УДК 004.01

СИСТЕМА ВІДДАЛЕНОГО УПРАВЛІННЯ РОБОТОТЕХНІКОЮ З МОДУЛЬНИМ ТИПОМ ПОВЕДІНКИ НА ПЛАТФОРМІ WEBOTS

М. В. Коломієць, Т. А. Мартьянова, Л. В. Загоруйко,

Анотація. Сучасна галузь робототехніки вимагає постійного розвитку штучного інтелекту. Саме алгоритм поведінки робота визначає його продуктивність та призначення. Аби оцінити ефективність логіки управління, потрібно виконати ряд тестувань. Тому метою курсової роботи стало створити середовище для тестування користувацьких модулів керування поведінкою роботів. Концепція програми, що розробляється, лежить в створені системи випробовування функціональності робота, яка зв'язується із середовищем симуляції, обробляє та зберігає інформацію з датчиків та сенсорів і формує подальшу поведінку робота відповідно до вбудованого модулю управління. Для створення повноцінного навколишнього середовища обрано симулятор Webots. Зв'язок із середовищем встановлюється за протоколом TCP/IP. Сервером виступає клієнтський додаток у якому виконуються обчислення, робота з базою даних, впровадження нових модулів, а клієнтом – сесія симуляції у Webots.

Ключові слова: управління, робот, робототехніка, поведінка, модуль.

Стрімкий розвиток галузі робототехніки, зумовлює зростання якості технічних характеристик та вимагає постійної модернізації рішень управління. Актуальність й призначення робота визначається його поведінкою та алгоритмом дій. Сьогоднішнє суспільство потребує автоматизації рішень в найрізноманітніших сферах діяльності: від зондів-дослідників та хірургічних роботів до автоматизованих машин доставки товарів. Кожен такий винахід перед початком використання проходить ряд тестувань. Особливо детального вивчення потребує штучний інтелект майбутнього приладу.

Контролери роботів зменшуються в габаритах і стають більш доступними для інженерів і розробників. На сьогоднішній день програмовані логічні контролери (Programmable Logic Controller, PLC), контролери для автоматизації (Programmable Automation Controller, PAC), промислові комп'ютери та модульні контролери, які випускаються сторонніми виробниками, можуть використовуватися для керування роботами. [1]

Компанії розробки програмного забезпечення постають перед задачею тестування спроектованого контролеру для його подальшого впровадження. Тестування на реальних прототипах роботів не є раціональним підходом, так як потребує значних матеріальних затрат та не менш цінного ресурсу – часу.

Вирішенням задачі оптимізації процесу стало використання симуляторів. Комп'ютерні середовища моделювання дозволяють детально зімітувати поведінку робота і його оточення. Це дає можливість провести повне тестування виконавчих пристроїв (інструментів і маніпуляторів), елементів безпеки в критичних ситуаціях, виконання завдань і місій в спеціальних умовах, яке б засвідчувало правильність проектування і функціонування робота ще до його впровадження або придбання. Однак, існують і недоліки приведених програмних продуктів. Відповідно до висновків НДЦ. Програма наукової роботи ставить на меті втілення цих рекомендацій для абстрагування розробників від специфіки налаштування симуляторів, впровадження роботів у середовище й програмного коду в контролер об'єкту управління. Додаток стане посередником між віртуальним середовищем та користувачем. Проектований графічний інтерфейс матиме інтуїтивно зрозумілі команди, можливість впроваджувати сторонні модулі управління та роботу з базою даних для експорту даних.

Головна ідея проектованого програмного забезпечення, полягає у створені платформи, яка комунікує з середовищем симуляції за принципом «обмін повідомленнями» та дозволяє віддалене керування роботом за користувацьким алгоритмом управління. В результаті отримуємо систему з інструментами обробки та збереження інформації від

робота та його оточення, засобами роботи з графічною обробкою даних та функціоналом вбудовування власних модулів керування.

Принцип функціонування полягає в співпраці трьох підрозділів системи: симулятор Webots, програма користувача, сторонній модуль управління.

Реалізація такого процесу буде виконуватись за допомогою пакету ClassLoader. System Classloader – системний завантажувач, реалізований уже на рівні JRE. Цією бібліотекою завантажуються класи, шляхи до яких вказані в змінній оточення CLASSPATH.

За мову програмування було обрано Java. Насамперед цю мову обрано через її кросплатформність. Незалежність від платформи на якій виконуються програми, це дійсно одна з основних переваг мови, адже один і той же код можна запускати під управлінням операційних систем Windows, Solaris, Linux, Machintosh та ін. Це необхідно, коли програми завантажуються через Інтернет для подальшого виконання під управлінням різних операційних систем. Крім того, Java – повністю об'єктно-орієнтована мова. Всі сутності в мові Java є об'єктами, за винятком небагатьох основних типів (primitive types), наприклад чисел. (За допомогою Об'єктно-орієнтованого програмування легко розробляти складні проекти.) Java володіє високою надійністю. Творці забезпечили мову Java засобами, що дозволяють виключити саму можливість створювати програми, в яких були б приховані найбільш поширені помилки.

Програма наукової роботи була написана на мові Java в середовищі InteliJIDEA з дотриманням вимог ООП, використовуючи патерни проектування та сторонні бібліотеки.

В проекті втілено два патерни проектування: MVC для побудови моделі взаємозв'язку інтерфейсу та функціоналу, та DAO для ефективного управління базою даних. MVC архітектура в структурі програмного проекту представлена трьома пакетами (packadge): Model несе в собі логіку додатку, View – форми графічного інтерфейсу, та Controller – функціонал обробки дій користувача. Описана модель продемонстрована на рисунку 2.3.1. [7]

Для імплементації інтерфейсу було використано бібліотеку SWING. За допомогою SWING створено три екрані форми: головне вікно додатку і автентифікація та реєстрація користувача. Використано різноманітні компоненти й контейнери, зокрема: JScrollPanel, JTextPane, JButton, JLabel, JTextField, JRadioButton, JPanel. Головне вікно складається з трьох робочих зон розділених компонентом JPanel. Перша зона призначення для відображення основної інформації симуляції. Друга для вибору логіки управління роботом. Третя зона визначена для інформаційних повідомлень користувачеві про стан системи.

Для втілення можливості клієнт серверної архітектури за протоколом TCP/IP використано бібліотеку Java – net.Socket. Щоб реалізувати головний інструмент програмного забезпечення – динамічне завантаження сторонніх модулів управління – імпортовано бібліотеки io.File та io.FileInputStream й розширено клас CLassLoader. Аби полегшити математичні обчислення, в клас стандартного набору модулів поведінки програми, імпортовано бібліотеку «Math». Зокрема, у визначені подальшого руху робота використовувались функції піднесення до степеню, отримання значеня косинусу, синусу, тангенсу, арктангенсу.

Для роботи з базами даних в науковій роботі, було використано стандарт JDBC. JDBC представляє собою опис інтерфейсів і деяких класів, які дозволяють працювати з базами даних для Java. Головним принципом архітектури є універсальний спосіб спілкування з різними базами даних. Самі SQL-запити відрізняються за рахунок різного набору функцій для дат, рядків і т. д. Алгоритм і набір команд для доставки запиту на SQL-сервер і отримання даних від SQL-сервера не відрізняються.

Для першої групи використовується метод інтерфейсу Statement – executeQuery (). Він покриває дуже великий відсоток запитів, наприклад для отримання даних з таблиці. Для другої групи запитів використовується інший метод інтерфейсу Statement – executeUpdate(). На відміну від executeQuery() (який повертає ResultSet) цей метод повертає ціле число, яке повідомляє скільки рядків у таблиці було змінено під час виконання запиту. Інтерфейс

Statement впроваджений для роботи простих запитів. PreparedStatement використаний там, де потрібні параметри з запитом. Запит такого типу дозволяє зробити дві речі: заздалегідь підготувати запит із зазначенням місць, де будуть підставлятися параметри; встановити параметри певного типу і виконати після цього запит з уже встановленими параметрами. [8]

Використовуючи Data Access Object (DAO) для абстрагування і інкапсулювання доступу до джерела даних було створено компоненти з простим інтерфейсом, що надають доступ об'єктам DAO до СУБД. DAO повністю приховує деталі проектування джерела даних від користувачів. Оскільки при змінах реалізації, джерелам даних надається незмінний DAO інтерфейс, цей патерн дає можливість приймати різні схеми сховищ без впливу на компоненти ПЗ. По суті, виконує функцію адаптера між компонентом і джерелом даних [9].

Аби налаштувати підключення до спроектованої бази даних, в класі MySqlDaoFactory було визначено атрибути для підключення користувача. Приватне поле «user» містить зареєстроване раніше в системі ім'я користувача, «password» пароль підключення, «url» адрес локального хоста з портом 3306, «driver» несе інформацію про драйвер. Метод getContext() виконує підключення за вказаними параметрами до СУБД.

Алгоритм реалізації запитів до бази даних втілюється в методах які несуть в собі синтаксис sql запитів, та процеси заповнення аргументів цих запитів. Прикладом може стати запит до таблиці «Session». Доступ до CRUD операцій імплементований в класі «MySqlSessionDao». Рисунок 2.3.4 описує запит на оновлення даних по ідентифікатору в MySQL якби це виглядало в консолі, при звичайній роботі із СУБД. Лістинг демонструє підстановку, в параметри раніше написаного запиту, атрибутів що необхідно передати.

Так як програмний продукт наукової роботи проектувався за архітектурою MVC патерну, тому в своїй основі має три пакети Model, View, Controller.

Схема пакету Model представлена на рисунку 1 Ця частина проекту містить всю бізнес-логіку програми та вважається найбільш незалежною від інших. У ній міститься код, який опрацьовує запити до бази даних, алгоритми управління роботом та впровадження сторонніх модулів управління.

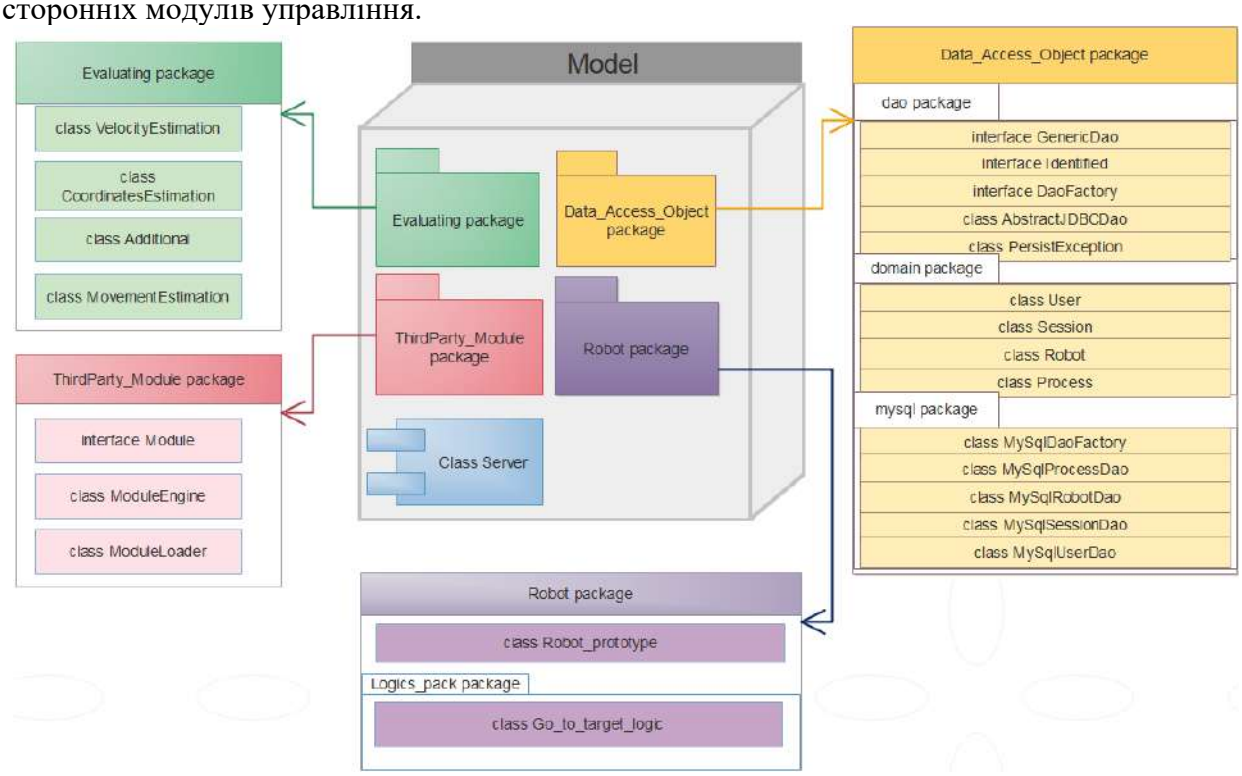

Рис. 1 Діаграма пакету «Model»

Пакет класів Robot містит об'єкт самого робота. Атрибутами класу Robot_prototype стали радіус колеса, відстань між колесами, максимальна швидкість, поточна позиція

об’єкту управління, координати його цілі й остання зафіксована кутова швидкість. Окрім властивостей усіх приватних полів, в класі робота визначені методи для руху робота: обрахування швидкості на основі направленого до цілі вектору («move») й обчислення переміщення пристрою в системі координат («update_movement_indicators»).

Всі частини пов’язані між собою протоколом TCP/IP. Саме цей протокол характеризується як механізм з попереднім встановленням з’єднання, що забезпечує надійність потоку даних, дає впевненість у безпомилковості одержуваних даних і усуває дублювання даних. TCP дозволяє регулювати навантаження на мережу, а також зменшувати час очікування даних під час передачі на великі відстані. Більш того, TCP гарантує, що отримані дані були відправлені точно в такій же послідовності.

Така точність процесу передачі даних необхідна для подальших розрахунків переміщення робототехнічного пристрою в просторі симуляції.

Представлена на рисунку 2 діаграма потоку даних описує зовнішні по відношенню до системи джерела і адресати інформації, логічні операції, потоки і сховища даних, до яких здійснюється доступ. Такий структурний аналіз демонструє, звідки формуються дані для зберігання й момент їх внесення до бази, та процеси для яких необхідно ці відомості дістати із сховища. Для зображення такого процесу обрано модифіковану нотацію Йордана. Коло означає процес, прямокутник світлого тону – зовнішню сутність, темного – сховище даних, направляючі вказівники – потік даних.

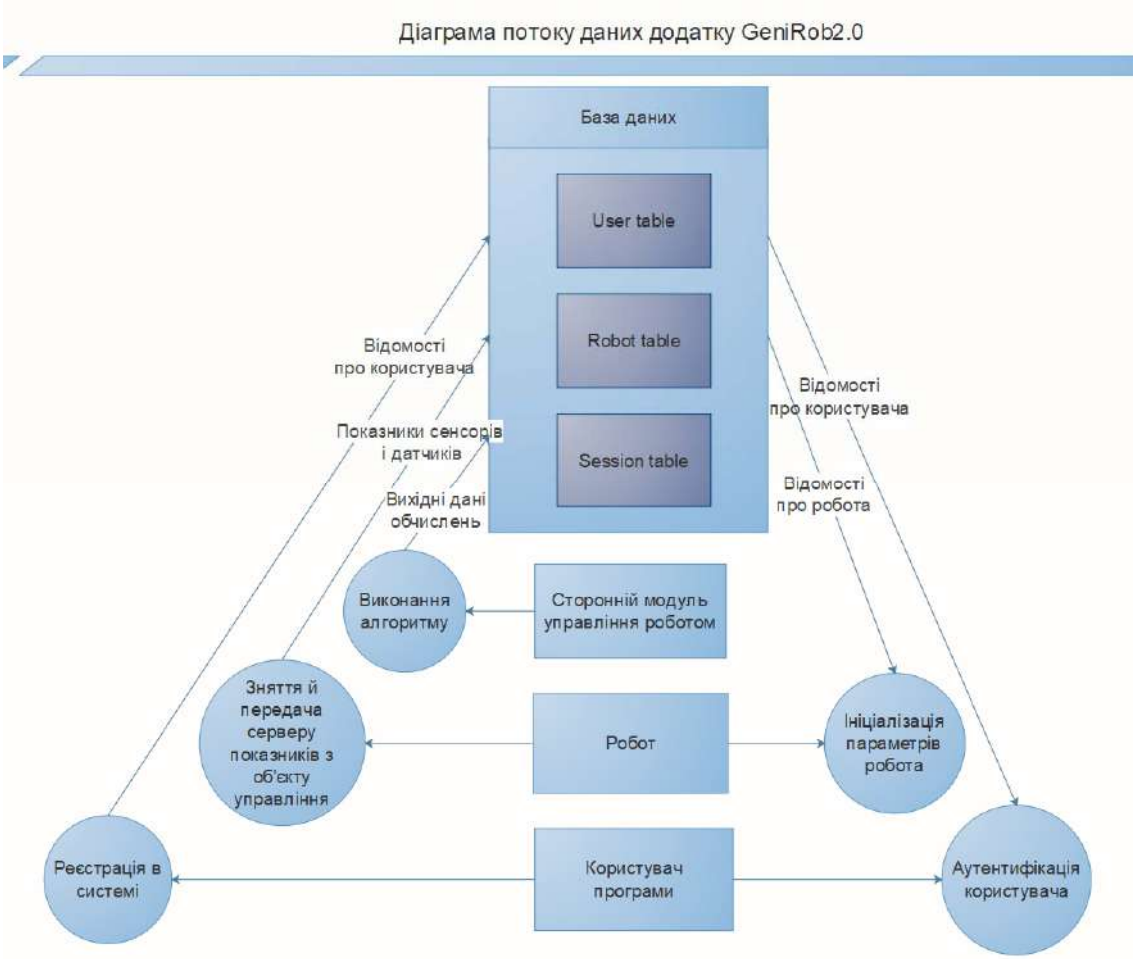

Рис. 2. Діаграма потоку даних додатку

Відповідно до діаграми нижче визначено, коли користувач реєструється в системі, потік введених даних направляється в базу даних. Під час процесу автентифікації відомості користувачів витягуються із сховища. Наступний потік даних стосується робота і його показників. Джерелом процесу «Зняття й передача серверу відомостей» виступає об’єкт управління, потік даних якого направлений в базу даних на збереження. Зворотна

процедура – ініціалізація параметрів робототехнічного пристрою на початку програми. Останній потік, що розглядається, несе вихідні дані обчислень процесу виконання алгоритму управління до бази даних. Таким чином до сховища додається результат виконання програми – команди для формування поведінки робота.

Програмне забезпечення призначене для контролю і управління роботом в симуляції, запису інформації активності і її аналіз. Програма може бути використана для віддаленого контролю робототехнікою, збереження даних переміщення і відомостей з датчиків для подальшого аналізу матеріалу.

Система призначена для використання інженерами ВНЗ в якості навчання і тестування власних винаходів. Інженерами в галузі робототехніки. Передбачається середній рівень комп'ютерної підготовки користувача, розумінням базових принципів функціонування симуляцій середовищ. Також необхідно володіти знаннями в геометрії та фізики для повного розуміння обчислень і якісного аналізу обробки даних.

На стадії тестування та впровадження системи управління, внутрішні лабораторії Донецького університету використовували програмне забезпечення для оцінки ефективності власних алгоритмів управління чотирьох колісною робототехнікою. Студенти та лаборанти мали змогу тестувати власну логіку управління в спроектованому середовищі симуляції, вбудовувати алгоритм керування «мозком» робота, оцінювати ефективність проходження маршрутів і траєкторій, уникання перешкод і досягнення цілі(виконання завдання тесту) в цілому.

Для того щоб повноцінно використовувати ПЗ було інстальовано середовище Webots та створено проект робота яким планується керувати. При встановлені проекту обиралась модель робота з диференціальним приводом від Khepera. Цей контрольний приклад використовувався при проектуванні додатку, й продемонстрував безперебійну роботу під час тестувань. Створену симуляцію зберігали у файл після чого систему перезавантажували. Після автентифікації, в головному додатку, обирали логіку керування роботом зі списку встановлених зовнішніх модулів, або із запропонованих стандартних алгоритмів що постачається разом із ПЗ. Старт сесії симуляції починався після запуску серверу який очікує з'єднання з клієнтським підключенням. Коли сервер й клієнт обмінювались повідомленнями – робот починав свій рух/дії, що демонструвалось в головному вікні симуляції. Результати досліджень використовувались при вдосконалені власних рішень та під час тестування власних розробок логіки керування.

Ще однією областю застосування ПЗ наукової роботи, стала необхідність змоделювати рух об'єктів управління для розрахунку завантаженості транспортного потоку через пропускні пункти. Відповідно сторонній модуль управління імпортувався в клієнтський додаток і з заданими параметрами відбувалась імітація різного роду транспортних пригод. Результати враховувались під час модифікації окремих програмних рішень аудиторського ПЗ для логістичної компанії.

На перспективу планується розширити набір робототехніки який би включав не тільки колісних роботів, а й крокуючі, плавальні, маніпуляційні девайси, по типу: робот-рука. В подальшому, задля оптимізації процесів обчислення планується впровадження технології багатопоточності. Таке рішення дозволить запускати декілька симуляцій в середовищі одночасно, що збільшить швидкість тестування алгоритмів модулів управління. Одне з головних завдань стане модифікація відображення вихідної інформації графічними засобами, аби мати змогу промальовувати карти маршруту, зберігати зображення статистичних відомостей, графіків та діаграм.

Висновки. Розвиток робототехніки зумовлює постійну модифікацію рішень управління. Завдання автоматизації та створення нових, оптимальних алгоритмів керування лягає на науковців та розробників програмного забезпечення. З очевидних причин, не завжди є можливість протестувати щойно винайдений метод поведінки на реальних приладах. Більше того, використання симуляторів не завжди являється оптимальним виходом, так як несуть в собі певну складність налаштування й беруть час до адаптації.

Головним завданням стало створення зручного посередника між користувачем й симуляцією.

За допомогою мови Java, бібліотек управління базами даних, модулів створення TCP/IP зв'язку й динамічного завантаження класів, спроектовано систему яка керує двоколісним роботом віддалено. Таке рішення дозволяє абстрагуватись від специфіки робототехнічних пристроїв та зосередитись на тестуванні ефективності методів управління.

Не менш значущою функцією є робота з базою даних. Користувач може зберігати відомості симуляції, робота, своїх сесій та значень приладів, аби мати повне уявлення про свою роботу й виконувати глибший аналіз на основі співставлення даних за різний період.

В подальшому планується розширити набір підтримуваних робототехнічних пристроїв. На сьогоднішній день, додаток дозволяє керувати двоколісним роботом з диференціальним приводом. В зв'язку з тим, що попит на тип чотирьох колісну чи гусеничну робототехніку збільшується, то пріоритетним завданням є втілення цих типів в майбутньому. Наступним кроком буде створення можливості виконання програми в багатопотоковому режимі, що дозволить запускати декілька симуляцій одночасно і оптимізувати процес обчислення.

Аннотация. Современная отрасль робототехники требует постоянного развития искусственного интеллекта. Именно алгоритм поведения робота определяет его производительность и назначения. Чтобы оценить эффективность логики управления, нужно выполнить ряд тестов. Поэтому целью курсовой работы стало создать среду для тестирования пользовательских модулей управления поведением роботов. Концепция программы, разрабатываемой лежит в создании системы испытания функциональности работа, связывается со средой симуляции, обрабатывает и хранит информацию с датчиков и сенсоров и формирует дальнейшее поведение работа в соответствии с встроенного модуля управления. Для создания полноценного окружающей среды избран симулятор Webots. Связь со средой устанавливается по протоколу TCP / IP. Сервером выступает клиентское приложение в котором выполняются вычисления, работа с базой данных, внедрение новых модулей, а клиентом – сессия симуляции в Webots

Ключевые слова: управление, робот, робототехника, поведение, модуль.

Abstract. The modern branch of robotics requires the constant development of artificial intelligence. It is the algorithm of robot behavior that determines its performance and purpose. To evaluate the effectiveness of control logic, a number of tests are required. Therefore, the aim of the course work was to create an environment for testing custom modules for controlling the behavior of robots. The concept of the developed program lies in the created systems of testing the functionality of the robot, which communicates with the simulation environment, processes and stores information from sensors and sensors and forms the further behavior of the robot according to the built-in control module. The Webots simulator has been chosen to create a full-fledged environment. The connection to the environment is established via the TCP / IP protocol. The server is a client application in which calculations, database work, implementation of new modules are performed, and the client is a simulation session in Webots

Key words: control, robot, robotics, behavior, module.

СПИСОК ЛІТЕРАТУРИ

1. «Промислові роботи: тренди й типи». URL: https://www.controlengrussia.com/innovatsii/promy-e-roboty-trendy-i-tipy/
2. Робот. URL: https://uk.wikipedia.org/ wiki/Робот с
3. Webots. URL: https://en.wikipedia.org/ wiki/Webots
4. Динамічне завантаження класів. URL: https://habr.com/ru/sandbox/62803/
5. Loading a Java Class at Runtime. URL: https://stackabuse.com/loading-a-java-class-at-runtime
6. Шаблони проектування – MVC Pattern. URL: https://coderlessons.com/java-tekhnologi/shablony-proektirovania-mvc
7. Шаблон DAO в JAVA. URL: https://www.codeflow.site/ru/article/java-dao-pattern
8. Java. Реалізація шаблону DAO. URL: https://www.dokwork.ru/2014/02/daotalk.html
9. Технічні характеристики та вимоги до системи Windows 10. URL: https://www.microsoft.com/uk-ua/windows/windows-10-specifications
10. Розгляд засобів імітаційного моделювання роботів. URL: https://cyberleninka.ru/article/n/obzor-sredstv-imitatsionnogo-modelirovaniya
11. Why MySQL. URL: https://www.mysql.com/why-mysql/
12. All About Sockets. URL: https://docs.oracle.com/javase/tutorial/networking/sockets/