд А має бути менше або дорівнювати операнду В
NE	Операнд А не має дорівнювати операнду В

Приклад 4.1. На складі є місце для зберігання 500 ящиків гуталіну. Щоденний попит на товар коливається між 10 і 50 ящиками. Для поповнення запасів на складі раз на тиждень оформлюють замовлення, за яким кількість ящиків на складі має бути доведено до 500. Якщо кількість товару на момент замовлення становить 400 ящиків або більше, замовлення не виконується. Термін доставки гуталіну за замовленням – три дні. Треба промоделювати роботу складу протягом одного місяця і визначити, чи утвориться на складі дефіцит.

Текст програми, що моделює роботу складу, показано на рисунку 4.1.

Stock.gps

```
STOCK      STORAGE 500; вместимость склада
Stock_Table TABLE S$Stock,10,10,50; S$Stock - объем памяти Stock, занятой транзактами
           ORDER_QUALITY VARIABLE Target-S$STOCK; Объем заказа
           DEMAND     VARIABLE RN1@100+20; Делим значение RN1 по модулю 24 и прибавляем 40
           Target     EQU 500; Начальное число товаров
           Reorder    EQU 400;Граница заказа

           GENERATE 5,,,,1; каждые 5 единиц модельного времени создается 1 транзакт
           TEST L S$STOCK,Reorder,Skip; Если количество товаров на складе не меньше 400,
                                      ; ничего не заказываем
           ASSIGN 2,V$Orderqty ;Параметр 2 равен объему заказа
CustWait   ADVANCE 3 ;Время доставки 3 дня
           ENTER Stock,P2; Количество товаров увеличивается на P2
           TERMINATE
Skip       TERMINATE; заказ завершен

           GENERATE 1
           ASSIGN 1,V$Demand ;Параметр один равен дневному потреблению товаров
           TABULATE Stock_Table; Табулируем количество товаров в этот день
           TEST GE S$Stock,P1,Stockout; Если количество товаров на складе меньше, чем
                                      ; дневной расход, то отказываем в выдаче
                                      ; и переходим в блок Stockout

           TERMINATE 1 ;Дневной таймер
Stockout   TERMINATE 1; вход в этот блок происходит только в том случае, если есть дефицит товара.
                     ;  Количество случаев дефицита можно узнать с помощью СЧА N$Stockout

           GENERATE ,,,1,10 ; Создает единственный транзакт с приоритетом 10 для инициализации параметров склада
           ENTER Stock,Target
           TERMINATE

          GENERATE 20
          TERMINATE 1
          START 1
```

Рисунок 4.1 – Текст програми, що моделює роботу складу

4.2 Зміна маршруту руху вимог в AnyLogic

Бібліотека **EnterpriseLibrary** містить об'єкт *SelectOutput*, за допомогою якого можна керувати потоком заявок залежно від умови. Об'єкт має два виходи – true і false. Заявка, що надійшла на об'єкт, переходить на вихід true із заданою ймовірністю або за істинності умови.

Середовище AnyLogic також містить інші механізми керування виконанням моделювання: змінні, події, функції. У змінних можна зберігати будь-яку інформацію, яка є доступною протягом усієї симуляції. Події мають виконувати програмний код у визначені моменти часу із заданою періодичністю або при виконанні певної умови. Функції нічим не відрізняються від аналогічних конструкцій будь-якими мовами програмування.

Будь-який об'єкт із бібліотеки **EnterpriseLibrary** – це екземпляр певного класу. Під час симуляції можна змінювати стан об'єктів, викликаючи їхні методи. Наприклад, об'єкт *Source* має метод **inject(n)**, виклик якого створює *n* заявок.

Такий механізм керування об'єктами зручно застосовувати для моделювання складних систем, поведінка яких залежить від певних умов.

Приклад 4.2. На складі є місце для зберігання 500 ящиків гуталіну. Щоденний попит на товар коливається між 10 і 50 ящиками. Для поповнення запасів на складі раз на тиждень оформлюють замовлення, за яким кількість ящиків на складі має бути доведено до 500. Якщо кількість товару на складі на момент замовлення складає 400 ящиків або більше, замовлення не виконується. Термін доставки гуталіну за замовленням – три дні. Треба промоделювати роботу складу протягом одного місяця і визначити, чи утвориться на складі дефіцит.

Кількість ящиків гуталіну зручно зберігати у змінній. Для додавання змінної до моделі достатньо пересунути об'єкт *Проста змінна (Простая переменная)* в робочу область. Змінній необхідно задати ім'я, тип і початкове значення (рисунок 4.2).

Для з'ясування наявності дефіциту необхідно додати ще одну змінну з ім'ям isDeficite типом boolean і початковим значенням false.

Для моделювання попиту товару необхідно додати об'єкт *Source* з такими властивостями: *Заявки надходять згідно з/ Часом між прибуттями* – одна одиниця часу; *Кількість заявок, що надходять за один раз*, – uniform_discr(10, 50); *Дія при виході* – Gutaline--. Об'єкт *Source* необхідно зв'язати з об'єктом *Sink*.

Для моделювання поповнення запасів слід використати об'єкти *Source*, *SelectOutput*, *Delay* і *Sink*. Об'єкт *Source* має генерувати 500-Gutaline заявок кожні сім днів. В об'єкті *SelectOutput* необхідно вказати, що вихід true вибирається при виконанні умови Gutaline < 400.

Вихід true об’єкта *SelectOutput* слід поєднати з об’єктом *Delay*, а вихід false – з об’єктом *Sink*. Об’єкту *Delay* призначити час затримки три одиниці часу. Дія при виході – Gutaline++. На рисунку 4.3 зображено робочу область моделі.

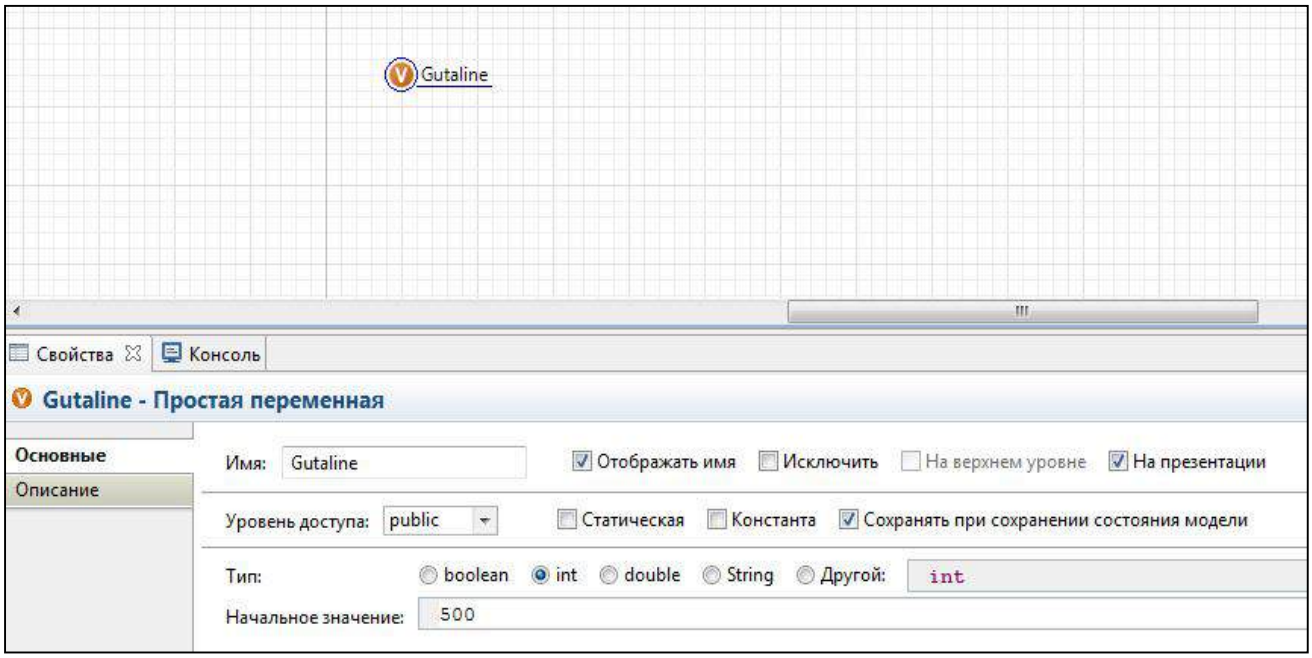


Рисунок 4.2 – Додавання змінної

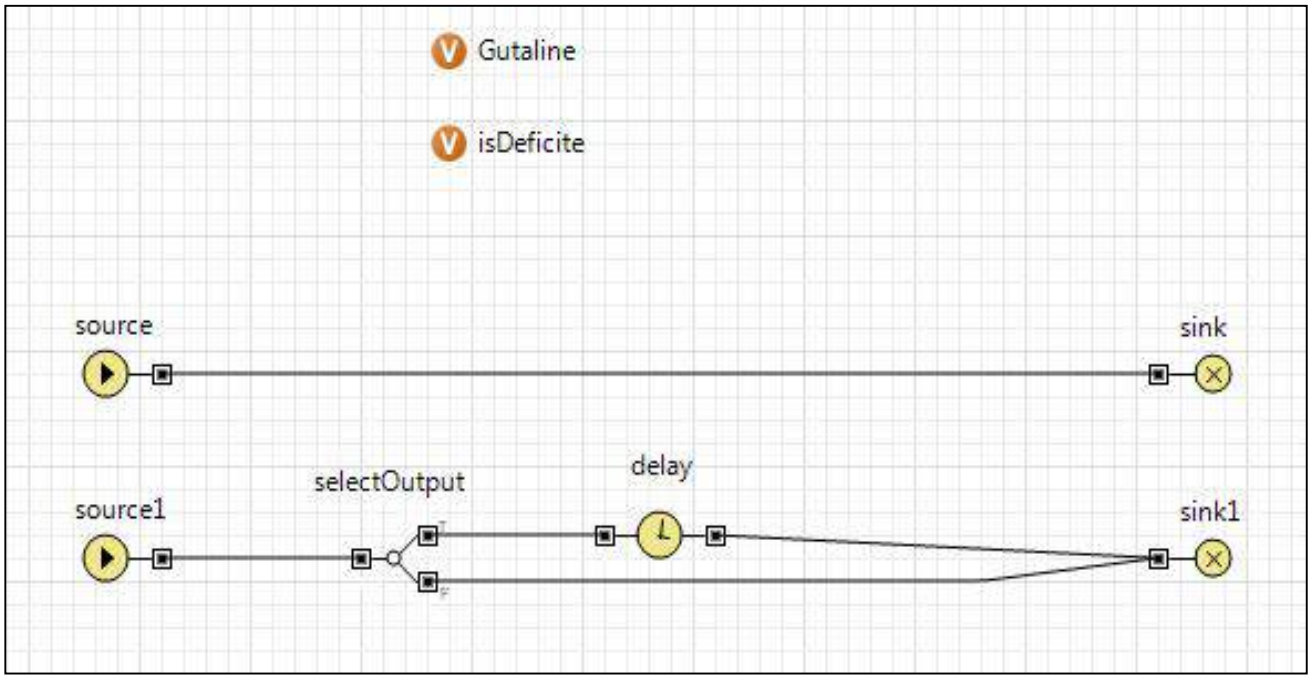


Рисунок 4.3 – Робоча область моделі

Установлюючи одиниці модельного часу у днях, а кінцевий час у 30 днів, можна виконувати моделювання. На рисунку 4.4 показано результат роботи моделі.

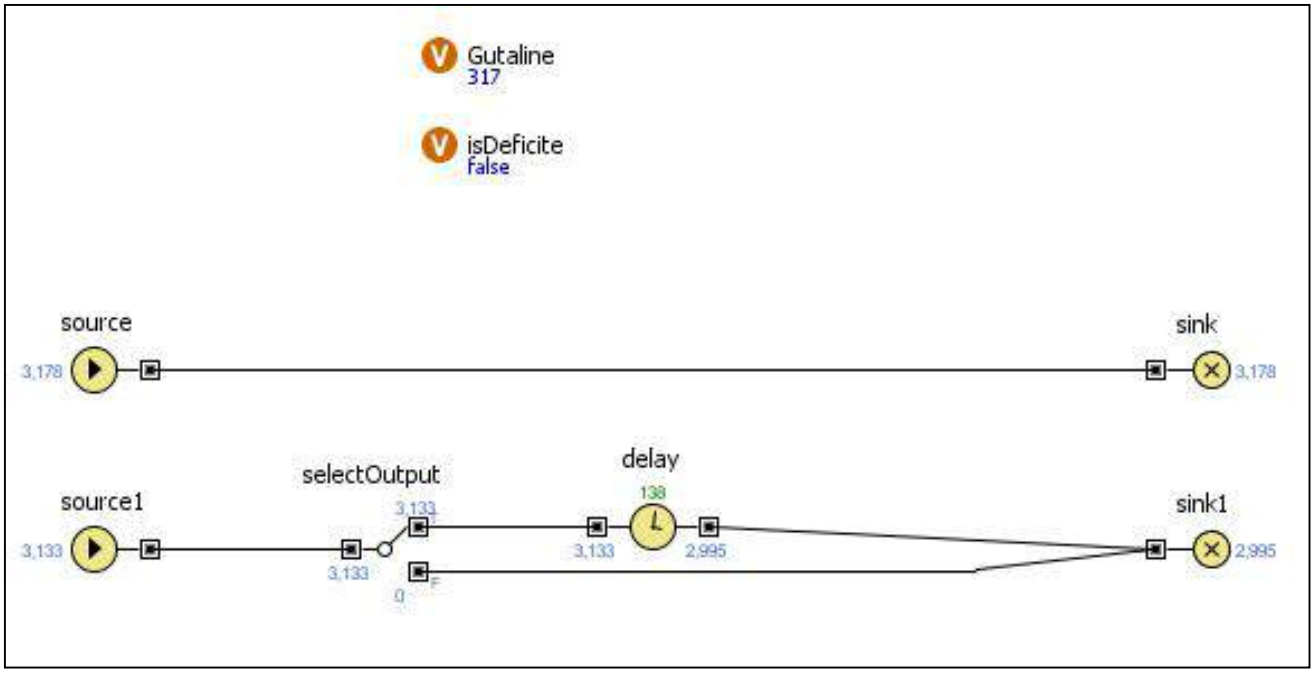


Рисунок 4.4 – Результат роботи моделі

Як видно зі значення змінної isDeficite, дефіциту не утворюється.

БІБЛІОГРАФІЧНИЙ СПИСОК

Литвинов, В. В. Методы построения имитационных систем [Текст] / В. В. Литвинов, Т. П. Марянович. – Киев: Наук. думка, 1991. – 120 с.

Томашевский, В. Н. Решение практических задач методами компьютерного моделирования [Текст]: учеб. пособие/ В. Н. Томашевский, Е. Г. Жданова, А. А. Жолдаков. – Киев: НАУ, 2001. – 268 с.

Сахирова, Н. П. Страхование [Текст]: учеб. пособие/ Н. П. Сахирова. – М.: ТК Велби, 2006. – 744 с.

Томашевский, В. Н. Имитационное моделирование средствами системы GPSS/PC [Текст]: учеб. пособие/ В. Н. Томашевский, Е. Г. Жданова. – Киев: І3МН, НТТУ КПИ, 1998. – 123 с.

Шрайбер, Т. Дж. Моделирование на GPSS [Текст] / Т. Дж. Шрайбер – М.: Машиностроение, 1980. – 593 с.

Карпов, Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 [Текст]/ Ю. Г. Карпов. – СПб.: БХВ-Петербург, 2006. – 400 с.

ЗМІСТ

Навчальне видання

Бакуменко Ніна Станіславівна
Трофимова Ірина Олексіївна
Хайленко Маргарита Олександрівна

ІМІТАЦІЙНО-ПОДІЙНЕ МОДЕЛЮВАННЯ З ВИКОРИСТАННЯМ СИСТЕМ GPSS І ANYLOGIC

Редактор В. М. Коваль

Зв. план, 2017
Підписано до видання 18.07.2017
Ум. друк. арк. 2,3. Обл.-вид. арк. 2,56. Електронний ресурс

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції сер. ДК № 391 від 30.03.2001