НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет
Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

До захисту допущено:
Завідувач кафедри
_____ Надія БУРАУ
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та системи навігації і керування»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Автоматизована система побудови плоского важільного механізму»

Виконав:
Студент IV курсу, групи ПГ-91
Гальченко Дмитро Анатолійович _____

Керівник:
Доцент кафедри ПСОН, к.т.н., доц. Цибульник С.О. _____

Рецензент:
Доцент,к.т.н., доцент Шевченко В.В. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані технології та системи навігації і керування»

ЗАТВЕРДЖУЮ

Завідувачка кафедри

_____ Н.І. Бурау

«___»_____2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Гальченку Дмитру Анатолійовичу

1. Тема роботи «Автоматизована система побудови плоского важільного механізму», керівник роботи Цибульник Сергій Олексійович, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи 01.06.2023р.

3. Вихідні дані до роботи: а) автоматизована система має давати змогу створювати плоский важільний механізм з наступними ланками: кривошип, коромисло, шатун, повзун, стійка; б) автоматизована система має автоматично проставляти нумерацію ланок, починаючи з кривошипа; в) автоматизована система має давати змогу автоматично проставляти назви шарнірів, починаючи з шарніру кривошипа, який не приєднаний до стійки.

4. Зміст роботи:

1) Огляд стану проблеми.

2) Завдання синтезу та аналізу механізмів.

3) Аналіз засобів та методів автоматизації процесу побудови плоского важільного механізму.

4) Реалізація автоматизованої системи.

5) Аналіз результатів.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): таблиці, графіки, рисунки, тощо.

6. Дата видачі завдання 06.03.2023р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
	Огляд стану проблеми	20.03.2023	
	Завдання синтезу та аналізу механізмів	27.03.2023	
	Аналіз засобів та методів автоматизації процесу побудови плоского важільного механізму	10.04.2023	
	Реалізація автоматизованої системи	01.05.2023	
	Аналіз результатів	15.05.2023	
	Оформлення пояснювальної записки	01.06.2023	

Студент Дмитро ГАЛЬЧЕНКО

Керівник Сергій ЦИБУЛЬНИК

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломної роботи складається з трьох розділів, містить 26 рисунків, 2 таблиці, 19 джерел.

Мета. Мета дипломної роботи полягає у автоматизації побудови плоского важільного механізму, за допомогою сучасних технологій та інструменті, з метою зменшення грошових та трудових витрат на побудову механізму класичними методами.

Основними задачами є створення робочого додатку, який буде мати зручний та простий інтерфейс користувача, можливість розширення функціоналу додатку.

У розділі аналізу предметної області і постановки задачі було розглянуто основну інформацію про теорію машин та механізмів, синтез та аналіз механізму. Було проведено аналіз предметної області та виділення проблеми. Розглянуто альтернативні системи для вирішення даної проблеми. Було зроблено огляд публікацій та статей на тему вирішення альтернативних проблем у суміжних темах.

У розділі огляду інструментів та технологій, та опису програмної реалізації було вибрано мову програмування, інструменти та технології для вирішення поставленої задачі. Розглянуто основну інформацію про вибрані технології, а також зроблено акцент на переваги вибраних технологій. Розглянуто основну інформацію про тестування додатків, а також створено тест план на прикладі розробленої системи.

КЛЮЧОВІ СЛОВА: ДОДАТОК, АВТОМАТИЗОВАНА СИСТЕМА, ПОБУДОВА ПЛОСКОГО ВАЖІЛЬНОГО МЕХАНІЗМУ

ABSTRACT

Structure and scope of work. The explanatory note of the diploma thesis consists of three sections, including 26 figures, 2 tables, and 19 references.

Objective. The objective of the diploma thesis is to automate the construction of a planar lever mechanism using modern technologies and tools, aiming to reduce the financial and labor costs associated with traditional construction methods.

The main tasks include the development of a working application that has a user-friendly and intuitive interface, with the ability to expand the application's functionality.

In the section of domain analysis and problem formulation, the fundamental information on machine and mechanism theory, synthesis, and analysis of mechanisms is discussed. An analysis of the domain and identification of the problem have been conducted, and alternative systems for addressing the problem have been examined. An overview of publications and articles related to solving alternative problems in related topics has been provided.

The section of tools and technologies overview and software implementation description covers the selection of programming language, tools, and technologies to address the defined problem. The essential information about the chosen technologies is presented, emphasizing their advantages. Additionally, an overview of application testing and a test plan for the developed system have been created.

KEYWORDS: APPLICATION, AUTOMATED SYSTEM, PLANAR LEVER MECHANISM CONSTRUCTION.

ЗМІСТ

ВСТУП

В рамках теорії машин і механізмів, автоматизовані системи займають важливе місце, втілюючи сучасні підходи до проектування та функціонування механізмів. Вони представляють собою комплекси, в яких поєднані механічні елементи з електронікою, програмним забезпеченням та сенсорами для досягнення автоматичного контролю, моніторингу та оптимізації роботи механізму.

Автоматизовані системи у теорії машин і механізмів виконують різноманітні завдання, такі як: автоматичне керування, точне позиціонування, синхронізація руху, безпека та ефективність роботи. Вони дозволяють досягати високої точності, надійності та продуктивності в роботі механізмів.

Застосування автоматизованих систем у теорії машин і механізмів охоплює широкий спектр галузей, включаючи: промислове виробництво, робототехніку, автомобільну промисловість, медичну техніку, авіацію та багато інших. Вони допомагають покращувати продуктивність, якість та безпеку роботи механізмів, сприяючи автоматизації процесів та забезпечуючи більш ефективне використання ресурсів.

Таким чином, автоматизовані системи у теорії машин і механізмів є ключовим елементом, що революціонізує підхід до проектування, функціонування та вдосконалення механізмів, сприяючи досягненню більшої ефективності та продуктивності.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Загальна інформація про теорію машин та механізмів

Теорія машин і механізмів є науковою дисципліною, що досліджує будову, кінематику та динаміку механізмів, з метою їх аналізу та синтезу. У теорії машин і механізмів розрізняють дві групи завдань: аналіз механізмів, який включає в себе дослідження їх структурних, кінематичних та динамічних властивостей та синтез механізмів, який полягає у проектуванні механізмів з заданими параметрами для здійснення необхідних рухів. Завдання структурного та кінематичного аналізу механізмів полягає у вивченні теорії побудови механізмів та руху тіл з геометричної точки зору, незалежно від сил, що обумовлюють їх рух. Динамічний аналіз механізмів містить у собі: визначення сил, що діють на тіла, утворюючи механізм та вивчення взаємозв'язку їх рухів. Синтез механізмів має на меті проектування механізмів з урахуванням заданих структурних, кінематичних та динамічних умов.

1.2. Загальна інформація про механізм та його складові. Розгляд понять структурна та кінематична схема механізму

Механізм – це штучно створена система тіл, призначена для отримання необхідного руху одного або декількох тіл. Механізми можуть мати як дуже просту, так і досить складну та різноманітну будову (структуру). Будова механізму визначає такі найважливіші характеристики, як: види здійснюваних рухів, способи їхнього перетворення, формування механізму, тобто з'єднання окремих його частин у єдину

систему, що супроводжується накладенням умов зв’язку. Правильний розподіл у будові механізму сильно визначає його надійну експлуатацію. Тому, при проектуванні потрібно з безлічі різноманітних механізмів вибрати відповідний і правильно підібрати його основні структурні елементи. А для цього насамперед треба знати основні види сучасних механізмів, їх структурні характеристики та закономірності будови [1].

Проектування та дослідження механізму починається зі складання його схеми. Розрізняють структурні та кінематичні схеми. Структурна схема - схема, викреслена з використанням умовних позначень (ГОСТ 2770-68). Кінематична схема - схема, виконана з урахуванням розмірів ланок. Основними елементами механізму є ланки та кінематичні пари.

Ланка є складовою частиною механізму та представляє собою тверде тіло. Ланка є головним елементом механізму. Вони бувають рухомими і нерухомими. Термін "нерухома ланка" відноситься до ланки, яка служить відліком для вивчення руху інших ланок в механізмі. У залежності від механізму, ця нерухома ланка може мати різні назви: у верстатах - станина, у редукторах - корпус, а в автомобілях - кузов або рама.

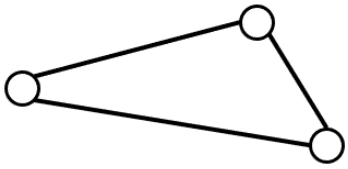
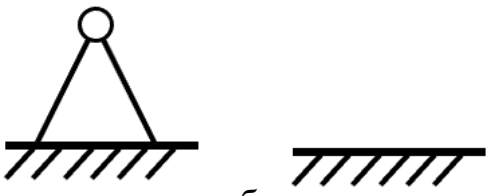
Залежно від характеру руху та деяких конструктивних особливостей розрізняють такі типи ланок:

- Стійка - це нерухома ланка, відносно якої інші ланки вчиняють рух.
- Кривошип - це обертова ланка, яка здійснює повний оберт навколо нерухомої осі.
- Шатун - це ланка з рухомими з'єднаннями лише з іншими рухомими ланками, здійснює плоско паралельний рух;
- Коромисло - це ланка, яка здійснює коливальні рухи при неповному оберті навколо нерухомої осі (наприклад, коромисло клапана).
- Повзун - це ланка, що утворює поступальне рухоме сполучення зі стійкою.

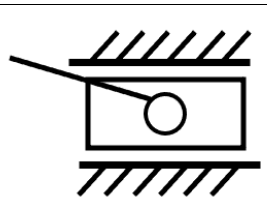
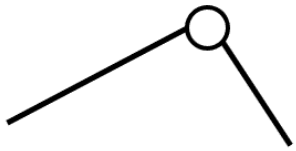
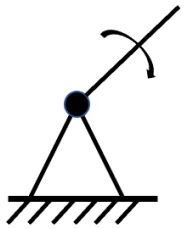
- Куліса - рухома ланка важільного механізму, що здійснює обертальний рух щодо нерухомої осі створюючи поступальну пару з іншою ланкою (кулісним каменем).
- Камінь - це ланка, що утворює поступальне рухоме з'єднання з кулісою.

Ланки також поділяються на дві категорії: вхідні і вихідні. В термінах механізмів і машин, вхідні ланки - це ланки, що призначені для надання заданого руху та відповідних силових факторів (сил або моментів). Вони також називаються ведучими ланками. З іншого боку, вихідні ланки - це ті ланки, на яких одержують необхідний рух та сили. Вони також можуть називатися веденими ланками.

Таблиця 1. Умовні позначення ланок в схемах (ГОСТ 2.770-68)

Об'єкт зображення	Позначення
Важіль, стрижень, вісь, вал тощо.	
Ланка звичайна, шатун	
Ланка складна з трьома кінематичними парами	
Стійка (нерухома ланка) а - для обертальної пари б - для поступальної пари	 а б

Продовження таблиці 1.

Повзун у нерухомих напрямних	
Обертальна пара у площині обертання	
Ланка, що утворює зі стійкою обертальну кінематичну пару	

По розміщенню траєкторій ланок розрізняються: плоскі та просторові. Так як в цій роботі не буде розглянуто просторові, дамо визначення плоским. Плоскі ланки - це ланки, у яких точки ланок описують траєкторії, що лежать в паралельних площинах. Один з таких механізмів - кривошипно-шатунний механізм [2].

Поєднання двох рухомих ланок утворює кінематичну пару. Кінематичною парою називають з'єднання двох сполучних ланок, що допускає їх відносний рух. Ланки цієї пари можуть стикатися один з одним іншим поверхнями, по лінії та точками, елементами кінематичної пари. Якщо елементами є точки або лінії, то пари називають вищими, якщо поверхні – нижчими.

В залежності від типу елементів в кінематичній парі, класифікують вищі та нижчі кінематичні пари. В нижчих КП елементами є поверхні: сферичні, гвинтові, поступальні, обертальні. В вищих КП елементами є точка або лінія. Така класифікація називається за геометричною ознакою, тобто залежно від характеру стикання елементів. Також КП розрізняють від виду відносного руху ланок, що утворюють пару: обертальні та поступальні. В додаток КП класифікуються залежно від замкнення кінематичної пари, а також залежно від кількості накладених на відносний рух зв’язків.

Для кінематичних пар притаманна властивість - число ступенів вільності. Число ступенів вільності (або рухомості) механічної системи відповідає кількості незалежних можливих переміщень системи. Кожне абсолютно тверде тіло має шість ступенів вільності (H=6), тобто шість незалежних параметрів координат, що визначають його положення в просторі (три параметри для обертання навколо осей x, y та z і три параметри для ковзання вздовж цих осей). Це означає, що на тіло не накладено ніяких обмежень у русі. Обмеження, накладені на відносний рух ланок, називаються в'язями S. Якщо ланки утворюють кінематичну пару з іншою ланкою, це накладає певні обмеження на їх відносний рух. Щоб ланки не втратили рухомості повністю, число в'язей повинно бути меншим за 6 і не меншим за 1. Якщо число в'язей S=6, ланки втрачають рухомість, а при S=0 не буде і кінематичної пари.

Таким чином, число ступенів вільності H визначається рівнянням:

$$H = U - S$$

де S – кількість умов зв’язків в кінематичній парі, U – число ступенів свободи ланки.

Так я було визначено основні терміни ТТМ, розглянемо поняття «структурна та кінематична схеми механізму». Структурна схема описує загальні параметри механізму: кількість ланок та кінематичних пар, послідовність, способи з’єднання ланок та види можливих рухів. Кінематична схема механізму будується у вибраному масштабі з дотриманням розмірів і форм, від яких залежить рух ланки. На такій схемі повинно бути визначено все, для того, щоб вивчити рух. Все що не стосується руху, повинно бути вилучено, щоб не ускладнювати креслення.

Ланки на схемах позначають цифрами (як правило): 0 – стійка, 1 – ведуча ланка, а кінематичні пари великими буквами латинського алфавіту. Нище наведено приклад кінематичною схеми.

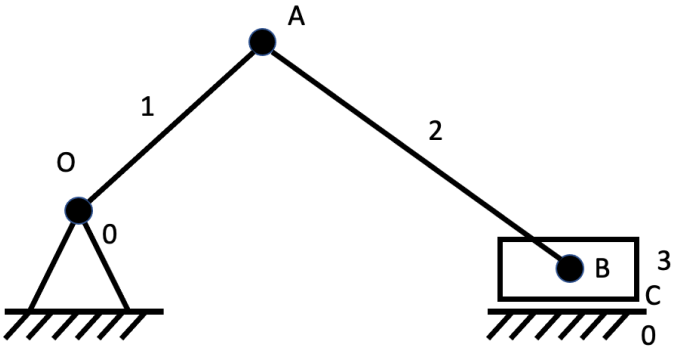


Рис 1.1. Кінематична схема кривошипно-повзунного механізму

На прикладі (Рис 1.1.) на якому зображено кривошипно-повзунний механізм розглянемо позначення механізму. Механізм складається із кривошипа, шатуна, повзуна (поршня) та стійки. Ланка 1 – кривошип, це ведуча ланка; ланка 2 – шатун, це ведена ланка, яка здійснює складний плоско паралельний рух; ланка 3 – повзун, який здійснює поступальний рух відносно стійки.

Кривошип 1 утворює обертальну кінематичну пару зі стійкою 0, це точка О. Шатун 2 утворює обертальну кінематичну пару з кривошипом 1, це точка А. Повзун 3 утворює обертальну кінематичну пару з шатуном 2, це точка В. А також точка С, де повзун 3 утворює поступальну кінематичну пару зі стійкою [1].

1.3. Аналіз та синтез плоских важільних механізмів

Механізм, у якому ланки складаються лише з обертальних та поступальних кінематичних пар, є важільним механізмом. Важільні механізми мають дуже давнє походження, вони мають свої корені в примітивних важелях, які були одними з найдавніших інструментів, використовуваних людством. Однак у складі машин вони з'явилися значно пізніше. В сучасності, важільні механізми знаходять широке застосування у гірничодобувній промисловості, металургії, вібраційній техніці та інших галузях. За допомогою зміни довжини ланок можна отримати механізми, які мають однакову структуру, але відрізняються відтворюваним рухом. Для визначення

основних функцій та характеристик важільних механізмів було створено тридцять моделей механізмів на основі їх структурних схем.

За допомогою моделювання можна проводити структурний аналіз, визначати основні кінематичні характеристики та основну функцію досліджуваного механізму. Основна функція механізму полягає в безрозмірній залежності положення вихідної ланки від узагальненої координати, яка не залежить від довжин ланок, а визначається лише їх співвідношеннями та положенням. Основна функція механізму та її похідні за узагальненим параметром відображають геометричні характеристики механізму, які мають взаємозв'язки з його кінематичними характеристиками. Таким чином, кінематику механізму можна досліджувати у формі кінематичних аналогів, навіть без наявності точного закону руху початкової ланки.

Аналіз та синтез механізму - це дві взаємопов'язані задачі в теорії машин і механізмів.

Синтез механізмів є розділом теорії машин і механізмів, що займається розробкою методів проектування кінематичних схем механізмів на основі заданих кінематичних і динамічних властивостей.

Синтез механізмів складається з двох етапів: структурного синтезу та метричного синтезу. Кожен етап має свої особливі цілі, проте загальними критеріями на обох етапах є мінімізація розмірів, маси та вартості механізму, а також забезпечення технологічності ланок та виконання умов працездатності. Структурний синтез спрямований на вибір оптимальної кінематичної схеми механізму, що відповідає вимогам проекту. На цьому етапі розглядаються різні варіанти структури механізму і здійснюється вибір найбільш відповідного. Метричний синтез, у свою чергу, визначає конкретні параметри ланок механізму (довжина, величина кутів, передавальні числа, тощо), щоб задовольнити вимогам проекту. Таким чином, синтез механізмів включає два етапи з різними цілями, але спрямованих на досягнення оптимальних характеристик механізму з урахуванням вимог проекту.

При структурному синтезі визначається оптимальна структурна схема механізму, яка включає необхідну кількість ступенів вільності, ланки і кінематичні пари для забезпечення потрібних рухів ланок та їх взаємне розташування.

Структурний синтез плоских важільних механізмів базується на структурній класифікації Л.В. Ассура. Під час структурного синтезу встановлюється структурна схема механізму, яка відповідає прийнятим критеріям.

Метричний синтез означає визначення основних геометричних розмірів ланок механізму та конфігурації профілів робочих поверхонь, які найкраще відповідають заданим умовам і забезпечують оптимальне поєднання якісних показників. Цей етап розпочинається з кінематичного синтезу.

Кінематичний синтез є одним з ключових етапів у процесі проектування механізму, оскільки саме на цьому етапі визначаються основні кінематичні характеристики, які необхідні для виконання функцій, покладених на механізм.

В синтезі механізмів виділяються: вхідні та вихідні параметри. Вхідні параметри є відомими на початку синтезу, тоді як вихідні параметри визначаються в результаті синтезу. Наприклад, вхідними параметрами можуть бути траєкторії руху точок механізму, а вихідними - геометричні розміри ланок.

Методи синтезу механізмів можна поділити на дві категорії: прямий синтез (аналітичний, графічний і графоаналітичний) і синтез методами аналізу (оптимальне і автоматизоване проектування). Вибір конкретного методу залежить від поставлених умов проектування. Наприклад, для оцінки відхилення фактичного руху від заданого руху при наближеному виконанні, застосовуються аналітичні методи, оскільки графічні методи не можуть надати повної відповіді на це питання. Результатом метричного синтезу є кінематична схема механізму, яка задовольняє критеріям обох етапів синтезу.

Розглянемо типову задачу синтезу механізму за заданими положеннями ланок. На прикладі кривошипно-повзунного чотири ланкового механізму, розглянемо графічний метод рішення таких задач (цей метод простий та наочний).

Умови задачі:

1. Маємо кінематичну схему механізму з кривошипно-повзунною структурою, в якій траєкторія руху повзуна проходить через центр обертання кривошипа.
2. Параметри кривошипно-повзунного механізму:

a. Величина ходу повзуна (H): Визначає максимальне відхилення повзуна від його початкового положення.

b. Відношення довжин шатуна до кривошипа (λ): Позначається як λ і обчислюється як відношення довжини відрізка АО до довжини відрізка АВ.

Треба визначити: довжину кривошипа АО та довжину шатуна АВ.

Очевидно, що в крайніх положеннях повзуна, кривошип АО і шатун АВ будуть знаходитися вздовж однієї прямої або шатун буде перекриватися з кривошипом.

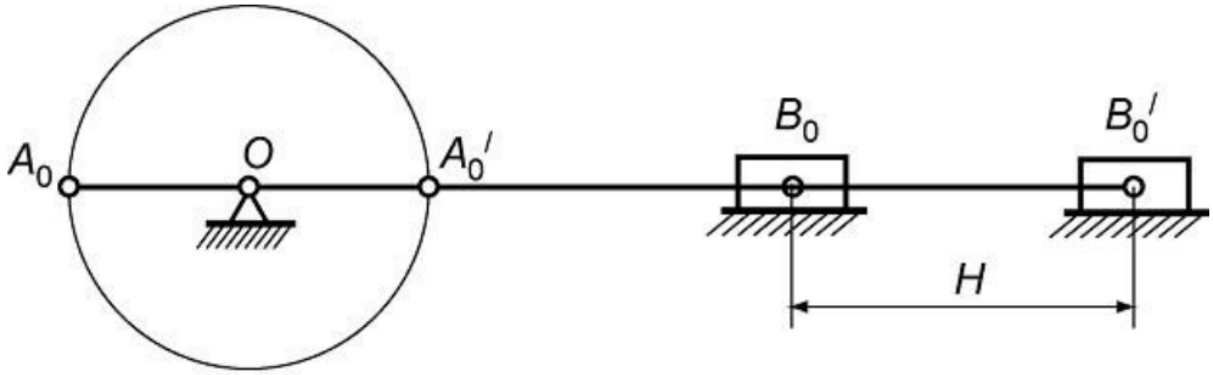


Рис 1.2. Кінематичну схему механізму з кривошипно-повзунною структурою [3]

З (Рис 1.2.) очевидно, що:

$$OB'_O - OB_O = H \tag{1.1}$$

З іншого боку:

$$\begin{cases} OB'_O = AB - OA \\ OB_O = AB - OA \end{cases} \tag{1.2}$$

Підставимо вирази для OB'_O і OB_O в рівняння (1.1), отримаємо:

$$OA = \frac{H}{2}$$

Довжину шатуна АВ визначено як: $AB = \lambda \cdot OA$

Отже, було проведено синтез механізму за допомогою геометричного методу, визначено довжину кривошипу та шатуна, які задовольняють умовам при яких можливе існування механізму, а також зображено кінематичну схему з 2-ма крайніми положеннями механізму [3].

Кінематичний аналіз механізму - це вивчення основних параметрів механізму з метою розуміння законів його руху та вибору найбільш вдалого механізму з існуючих варіантів. В порівнянні з синтезом, кінематичний аналіз широко використовується у практичних дослідженнях. Кінематичний аналіз механізму може бути проведений для конкретного моменту часу або для заданого положення вхідної ланки. Іноді вивчають взаємне розташування ланок механізму у певному положенні.

Цілі кінематичного аналізу:

1. Вивчити кінематичні властивості ланок механізму, такі як: Задаймо умови рух, швидкість, прискорення, траєкторія руху та функція становища, при відомих законах руху вхідних (провідних) ланок.
2. Оцінити кінематичні умови роботи вихідної (робочої) ланки механізму.
3. Визначити необхідні числові дані для проведення розрахунків, пов'язаних з силовими, динамічними, енергетичними та іншими характеристиками механізму.

Вхідні дані: (вкажіть вхідні дані, які використовуються для проведення кінематичного аналізу)

1. Кінематична конфігурація механізму.
2. Геометричні розміри та інші параметри ланок, які залишаються постійними під час руху механізму.
3. Закони руху вхідних ланок (або параметри руху, наприклад, кутова швидкість та кутове прискорення вхідної ланки для обраного положення механізму, що аналізується).

Завдання:

1. Вивчення положення ланок механізму шляхом визначення траєкторій руху точок.
2. Аналіз швидкостей ланок або окремих точок механізму.
3. Визначення прискорень ланок або окремих точок механізму.

Методи:

- Графічний метод (включаючи метод графіків та діаграм) для аналізу механізму.

- Графоаналітичний метод (включаючи метод планів швидкостей та прискорень) для аналізу механізму.
- Аналітичний метод для аналізу механізму.
- Експериментальний метод для аналізу механізму.

Розглянемо графоаналітичний метод. Графоаналітичний метод, також відомий як метод планів швидкостей та прискорень, використовується для кінематичного аналізу механізмів. Завдання, пов'язані з положенням ланок, вирішуються шляхом побудови графічних планів механізму в певному масштабі, що включають кілька накладених один на одного планів. Завдання, що стосуються швидкостей та прискорень, розв'язуються шляхом побудови планів швидкостей та прискорень ланок механізму для заданих положень провідної ланки на основі наперед складених векторних рівнянь швидкостей та прискорень ланок.

Етапи графоаналітичного методу кінематичного аналізу включають наступні кроки:

1. Визначення кінематичної схеми механізму та його параметрів.
2. Вибір заданих положень провідної ланки механізму, для яких будуть проводитись розрахунки швидкостей та прискорень.
3. Складання векторних рівнянь швидкостей та прискорень для кожної ланки механізму на основі аналітичних співвідношень між швидкістю та прискоренням точок ланки.
4. Побудова планів швидкостей та прискорень для заданих положень провідної ланки. Кожен план представляє собою графічне зображення швидкостей або прискорень відповідних точок механізму.
5. Аналіз отриманих планів швидкостей та прискорень з метою вивчення кінематичних характеристик механізму, таких як: швидкість та прискорення точок, залежність швидкості та прискорення від положення провідної ланки тощо.
6. Оцінка кінематичних умов роботи механізму та визначення необхідних числових даних для подальших розрахунків та проектування.

Ці етапи дозволяють отримати детальний кінематичний аналіз механізму з використанням графічних методів та векторних рівнянь [4].

Аналіз та синтез механізму є важливими етапами у розробці нових технологій, виробництві машин та удосконаленні існуючих систем. Вони сприяють досягненню оптимального функціонування механізму, забезпечують його надійність, ефективність та відповідність вимогам сучасного ринку.

1.4. Виділення проблеми

У наш час аналіз та синтез механізмів стикаються з рядом викликів та проблем, які впливають на їх ефективність та результативність. Деякі з цих проблем включають:

1. Складність систем: сучасні механізми стають все більш складними і включають в себе багато різних компонентів та підсистем. Аналіз та синтез таких складних систем можуть бути викликані проблемами зі збором та обробкою великого обсягу даних, а також необхідністю врахування великої кількості параметрів та обмежень.
2. Взаємодія між компонентами: механізми все частіше використовуються як складові частини більших систем, таких як: роботи або автоматизовані виробничі лінії. Аналіз та синтез механізмів у таких випадках потребують врахування взаємодії між різними компонентами та забезпечення їх сумісної роботи.
3. Високі вимоги до продуктивності: сучасні ринкові вимоги ставлять перед механізмами високі стандарти продуктивності, ефективності та точності. Аналіз та синтез механізмів повинні враховувати ці вимоги та забезпечувати високу якість та надійність продукту.
4. Інноваційні технології: впровадження новітніх технологій, таких як: штучний інтелект, комп'ютерне моделювання та аналіз, розширює можливості

аналізу та синтезу механізмів. Однак, виникають нові виклики, пов'язані з розробкою та використанням цих технологій, включаючи необхідність високої експертизи та навичок у використанні.

Аналіз та синтез механізмів є важливими етапами в теорії машин і механізмів. Однак, у сучасних умовах вони стикаються з рядом викликів та проблем. Складність систем, взаємодія між компонентами, високі вимоги до продуктивності та інноваційні технології є основними факторами, які впливають на ефективність та результативність аналізу та синтезу механізмів.

Для успішного аналізу та синтезу механізмів необхідно використовувати сучасні методи та інструменти, такі як: комп'ютерне моделювання, штучний інтелект та експертні системи. Вони допомагають збирати та обробляти великі обсяги даних, враховувати взаємодію між компонентами, виконувати оптимізацію та забезпечувати високу якість та надійність продукту.

Розвиток аналізу та синтезу механізмів є актуальним завданням у зв'язку з постійним розвитком технологій та зростанням вимог до продуктивності та якості механізмів. Інноваційні підходи та технології, спрямовані на покращення аналізу та синтезу механізмів, відкривають нові можливості для розробки ефективних, надійних та інноваційних механізмів.

Отже, розвиток аналізу та синтезу механізмів вимагає постійного вдосконалення методів, використання новітніх технологій та співпраці між науковцями, інженерами та дослідниками. Це сприятиме подальшому розвитку теорії машин і механізмів та забезпеченню прогресу в інженерній сфері.

1.5. Аналіз наявних автоматизованих систем аналізу та синтезу механізмів

У теорії машин і механізмів існує кілька типів автоматизованих систем, що застосовуються у побудові механізмів. Деякі з них включають:

1. Комп'ютерне проектування механізмів (Computer-Aided Design, CAD): CAD-системи дозволяють інженерам створювати, моделювати і аналізувати механізми на комп'ютері. Вони надають можливість виконувати віртуальне тестування, оптимізацію та візуалізацію механізмів перед їх фізичною реалізацією.
2. Чисельне моделювання та симуляція: ці системи використовуються для математичного моделювання руху механізмів і виконання симуляційних експериментів. Вони дозволяють аналізувати кінематику, динаміку та взаємодію елементів механізмів.
3. Автоматизоване керування (Computer Numerical Control, CNC): ці системи використовуються для автоматичного керування рухом механізмів, зокрема верстатів і робочих станцій. Вони базуються на програмному забезпеченні, яке координує виконання операцій з точністю, швидкістю і повторюваністю.
4. Робототехніка: автоматизовані роботи використовуються для виконання різних завдань у виробничих, логістичних, медичних та інших сферах. Роботи мають програмне керування та датчики, які дозволяють їм взаємодіяти з оточенням і виконувати рухи та дії за заданими алгоритмами.
5. Автоматизоване тестування та випробування: ці системи використовуються для автоматичного тестування та випробування механізмів з метою оцінки їх функціональності, міцності, надійності та виконання вимог стандартів. Ці автоматизовані системи сприяють покращенню ефективності, точності, безпеки та якості побудови механізмів. Вони дозволяють зменшити час

і затрати на проектування, виробництво та валідацію механізмів, а також забезпечити більшу гнучкість і можливість вдосконалення.

Синтез механізмів є важливою складовою процесу проектування, а використання сучасного програмного забезпечення, такого як Solidworks, може значно спростити цей процес.

Solidworks - це потужне програмне забезпечення для 3D-моделювання та проектування, яке надає широкі можливості для створення різноманітних механізмів. З його допомогою можна моделювати та аналізувати рухові властивості механізмів, виконувати кінематичний та динамічний аналіз, визначати параметри руху та залежності між компонентами.

Синтез механізму за допомогою Solidworks включає наступні кроки:

1. Визначення вимог та цілей проекту. Це включає визначення потрібної функціональності механізму, обмежень і вимог до продукту.
2. Створення 3D-моделі механізму. За допомогою Solidworks можна створювати детальні 3D-моделі компонентів механізму, а потім збирати їх в одну збалансовану систему.
3. Встановлення взаємодій та обмежень. Solidworks дозволяє встановлювати різноманітні з'єднання та обмеження між компонентами, такі як: шарніри, кулісні пари, ковзання і т. д.
4. Аналіз руху та властивостей механізму. Solidworks надає інструменти для виконання кінематичного та динамічного аналізу механізму, включаючи вимірювання швидкостей, прискорень, сил та моментів.
5. Оптимізація та вдосконалення механізму. З використанням Solidworks можна виконувати оптимізацію параметрів механізму для досягнення бажаних властивостей, таких як: ефективність, міцність або точність.

Застосування Solidworks для синтезу механізмів дозволяє ефективно проектувати та аналізувати різноманітні механізми перед їх фізичною реалізацією. Це дозволяє збільшити швидкість розробки та забезпечити високу якість та надійність механізмів.

Хоча Solidworks є потужним і корисним інструментом для синтезу механізмів, використання цієї системи також має деякі мінуси:

1. Вартість: Solidworks є комерційним програмним забезпеченням і його ліцензії можуть бути досить дорогими. Це може стати перешкодою для студентів, невеликих компаній або окремих користувачів з обмеженим бюджетом.
2. Складність в освоєнні: Solidworks є потужним і комплексним програмним забезпеченням, і для повного освоєння його можуть знадобитися значні зусилля та час. Вимагається певний рівень навичок та знань для ефективного використання всіх можливостей програми.
3. Обмеження на обладнання: Solidworks вимагає потужного обчислювального обладнання для оптимальної роботи, особливо при обробці складних і великих моделей. Це може бути проблемою для користувачів зі слабкими комп'ютерами або обмеженими ресурсами.
4. Залежність від програмного забезпечення: використання Solidworks пов'язане залежністю від конкретного програмного забезпечення. Це означає, що користувачі можуть бути обмежені функціональністю та сумісністю з іншими програмами або форматами файлів.
5. Потреба в навичках інженерів: хоча Solidworks є потужним інструментом, його ефективне використання вимагає від користувачів певного рівня навичок та знань в галузі механіки та проектування механізмів. Без необхідних навичок інженера може бути складно використовувати всі можливості програми.

Враховуючи ці мінуси, важливо ретельно оцінювати потреби та можливості перед використанням Solidworks або будь-якої іншої системи для синтезу механізмів.

Окрім Solidworks, існує кілька інших систем, які також використовуються для синтезу та аналізу механізмів. Деякі з них включають:

1. Autodesk Inventor: Це інтегроване програмне забезпечення для механічного проектування, яке надає інструменти для моделювання, симуляції та аналізу механізмів. Воно має потужні функції для створення 3D-моделей,

виконання рухових симуляцій та визначення кінематичних характеристик механізмів [5].

2. Siemens NX: Ця система CAD/CAM/CAE також має широкий набір інструментів для моделювання, аналізу та симуляції механізмів. Вона дозволяє проектувати та оптимізувати рухові системи, виконувати динамічний аналіз, визначати кінематичні параметри та проводити інші розрахунки [6].

3. Dassault Systèmes CATIA: Ця комплексна система CAD/CAM/CAE також підтримує синтез та аналіз механізмів. Вона надає інструменти для створення 3D-моделей, виконання кінематичного та динамічного аналізу, оптимізації рухових систем та багато іншого [7].

4. PTC Creo: Ця система CAD/CAM/CAE надає інструменти для проектування, аналізу та симуляції механізмів. Вона має функції для створення 3D-моделей, проведення кінематичного аналізу, оптимізації рухових систем та багато іншого [8].

Незважаючи на широкий функціонал, системи для синтезу та аналізу механізмів такі як: Solidworks, Autodesk Inventor, Siemens NX, CATIA, MSC Adams та PTC Creo, можуть мати деякі недоліки, зокрема:

1. Складність використання: багато з цих систем мають складний інтерфейс та вимагають великої кількості навчання для ефективного використання. Вони можуть бути важкими для освоєння для новачків або користувачів без попереднього досвіду в CAD/CAM/CAE.

2. Вартість: деякі з цих систем є комерційними продуктами і вимагають придбання ліцензій або підписки. Вартість таких програм може бути високою, особливо для індивідуальних користувачів або невеликих підприємств.

3. Потужність обчислювальних ресурсів: використання цих систем часто вимагає потужних обчислювальних ресурсів. Для обробки великих моделей або проведення складних симуляцій може бути необхідне потужне обладнання.

4. Обмежена спеціалізація: кожна система має свої особливості та обмеження. Деякі можуть бути більш спрямовані на певні галузі або типи

механізмів, що може обмежити їх універсальність та застосування в інших областях.

5. Залежність від виробника: використання цих систем пов'язане з виробниками програмного забезпечення. Це означає, що користувачі можуть бути обмежені в оновленнях, підтримці та майбутніх розширеннях, які залежать від рішень виробника.

Існує кілька інструментів для аналізу або синтезу механізмів, які працюють в хмарному середовищі. Ось кілька прикладів таких інструментів:

1. Onshape (https://www.onshape.com/): Onshape є професійною системою комп'ютерного проектування (CAD) в хмарі, яка дозволяє створювати 3D-моделі механізмів, виконувати їх аналіз та синтез прямо у веб-браузері [9].

2. Fusion 360 (https://www.autodesk.com/products/fusion-360/): Fusion 360 - це інтегрована платформа для CAD, CAM та CAE, яка також працює в хмарному середовищі. Вона має інструменти для проектування, аналізу та симуляції механізмів [10].

3. SimScale (https://www.simscale.com/): SimScale - це веб-платформа для чисельного моделювання та симуляції, яка дозволяє аналізувати різні аспекти механізмів, включаючи кінематику та динаміку. Вона надає доступ до потужних інструментів, таких як: скінченно-елементний аналіз (FEA) та обчислення течії [11].

Ці інструменти працюють у хмарному середовищі, що дозволяє зручний доступ до них з будь-якого комп'ютера з Інтернет-підключенням без необхідності встановлення спеціального програмного забезпечення. Вони надають широкий спектр функцій для аналізу та синтезу механізмів, спрощуючи процес проектування та інженерного аналізу.

1.6. Огляд публікацій за темою роботи

У статті під назвою “Моделювання і кінематичний аналіз кривошипна-важільного механізму”, написаній Н.С. Ащепковою, кандидатом технічних наук та доцентом у Дніпропетровському національному університеті імені О. Гончара, розглядається моделювання та кінематичний аналіз важільного механізму. Стаття розглядає задачу кінематичного аналізу важільного механізму, визначає особливості його кінематики та розробляє програмне забезпечення з використанням пакету прикладних програм Mathcad.

Автор зазначає, що промислові роботи широко використовують важільні механізми, що складаються з жорстких з’єднань, з'єднаних кінематичними парами. Синтез та кінематичний аналіз таких механізмів вимагають великого обсягу обчислень, а існуючі інженерні програми не завжди надають достатні можливості для розв'язання подібних задач. Тому, стає актуальною розробка програмного забезпечення, заснованого на універсальних, інструментальних, інформаційних та графічних засобах для кінематичного аналізу механізмів промислових роботів.

При синтезі механізму промислового робота необхідно вибрати структуру механізму, визначити розміри і форми з’єднань, що забезпечують потрібні закони їх руху. Кінематичний аналіз проводиться для кожного варіанта конструкції та всіх можливих законів руху ведучої ланки. Традиційні методи кінематичного аналізу механізмів включають аналітичний, графоаналітичний та графічний підходи. Однак вони потребують аналізу восьми або дванадцяти положень ведучої ланки. Використання пакету Mathcad дозволяє розрахувати координати ланок та особливих точок важільних механізмів при будь-якому куті ведучої ланки.

Система вбудованих функцій Mathcad дозволяє виконувати символьну диференціацію, що дозволяє обчислювати швидкості та прискорення особливих точок важільних механізмів та розраховувати траєкторії руху. Результати цих розрахунків дають змогу оцінити межі досяжності захвату промислового робота та вибрати прийнятну траєкторію руху.

Розроблене програмне забезпечення для пакету прикладних програм Mathcad дозволяє ефективно вирішувати задачі кінематичного аналізу важільного механізму. Розв'язання задач розбивається на прості обчислювальні процедури. Контроль за правильністю розрахунку легко виконувати самостійно на кожному етапі вирішення. Крім того, це програмне забезпечення дозволяє моделювання руху важільного механізму з навантаженням або в режимі холостого ходу. Результати моделювання дозволяють анімацію та імпорт даних.

Наведено обчислювальні приклади, які підтверджують ефективність використання пакету прикладних програм Mathcad для вирішення даного типу задач.

Використання середовища Mathcad дозволяє не тільки виконувати інженерні розрахунки, але й використовувати його при навчанні студентів, надаючи їм вміння та навички розв'язання традиційних інженерних задач з використанням програмного забезпечення [12].

У статі під назвою “Кінематичний аналіз плоского важільного механізму програмними засобами”, написана Олександром Ступаком та Сергієм Цибульником з Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського", присвячена кінематичному аналізу плоского важільного механізму за допомогою програмних засобів.

У статті зазначається, що у наші часи кінематичний аналіз механізмів зазвичай проводиться за допомогою графічних або аналітичних методів. Однак з появою мов програмування і відповідних середовищ розробки стало можливим використання програмних методів для реалізації кінематичного аналізу плоских важільних механізмів. Автори пропонують комбінований графоаналітичний метод, який поєднує аналітичні розрахунки з графічними побудовами та розв'язанням геометричних задач.

Для реалізації цього методу авторами було створено програмне забезпечення на основі мови програмування Processing. Це програмне забезпечення дозволяє проводити кінематичний аналіз простих важільних механізмів.

Стаття визначає основні завдання кінематичного аналізу, такі як: побудова плану положень механізму, плану швидкостей шарнірів та плану прискорень шарнірів. План

положення механізму відображає розташування ланок механізму у певний момент руху. Графічні методи базуються на геометричних побудовах траєкторій руху ланок та швидкостей, та прискорень їх шарнірів. Вони надають наочне уявлення про рух механізму, але вимагають побудови для кожного конкретного положення механізму, що ускладнює отримання загального універсального рішення.

У статті акцентується увага на використанні програмних засобів для кінематичного аналізу механізмів, що дозволяє зручно і швидко отримувати результати. Програмне забезпечення, розроблене авторами, може бути корисним для інженерів та науковців, які займаються дослідженням і проектуванням важільних механізмів.

У даній роботі розглядається використання мови програмування, яка дає можливість побудови планів положень, швидкостей і прискорень плоского важільного механізму. Для цього використовується мова програмування Processing, яка є діалектом мови Java з додатковими командами для роботи з графікою і зовнішніми пристроями.

Processing був розроблений, як мова програмування з 2002 року і має широкий функціонал. Він може бути використаний як для простих проектів, так і для детального дослідження. Мова програмування Processing має синтаксис, схожий на Java, і може бути використана, як введення до програмування на інших мовах.

Використання програмного методу для побудови планів положень, швидкостей і прискорень механізмів має свої переваги і недоліки. Однією з переваг є можливість розвитку навичок програмування студентами при вирішенні задач механіки. Також програмний підхід дозволяє розв'язувати подібні задачі швидше, ніж за допомогою аналітичних методів. За допомогою мови програмування Processing також можна анімувати рух механізму будь-якої складності.

Для побудови планів положень, швидкостей, прискорень і анімації руху механізму використовуються елементи векторної алгебри та аналітичної геометрії. Наприклад, для знаходження координат точок перетину прямих або кола з прямою використовуються відповідні геометричні методи.

У роботі проведено перші ітерації розробки універсального програмного забезпечення для кінематичного аналізу плоских важільних механізм.

Узагальнюючи, стаття розглядає можливості програмного підходу до кінематичного аналізу плоских важільних механізмів. Вона пропонує комбінований графоаналітичний метод та описує програмне забезпечення для реалізації цього методу. Ця стаття доводить актуальність засобів автоматизації для вирішення задач механіки [13].

Розробка власної системи для побудови плоского важільного механізму може мати кілька переваг:

1. Кастомізація: розробка власної системи дозволить налаштувати її під конкретні потреби та вимоги проекту. Ви зможете вбудувати специфічні функціональності та алгоритми, що полегшають побудову та аналіз важільного механізму.

2. Відповідність вимогам: власна система дозволить точно враховувати всі вимоги та обмеження, що стосуються побудови плоского важільного механізму в конкретному проекті. Ви зможете здійснювати точний контроль над параметрами механізму та гарантувати його відповідність заданим критеріям.

3. Інтеграція: розробка власної системи дозволить вам легко інтегрувати її з іншими програмами та інструментами, які ви використовуєте у проектуванні та аналізі механізмів. Це забезпечить зручну та ефективну роботу з механізмом у вигляді інтегрованої системи.

4. Незалежність: розробка власної системи дає вам незалежність від сторонніх розробників та залежності від комерційного програмного забезпечення. Ви будете мати повний контроль над системою та зможете вносити зміни та вдосконалення відповідно до ваших потреб.

5. Навчання та дослідження: розробка власної системи є цікавим та навчальним процесом. Вона надає можливість більш глибоко вивчити принципи роботи важільних механізмів, алгоритми аналізу та синтезу, а також розвивати навички програмування та розробки.

Загалом, розробка власної системи для побудови плоского важільного механізму дозволяє досягти більшої точності, гнучкості та відповідності вимогам проекту, а також надає можливість розширювати та вдосконалювати функціональність залежно від потреб користувача.

РОЗДІЛ 2. ОГЛЯД ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування Java

Під час розробки автоматизованої системи для побудови плоского важільного механізму, була обрана мова програмування Java.

Java, яку спочатку представила компанія Sun Microsystems у 1995 році, є мовою програмування та обчислювальною платформою. За свою історію вона пройшла значний шлях розвитку і стала однією з основних в цифровому світі. Java забезпечує надійну основу для безлічі додатків та сервісів, що домінують у сучасному цифровому ландшафті. Крім того, Java і надалі використовується для створення новаторських продуктів і передових цифрових рішень, які відповідають потребам майбутнього.

Мова Java була обрана так, як вона має багато переваг порівняно з іншими мовами програмування. Нижче описано основні переваги мови програмування Java:

1. Кросплатформовість: Java відома своєю кросплатформовістю, що означає, що додатки, написані на Java, можуть працювати на різних операційних системах (Windows, macOS та Linux), без необхідності великого переписування коду. Це робить Java ідеальним вибором для desktop-додатків, які мають бути доступними на різних платформах
2. Повністю об'єктно-орієнтована: Java є повністю об'єктно-орієнтованою мовою програмування.
3. Багата стандартна бібліотека: Java поставляється з великою стандартною бібліотекою, яка містить безліч готових класів і функцій для різних потреб розробки desktop-додатків. Це включає в себе роботу з графікою, мережами, базами даних, введенням/виведенням, обробкою файлів та багато іншого.

Завдяки цій бібліотеці, ви можете прискорити розробку, використовуючи готові компоненти та функціональні можливості.

4. Потужна віртуальна машина: Java використовує віртуальну машину Java (JVM), яка перекладає Java-код у машинний код, зрозумілий операційній системі. JVM забезпечує багато переваг, включаючи: автоматичне управління пам'яттю, оптимізацію виконання та захист від багатьох типів помилок (витоки пам'яті та переповнення буфера). Це допомагає забезпечити високу продуктивність і надійність вашої системи.

5. Велика спільнота та екосистема: Java має широку спільноту розробників, що означає, що ви можете знайти багато документації, підручників, форумів та інших ресурсів для вирішення своїх питань і проблем. Крім того, в Java є велика екосистема сторонніх бібліотек та фреймворків, які можуть значно спростити вашу роботу, забезпечити готові рішення та прискорити розробку.

Загалом, Java є потужною мовою програмування для розробки desktop -додатків, завдяки своїй кросплатформовості, широкому спектру функціональності, продуктивності, безпеці та підтримці спільноти. Вона є вдалим вибором для розробки автоматизованої системи для побудови плоского важільного механізму [14].

Можна виділити дві основні переваги мови програмування Java, якщо її використовувати для розробки додатку. По-перше, Java – це об’єктно-орієнтована мова, яка дозволяє побудувати фундамент програми, на базі якого можна будувати додаток та легко його розширювати. По-друге, Java має бібліотеки, які дозволяють створювати інтерфейс користувача з вже готових компонентів або розширювати готові компоненти.

2.2. Середа розробки IntelliJ IDEA

Використання зручного інтегрованого середовища розробки (Integrated Development Environment, IDE), такого як IntelliJ IDEA, має декілька важливих переваг:

1. IntelliJ IDEA є найбільш розширеним редактором, який допомагає розробникам писати високоякісний код швидше. Він надає функції підтримки кодування, які виявляють можливі помилки та пропонують покращення під час введення. Крім того, він надає інформацію про найкращі практики спільноти щодо кодування, виділяє нові функції мови та багато іншого.
2. IntelliJ IDEA забезпечує глибоке розуміння коду, володіючи всіма деталями вашого коду. Вона використовує ці знання, щоб забезпечити швидку навігацію та інтелектуальний досвід, надаючи відповідні рекомендації в будь-якому контексті.
3. Інтеграція з іншими інструментами: зручна IDE підтримує інтеграцію з іншими корисними інструментами, такими як: системи збирання, фреймворки, бібліотеки, бази даних тощо. Це дозволяє легко і швидко використовувати ці інструменти у процесі розробки.
4. Підтримка спільної роботи: зручна IDE надає можливості для спільної роботи над проектом. Вона підтримує системи керування версіями, спільне редагування коду, обговорення та коментування коду, що допомагає командам розробників ефективно співпрацювати над проектом.

Узагалі, використання зручної IDE, такої як IntelliJ IDEA, полегшує розробку програм, підвищує продуктивність розробників і забезпечує високу якість коду [15]. На (рис 2.1.) продемонстровано інтерфейс IDE.

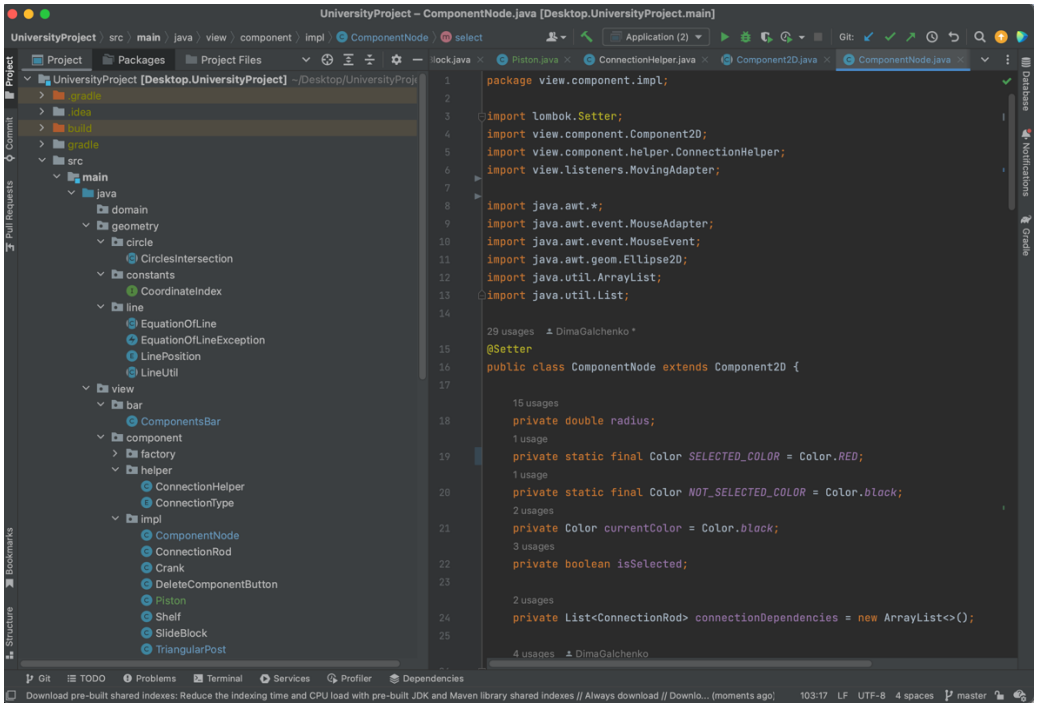


Рис 2.1. Інтерфейс IntelliJ IDEA

2.3. Структура проекту

Створення проекту в IntelliJ IDEA починається з головного меню. Для створення проекту треба натиснути на кнопку «New Project», як показано на малюнку (Рис 2.2.). Далі в меню «New Project» треба : дати ім’я проекту, вибрати мову програмування проекту, а також інструмент для складання проекту. Для даного проекту було вибрано мову програмування Java, а в якості інструменту для складання проекту – Gradle.

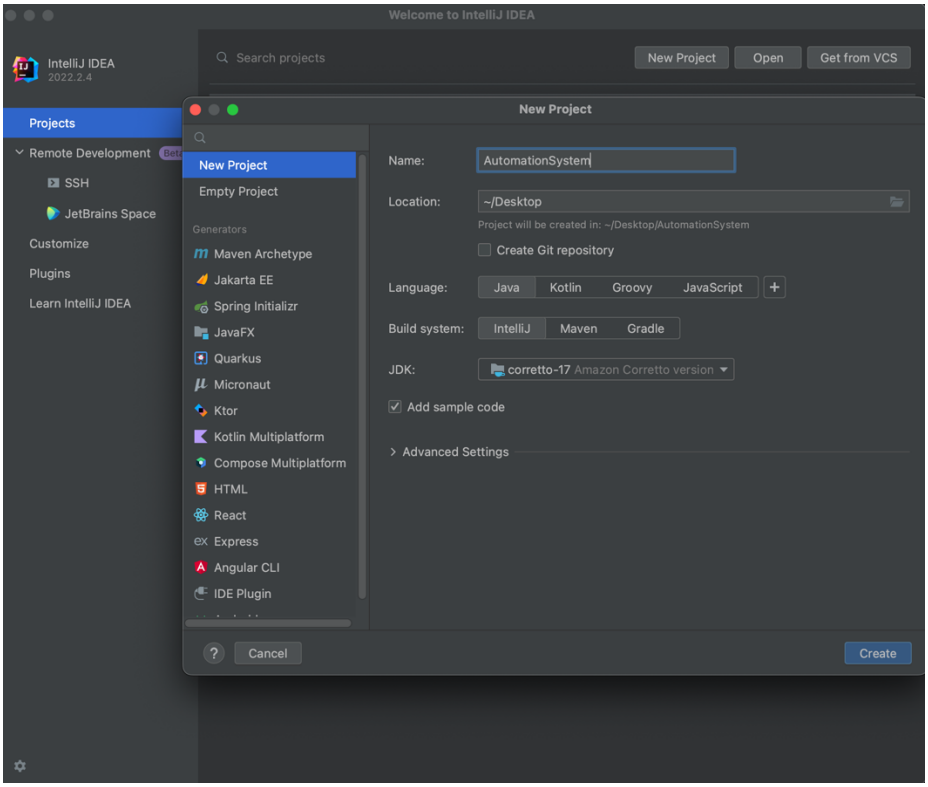


Рис 2.2. Головне меню для створення проекту

Після створення проекту, система розробки згенерує структуру проекту, як вказано на (Рис 2.3.). Всі файли програми будуть зберігатися в директорії з шляхом «src/main/java».

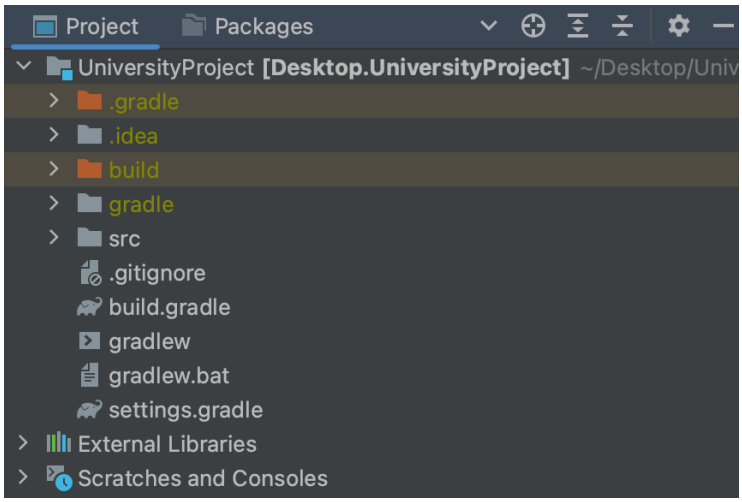


Рис 2.3. Початкова структура проекту

Після створення початкової структури проекту, було створено основні класи, які відповідають за відображення ланок. Для створення класу необхідно натиснути правою кнопкою миші на директорію (в якій потрібно створити файл класу) , далі в меню обрати пункт «New» та вибрати з випадаючого списку пункт «Java Class». Після натискання буде відображено поле, в якому треба дати назву класу.

В головній директорії було створено пакет під назвою «view», в якому зберігаються всі піддиректорії та класи, які стосуються відображення інтерфейсу користувача.

На (Рис 2.4.) зображено структуру пакетів розробленої системи. Структура складається з одного головного пакету «view», а також основного класу «Application». Клас «Application» відповідає тільки за запуск програми та створення головного вікна.

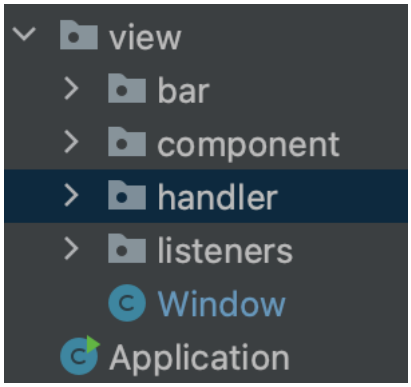


Рис 2.4. Структура пакетів програми

На (рис 2.5.) зображено розгорнуту структуру програми. В пакеті «view» розташовані 4 пакети: bar, component, handler, listeners та клас window. В пакеті «bar» знаходиться клас ComponentsBar, що відповідає за відображення лівого меню, в якому знаходяться іконки ланок, для перетягування на робочу область.

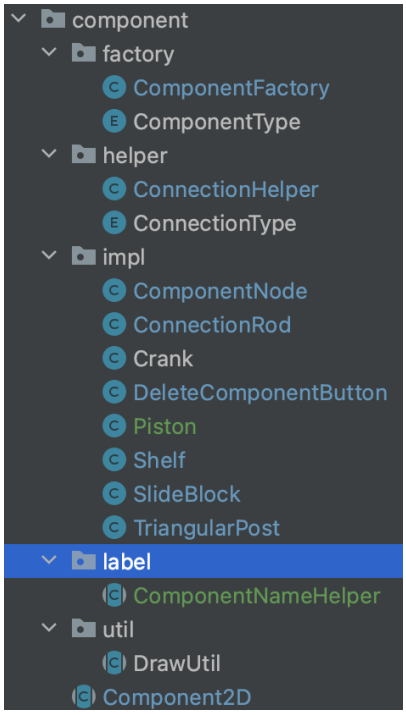


Рис 2.5. Структура пакету «component»

Пакет «component» має більш складну структуру та всі класи цього пакету відповідають за відображення ланок та взаємодію між ними. В пакеті «factory» знаходяться два класи: ComponentFactory та ComponentType. Для класу ComponentFactory застосовано шаблон проектування «фабрика», цей клас відповідає за створення ланок. Фабрика (Factory) - це шаблон проектування, який використовується для створення об'єктів без прямого виклику їх конструкторів.

На (рис 2.6.) зображено клас для створення ланок різних типів. Клас містить один метод «createComponent» з двома аргументами: тип ланки (ComponentType) та точка розташування ланки. ComponentType – має тип «enum» (перерахування), містить набір типів ланок.

Пакет «helper» містить два класи: ConnectionHelper та ConnectionType. ConnectionHelper – клас, який створює та відображає зв’язок між двома шарнірами. ConnectionType – перерахування типів зв’язку між шарнірами.

В пакеті «impl» містяться реалізації абстрактного класу Component2D для відображення конкретної ланки: ComponentNode (шарнір), ConnectionRod

(з’єднувальна ланка), Crank (кривошип), Piston (поршень), Shelf (полиця), SlideBlock (повзун) та TriangularPost (трикутна стійка).

```
public class ComponentFactory {

    1 usage   DimaGalchenko *
    public static Component2D createComponent(ComponentType type, Point point) {
        Component2D component2D = null;
        if (type == ComponentType.TRIANGULAR_POST) {
            component2D = new TriangularPost(point.x, point.y);
        } else if (type == ComponentType.SLIDE) {
            component2D = new SlideBlock(point.x, point.y);
        } else if (type == ComponentType.NODE) {
            component2D = new ComponentNode(point.x, point.y,  scale: 1,  radius: 10);
            ((ComponentNode) component2D).activateDrag();
        } else if (type == ComponentType.SHELF) {
            component2D = new Shelf(point.x, point.y);
        } else if (type == ComponentType.PISTON) {
            component2D = new Piston(point.x, point.y);
        }

        return component2D;
    }
}
```

Рис 2.6. Код класу ComponentFactory

В пакеті «label» знаходиться статичний клас ComponentNameHelper, який відповідає за зберігання та генерації порядкового номеру для кожної ланки.

Для генерації порядкового номеру в класі «ComponentNameHelper» знаходяться три статичні методи, як показано на (рис 2.7.): метод генерації порядкового номеру для з’єднань, метод генерації порядкової літери для шарніра та метод генерації порядкового номеру для шарнірів стійок у вигляді ‘O<порядковий номер>’. Для генерації літер використовується тип змінної char та початкове значення 64, що у таблиці ASCII відповідає літері «A». При кожному виклику метода, значення змінної збільшується, що супроводжується зміною літери в таблиці ASCII.

```java
public abstract class ComponentNameHelper {
    private static int lastConnectionSerialNumber = 0;
    private static char lastComponentNodeWord = 64;
    private static int lastFixedComponentSerialNumber = 0;

    public static String generateComponentNameForConnection() {
        return String.valueOf(++lastConnectionSerialNumber);
    }

    public static String generateSerialNameForNode() {
        return String.valueOf(++lastComponentNodeWord);
    }

    public static String generateFixedComponentName() {
        lastFixedComponentSerialNumber++;
        return "0" + lastFixedComponentSerialNumber;
    }
}
```

Рис 2.7. Код класу ComponentNameHelper

В пакеті «util» знаходиться клас DrawUtil, який містить в собі уніфіковані методи для відображення спільних елементів кожної ланки.

В корені директорії «component» знаходиться клас Component2D – це абстрактний клас, в якому знаходяться спільні поля та методи всіх типів ланок. Від нього успадковуються всі класи в пакеті «impl».

Наступний пакет в директорії «view», «handler» містить два класи: ComponentExportTransferHandler та ComponentImportTransferHandler. Ці два класи відповідають за реалізацію функції перетягування компонентів з лівого меню на робочу область, тобто за експорт з лівого меню та імпорт на робочу область відповідно. На (рис 2.8.) зображено код класу ComponentExportTransferHandler, він містить в собі поле value зі значенням назви ланки, яку користувач намагається перетягнути. Цей клас реагує на подію перетягування пунктів лівого меню компонентів та передає цю зміну в клас ComponentImportTransferHandler. На (рис 2.9.) зображено код класу, який імпортує об’єкт на робочу область. В цьому класі знаходиться метод importData з аргументом «support», в якому зберігається така інформація: назва ланки, яку користувач перетягнув та точка, в яку користувач перетягнув ланку.

```java
public class ComponentExportTransferHandler extends TransferHandler {
    public static final DataFlavor SUPPORTED_DATE_FLAVOR = DataFlavor.stringFlavor;
    2 usages
    private String value;

    1 usage  DimaGalchenko
    public ComponentExportTransferHandler(String value) {
        this.value = value;
    }

    1 usage  DimaGalchenko
    public String getValue() {
        return value;
    }

    DimaGalchenko
    @Override
    public int getSourceActions(JComponent c) {
        return DnDConstants.ACTION_COPY_OR_MOVE;
    }

    DimaGalchenko
    @Override
    protected Transferable createTransferable(JComponent c) {
        Transferable t = new StringSelection(getValue());
        return t;
    }

    8 usages  DimaGalchenko
    @Override
    protected void exportDone(JComponent source, Transferable data, int action) {
        super.exportDone(source, data, action);
        // Decide what to do after the drop has been accepted
    }
```

Рис 2.8. Код класу ComponentExportTransferHandler

Як видно на (рис 2.9.) ми використовуємо ComponentType для визначення типу ланки, а також ComponentFactory для створення компоненту без прямого виклику конструктора конкретної реалізації.

В директорії «listeners» знаходиться клас MovingAdapter, який відслідковує та реагує на події натискання миші на компонент, затискання миші на компоненті та перетягування компонента в робочій області.

```java
public class ComponentImportTransferHandler extends TransferHandler {
    public static final DataFlavor SUPPORTED_DATE_FLAVOR = DataFlavor.stringFlavor;

    public ComponentImportTransferHandler() {}

    @Override
    public boolean canImport(TransferHandler.TransferSupport support) {...}

    @Override
    public boolean importData(TransferHandler.TransferSupport support) {
        boolean accept = false;
        if (canImport(support)) {
            try {
                Transferable t = support.getTransferable();
                Object value = t.getTransferData(SUPPORTED_DATE_FLAVOR);
                if (value instanceof String) {
                    JComponent panel = (JComponent) support.getComponent();
                    if (panel instanceof JPanel) {
                        ComponentType type = ComponentType.valueOf((String) value);
                        Point dropPoint = support.getDropLocation().getDropPoint();
                        dropPoint.setLocation( x: dropPoint.x - 50,  y: dropPoint.y - 20);
                        Component2D component2D = ComponentFactory.createComponent(type, dropPoint);
                        panel.add(component2D);
                        panel.repaint();
                    }
                }
            } catch (Exception exp) {
                exp.printStackTrace();
            }
        }
        return accept;
    }
}
```

Рис 2.9. Код класу ComponentImportTransferHandler

2.4. Бібліотека для створення графічного інтерфейсу – Swing

Розглянемо актуальну бібліотеку для створення інтерфейсу користувача – Swing. Java Swing є складовою Java Foundation Classes (JFC) і використовується для створення програм з графічним інтерфейсом. Він розроблений на базі API AWT (Abstract Windowing Toolkit) і повністю написаний на мові Java. На відміну від AWT, Java Swing надає незалежні від платформи та легкі компоненти. У пакеті javax.swing знаходяться класи, які використовуються для API Java Swing, такі як: JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser і багато інших.

JFC (Java Foundation Classes) є набором функцій для розробки графічних користувальницьких інтерфейсів (GUI), для надання розширених графічних

можливостей та інтерактивності до Java-програм. Вона охоплює низку функцій, перерахованих у таблиці нижче.

Таблиця 2.1. Таблиця функцій JFC та їх опис [16]

Функція	Опис
Елементи графічного інтерфейсу Swing	Включає все, від кнопок до роздільних панелей та таблиць. Багато компонентів можуть сортувати, друкувати та підтримувати функції перетягування.
Підтримка змінного вигляду	Вигляд і стиль застосунків Swing можна змінювати, дозволяючи вибрати потрібний вигляд. Наприклад, та сама програма може використовувати вигляд Java або вигляд Windows. Крім того, платформа Java підтримує вигляд GTK+, що робить доступними сотні існуючих виглядів для програм Swing. Доступні й інші пакети виглядів з різних джерел.
API доступності	Дозволяє адаптивним технологіям, таким як: програми читання екрану та пристрої з системою Брайля, отримувати інформацію з користувацького інтерфейсу.
Java 2D API	Дозволяє розробникам легко включати високоякісну 2D-графіку, текст та зображення в програми та аплети. Java 2D включає широкий спектр API для генерації та надсилання високоякісного виводу на друкарські пристрої.
Інтернаціоналізація	Дозволяє розробникам створювати програми, які можуть взаємодіяти з користувачами з усього світу на їх власних мовах та культурних конвенціях. За допомогою методу вводу розробники можуть створювати програми, які приймають текст на мовах з тисячами різних символів, таких як: японська, китайська або корейська.

Java Swing має ієрархію класів, що включає багато компонентів і утиліт, які допомагають в розробці графічного інтерфейсу користувача.

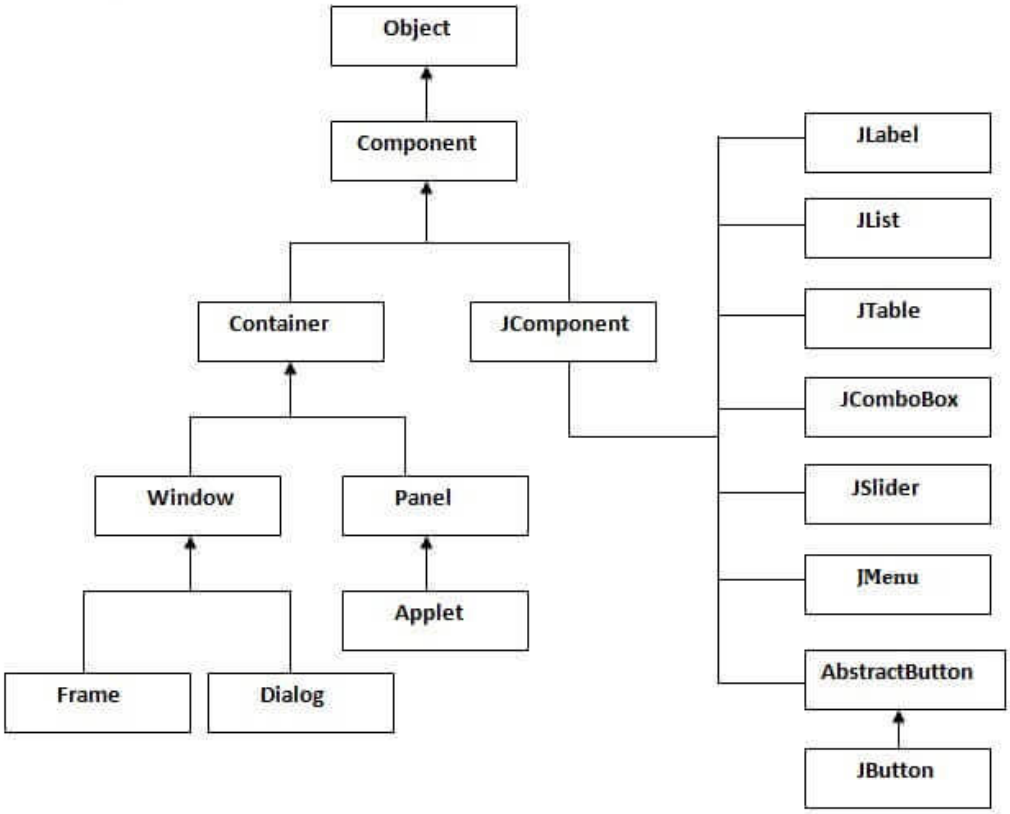


Рис 2.10. Ієрархія класів в бібліотеці Swing [17]

Основою ієрархії є клас «JComponent», який є базовим класом для більшості компонентів Swing. Нижче наведено деякі важливі класи, які належать до ієрархії класів Java Swing:

- «JComponent»: базовий клас для більшості компонентів Swing. Включає методи для малювання, обробки подій та роботи зі шрифтами та кольорами.
- «Container»: клас, що представляє контейнер, який може містити інші компоненти. Включає методи для додавання та управління компонентами всередині себе.
- «JPanel»: контейнер, який може містити інші компоненти. Часто використовується для організації компонентів у групи або панелі.

- «JFrame»: головне вікно програми. Включає заголовок, панель меню та можливості зміни розмірів та закриття вікна.
- «JButton»: компонент, що представляє кнопку, яку можна натиснути.
- «JLabel»: компонент, що являє собою текстову мітку, яку можна використовувати для відображення назви або пояснювального тексту.
- «JTextField»: компонент, що дозволяє користувачеві вводити однорядковий текст.
- «JTextArea»: компонент, що дозволяє користувачеві вводити багаторядковий текст.
- «JScrollPane»: компонент, який надає прокрутку для інших компонентів, що перевищують доступний простір.
- «JTable»: компонент, що використовується для відображення даних у вигляді табличної структури.

Це лише кілька прикладів класів з ієрархії Java Swing. Кожен клас має свої властивості, методи та можливості, які допомагають розробникам створювати різноманітні інтерфейси користувача [17].

Розроблення додатку за допомогою бібліотеки Swing, починається із створення головного вікна. На (рис 2.11.) продемонстровано клас Window, при проектуванні даного класу було використано шаблон проектування сінглтон. Сінглтон (Singleton) - це шаблон проектування, який обмежує створення лише одного екземпляру класу і надає глобальну точку доступу до цього екземпляру. Такий шаблон було використано для того, щоб мати глобальну точку доступу до головного вікна. Для того, щоб створити вікно необхідно ініціалізувати клас JFrame, та виставити всі необхідні параметри вікна. Для того, щоб встановити розташування компонентів у вікні , потрібно змінити властивість layout. Для даної системи було використано BorderLayout. BorderLayout - це менеджер розташування в Java Swing, який розділяє контейнер на п'ять областей: північну (NORTH), південну (SOUTH), східну (EAST), західну (WEST) та центральну (CENTER). Кожна область може містити лише один компонент, який займає всю доступну площу відповідної області. На головне вікно

додано два компоненти: ConstructionPanel (робоча область) та ComponentsBar (меню компонентів). Для них було задано центральне та західне розташування відповідно.

```
public class Window{

    public static JFrame frame;

    public static JFrame getInstance() {
        if(frame == null) {
            initializeFrame();
            return frame;
        }
        return frame;
    }

    private static void initializeFrame() {
        frame = new JFrame();
        frame.setSize( width: 800,  height: 800);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        ConstructionPanel constructionPanel = ConstructionPanel.getInstance();
        frame.add(constructionPanel, BorderLayout.CENTER);

        ComponentsBar componentsBar = new ComponentsBar();
        frame.add(componentsBar, BorderLayout.WEST);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

Рис 2.11. Код класу Window

На (рис 2.12.) продемонстровано структуру класу для відображення меню з ланками. Для того, щоб додати компонент на вікно, він має успадковувати один із різновидів компонентів, які наведені на (рис 2.10.). Клас ComponentsBar успадковує JPanel. Для розташування компонентів в меню було використано менеджер розташування GridLayout. GridLayout - це менеджер розташування в Java Swing, який розділяє контейнер на сітку рядків і стовпців, де кожна клітинка сітки може містити один компонент. Клас ComponentsBar має поля:

- IMAGE_SRC_LIST – константний масив строкових змінних, який містить назви іконок ланок.
- COMPONENT_TYPES – константний масив типів ланок.

Кожен пункт меню належить класу JButton, а замість надпису на кнопці використовується іконка відповідної ланки. Для кожної кнопки в меню додано

TransferHandler. TransferHandler є класом в Java Swing, який надає підтримку перетягування і перетворення даних між компонентами.

```
public class ComponentsBar extends JPanel {
    private static final String[][] IMAGE_SRC_LIST = new String[][]{
            {"triangular-post.png", "post.png"},
            {"component-node.png"},
            {"piston.png", "slide.png"}
    };

    private static final ComponentType[][] COMPONENT_TYPES = new ComponentType[][]{
            {ComponentType.TRIANGULAR_POST, ComponentType.SHELF},
            {ComponentType.NODE},
            {ComponentType.PISTON, ComponentType.SLIDE}
    };

    public static final List<JButton> buttons = new ArrayList<>();

    public ComponentsBar() {
        setPreferredSize(new Dimension( width: 80,  height: 100));
        setBackground(Color.decode( nm: "#e1e0df"));
        for (int i = 0; i < IMAGE_SRC_LIST.length; i++) {
            JPanel panel = new JPanel();
            panel.setLayout(new GridLayout(IMAGE_SRC_LIST[i].length,  cols: 1,  hgap: 1,  vgap: 2));
            JScrollPane scrollPane = new JScrollPane(panel);
            for (int j = 0; j < IMAGE_SRC_LIST[i].length; j++) {
                JButton button = new JButton(new ImageIcon( filename: "src/main/resources/" + IMAGE_SRC_LIST[i][j]));
                panel.add(button);
                button.setPreferredSize(new Dimension( width: 60,  height: 60));
                button.setTransferHandler(new ComponentExportTransferHandler(COMPONENT_TYPES[i][j].name()));
                button.addMouseMotionListener(new MouseAdapter() {
                    @Override
                    public void mouseDragged(MouseEvent e) {
                        JButton button = (JButton) e.getSource();
                        TransferHandler handle = button.getTransferHandler();
                        handle.exportAsDrag(button, e, TransferHandler.COPY);
                    }
                });
                buttons.add(button);
            }
```

Рис 2.12. Код класу ComponentsBar

Java Swing дозволяє створювати власні шрифти та додавати надписи в компоненти. Для даної автоматизованої системи було застосовано надписи для нумерації ланок.

```
Font font = new Font( name: "Serif", Font.PLAIN,  size: 16);
g.setColor(Color.BLUE);
g.setFont(font);
graphics.drawString(serialName, paddingTop, paddingLeft);
```

Рис 2.13. Код створення шрифту та додавання тексту

На (рис 2.13.) наведено приклад коду з розробленої автоматизованої системи для створення шрифту та додавання тексту. У першому рядку коду створюється шрифт з

наступними параметрами: ім’я, стиль та розмір шрифту. У наступних рядках коду встановлюється синій колір для відображення та встановлюється відповідно шрифт. В останньому рядку використовується функція «drawString» з наступними параметрами: текст, координата по осі X та координата по осі У.

Для обробки подій миші в Java Swing використовується адаптер – MouseAdapter. На (рис 2.14.) зображено код адаптеру, який дозволяє відслідковувати події миші в робочій області та реалізує перетягування компонентів. Клас MovingAdapter відслідковує дві події: MousePressed (натискання миші) та MouseDragged (перетягування миші). Коли користувач натискає на компонент, клас MovingAdapter запам’ятовує координату миші. Коли користувач перетягує курсор по робочій області не відпускаючи клавішу миші, метод mouseDragged рахує відстань між координатою першого натискання клавіші та координатою курсора в даний момент часу. Маючи відстань між цими координатами, можна перемалювати ланку у новій координаті додав до її координат відстань по кожній осі.

```
public class MovingAdapter<T extends Component2D> extends MouseAdapter {
    private int x;
    private int y;
    private T component;

    private boolean isSelected;

    public MovingAdapter(T component) {
        this.component = component;
    }

    public void mousePressed(MouseEvent e) {
        x = e.getXOnScreen();
        y = e.getYOnScreen();
    }

    public void mouseDragged(MouseEvent e) {
        Point point = e.getLocationOnScreen();
        int dx = (int) (point.getX() - x);
        int dy = (int) (point.getY() - y);

        if (dx != 0 || dy != 0) {
            component.setXPosition(component.getXPosition() + dx);
            component.setYPosition(component.getYPosition() + dy);
            component.setLocation(component.getXPosition(), component.getYPosition());
            component.repaint();
            x += dx;
            y += dy;
        }
    }
```

Рис 2.14. Код класу MovingAdapter

Для відображення ланок, було використано метод «paint», який застосовується для відображення вмісту контенту на екрані. В Java Swing, для створення власних компонентів, необхідно розширити клас JComponent або його підкласи і перевизначити метод «paint». Для відображення ланок використовувались звичайні геометричні фігури. Процес побудови стійки можна розбити на наступні етапи:

1. Побудова трикутника
2. Додавання шарніра
3. Додавання полиці

Так, Java Swing надає розробникам можливість створювати гнучкий інтерфейс користувача. З використанням різноманітних компонентів та менеджерів розташування, можна створювати різноманітні макети та розміщувати компоненти на екрані відповідно до своїх потреб.

Нижче наведено деякі функціональні можливості, які дозволяють розробляти гнучкий інтерфейс у Java Swing:

1. Компоненти: Java Swing має широкий спектр готових компонентів, таких як: кнопки, поля введення, списки, таблиці, панелі тощо. Це дає можливість налаштовувати вигляд та поведінку цих компонентів за допомогою різних властивостей та методів.
2. Менеджери розташування: Swing надає різні менеджери розташування (layout managers), такі як: `FlowLayout`, `BorderLayout`, `GridLayout`, `GridBagLayout`, `BoxLayout` тощо. Можна вибрати відповідний менеджер розташування для контейнера та визначити, як компоненти будуть розміщуватися на екрані.
3. Події та обробники: Java Swing надає систему подій та обробників подій, що дозволяє реагувати на дії користувача, такі як: натискання кнопок, введення тексту, переміщення миші тощо. Є можливість встановлювати обробники подій для компонентів і виконувати відповідні дії відповідно до подій.
4. Налаштування стилів: Swing дозволяє налаштовувати стилі візуального вигляду компонентів за допомогою плафонів (look and feel). Є можливість

вибрати плафон, який найкраще підходить до обраного дизайну та вимог інтерфейсу.

Загалом, Java Swing надає розробникам багато інструментів і можливостей для створення гнучкого та адаптивного інтерфейсу користувача.

2.5. Об'єктно-орієнтоване програмування

Об'єктно-орієнтоване програмування (ООП) зараз є провідним підходом до програмування, який замінив структурний або процедурний підхід, розроблений в 1970-х роках. Мова Java є повністю об'єктно-орієнтованою і для продуктивного програмування на ній необхідно знати основні принципи ООП.

ООП базується на понятті об'єктів, кожен з яких має власні функціональні можливості, які доступні користувачам, а також приховану реалізацію. Об'єкти можна використовувати готові, зібрані з бібліотеки або проектувати нові з врахуванням власних потреб. Чи обирати готові об'єкти, чи будувати їх самостійно, залежить від обсягу роботи та ресурсів. Однак, поки об'єкти задовольняють вимоги і виконують необхідні функції, програмісту не потрібно докладати особливих зусиль щодо реалізації цих функціональностей.

Традиційний структурний підхід до програмування передбачає розробку набору процедур або алгоритмів для вирішення поставленої задачі. Після визначення цих процедур програміст повинен вирішити, які структури даних використовувати. Основний акцент робиться на алгоритмах, а структури даних обираються під них. Однак, ООП підходи змінюють цю ситуацію, пріоритет ставиться на даних, а алгоритми служать для їх обробки.

При вирішенні невеликих задач процедурний підхід може бути прийнятним, але об'єктно-орієнтоване програмування є більш придатним для вирішення складних задач. Наприклад (рис 2.15.), розглянемо невеликий веб-браузер. У структурному підході до його реалізації можуть бути задіяні 2000 процедур, кожна з яких маніпулює

глобальним набором даних. За ООП підходом та сама програма може бути побудована з 100 класів, в кожному з яких в середньому визначено по 20 методів. Така структура програми зручніша для розробки і в ній легше виявляти помилки. Наприклад, якщо дані об'єкта перебувають у некоректному стані, виявити причину проблеми набагато простіше серед 20 методів, які мають доступ до цих даних, ніж серед 2000 процедур.

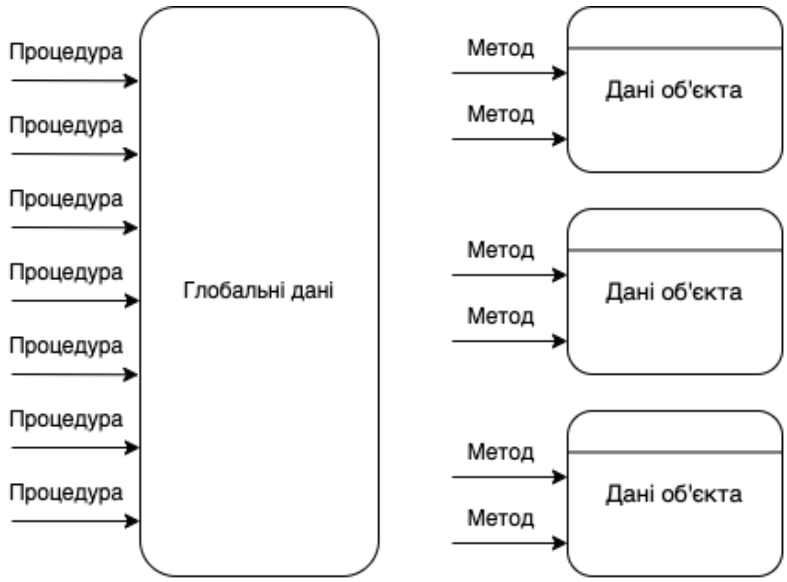


Рис 2.15. Порівняння процедурного та об'єктно-орієнтованого програмування

Одним з найважливіших понять є клас, він є шаблоном або зразком, на основі якого буде створено об'єкт. Зазвичай клас порівнюють з формою для випічки печива, а об'єкт - печиво. Процес створення об'єкта на основі певного класу називається створенням екземпляра цього класу. В ООП, об'єкти мають наступні ключові характеристики:

- Поведінка об'єкта визначає, які дії можна виконати з об'єктом та які методи доступні для його використання.
- Стан об'єкта відображає, як об'єкт реагує на викликані методи та які зміни можуть відбуватися в його внутрішньому стані.
- Ідентичність об'єкта - це те, що робить об'єкт унікальним і відрізняє його від інших об'єктів зі схожою поведінкою та станом.

Всі об'єкти, які є екземплярами одного і того ж класу, ведуть себе аналогічно. Поведінка об'єкта визначається методами, які можна викликати. Кожен об'єкт зберігає інформацію про свій стан. З часом стан об'єкта може змінюватися, але це не відбувається само по собі. Стан об'єкта може змінюватися тільки в результаті викликів методів (якщо стан об'єкта змінюється з інших причин, це означає, що принцип інкапсуляції порушений).

Стан об'єкта не повністю описує його, оскільки кожен об'єкт має свою власну ідентичність. Наприклад, у даній автоматизованій системі для побудови плоского важільного механізму дві ланки можуть відрізнятися одна від одної, навіть, якщо вони відносяться до одних і тих самих ланок. Варто зазначити, що окремі об'єкти, які є екземплярами класу, завжди відрізняються своєю ідентичністю і , як правило, своїм станом. Ці основні властивості об'єктів можуть взаємодіяти одне з одним.

Існують три загальні види взаємозв'язків між класами :

1. Залежність ("використовує" - щось): це найочевидніше і поширене відношення. Наприклад, клас "Замовлення" використовує клас "Рахунок", оскільки йому потрібний доступ до рахунків для здійснення операцій замовлення. Залежність виникає, коли методи одного класу виконують дії з екземплярами іншого класу.
2. Агрегація ("містить" - щось): це відношення, коли об'єкт одного класу містить або складається з об'єктів іншого класу. Наприклад, об'єкт "Замовлення" може містити об'єкти "Товар". Агрегація означає, що один клас містить інші об'єкти в своєму складі.
3. Наслідування ("є" - чимось): це відношення, коли один клас успадковує властивості та методи іншого класу. Наприклад, клас "Круг" може успадковувати властивості та методи класу "Фігура". Наслідування означає, що клас є розширенням або спеціалізацією іншого класу.

Уникайте надмірної залежності між класами. Якщо клас А не залежить від класу В, то він не буде враховувати будь-які зміни, які стосуються класу В. Мінімізація залежностей допомагає забезпечити більшу гнучкість та модульність в програмі.

Об’єктно-орієнтоване програмування базується на трьох основних принципах: інкапсуляція, наслідування та поліморфізм.

Інкапсуляція (іноді відома як приховування інформації) є важливим концептом в роботі з об'єктами. Формально інкапсуляція відображається у збереженні даних та відповідних операцій над ними в одній одиниці, а також у приховуванні даних від інших об'єктів. Дані в об'єкті називаються полями, а функції та процедури, які працюють з даними - методами. В конкретному об'єкті, який є екземпляром класу, поля об'єкта мають певні значення. Сукупність цих значень називається станом об'єкта. Виклик методу для об'єкта може змінити його стан [18].

В розробленій автоматизованій системі для побудови плоского важільного механізму при розробці було застосовано принцип інкапсуляції. Таким чином, для кожного класу в програмі було сховано всі поля об’єкта, а також додано методи для маніпуляції зі станом об’єкта. Розглянемо на прикладі класу ComponentNode, цей клас відображає шарнір.

```
29 usages   DimaGalchenko *
@Setter
public class ComponentNode extends Component2D {

    15 usages
    private double radius;
    1 usage
    private static final Color SELECTED_COLOR = Color.RED;
    1 usage
    private static final Color NOT_SELECTED_COLOR = Color.black;
    2 usages
    private Color currentColor = Color.black;
    3 usages
    private boolean isSelected;

    2 usages
    private List<ConnectionRod> connectionDependencies = new ArrayList<>();

    4 usages   DimaGalchenko
    public ComponentNode(int xPosition, int yPosition, double scale, double radius) {...}

    1 usage   DimaGalchenko
    public void activateDrag() {...}

    DimaGalchenko
    @Override
    public void paint(Graphics g) {...}

    2 usages   DimaGalchenko
    public void addDependency(ConnectionRod connectionRod) {...}

    4 usages   DimaGalchenko
```

Рис 2.16. Приклад приховування даних

Як видно на (рис 2.16.) до всіх полів класу додано модифікатор доступу private, це означає, що отримати доступ до цього поля можна лише в рамках даного класу. Таким чином, на прикладі цього класу було застосовано один з принципів ООП – інкапсуляція.

Наступний принцип ООП – спадкування, його важливо розглянути до поліморфізму, так як поліморфізм реалізується на основі спадкування.

Спадкування виражає зв'язок між конкретним класом та більш загальним класом. Наслідування - це відношення між двома класами, де один клас використовує властивості та методи іншого класу. Клас, який наслідує, називається батьківським класом або суперкласом, а клас, який успадковує - дочірнім класом або підкласом.

При наслідуванні, підклас успадковує всі властивості та методи свого батьківського класу, що дозволяє підкласу використовувати та розширювати їх функціонал. Підклас може додавати власні методи та властивості, а також перевизначати або розширювати методи батьківського класу.

Наслідування дозволяє створювати ієрархію класів з більш загальними та абстрактними класами, що мають спільний функціонал до більш спеціалізованих та конкретних класів. Це допомагає уникнути повторення коду та забезпечує більш структуровану та розширювану архітектуру програми [18].

В даній роботі було застосовано наслідування при проектуванні класів, які відповідають за відображення ланок механізму.

На (Рис 2.17.) продемонстровано спрощену ієрархію класів в програмі автоматизованої системи для побудови плоского важільного механізму. Найвищий клас в ієрархії JComponent – це базовий клас для більшості компонентів Swing. Всі його нащадки мають поведінку, яка дозволяє маніпулювати ними, як звичайними компонентами з бібліотеки Swing.

Наступний в ієрархії Component2D, цей клас абстрактний і у ньому зберігаються: координати елемента на екрані, масштабний коефіцієнт та метод для відображення компонента, його було додано в клас, так як він відображає загальні компоненти для кожної ланки, такі як: кнопка видалення компонента, кнопка повороту компонента та рамка компонента.

Далі в ієрархії розташовані класи конкретних ланок, які перевизначають метод paint(), у якому відбувається відображення для конкретної ланки, а також викликається метод супер класу для відображення загальних компонентів, які притаманні кожній ланці в програмі.

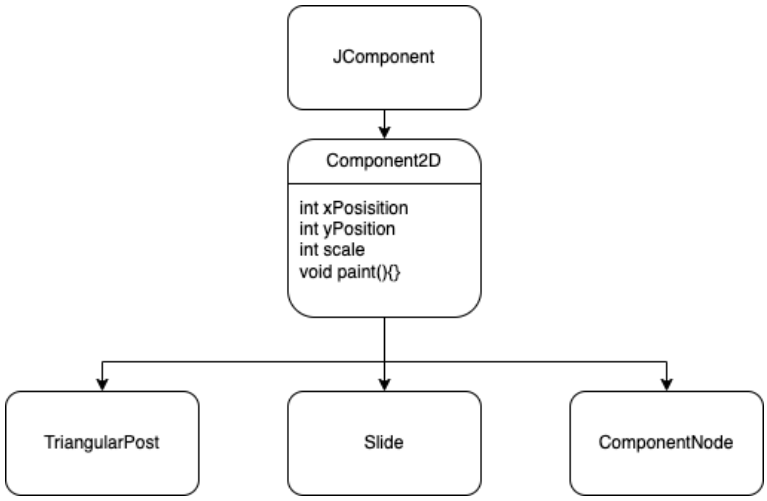


Рис 2.17. Ієрархія класів для відображення ланок

Наступний принцип ООП – поліморфізм. Це принцип об'єктно-орієнтованого програмування, який дозволяє об'єктам різних класів виконувати одну і ту саму операцію по-різному. Поліморфізм дозволяє використовувати спільний інтерфейс для обробки об'єктів різних типів.

У програмуванні поліморфізм може бути реалізований через перевантаження методів, де різні класи мають методи з однаковим ім'ям, але з різними параметрами або типами даних. Коли викликається цей метод, виконується відповідна версія методу в залежності від типу об'єкта, з яким працюємо. Це не найкраще використання поліморфізму.

Поліморфізм також може бути досягнутий через успадкування та перевизначення методів. Дочірні класи можуть перевизначити методи свого батьківського класу, щоб змінити їх реалізацію або додати додатковий функціонал. При цьому, коли викликається метод через посилання на батьківський клас, виконується версія методу з підкласу.

Завдяки поліморфізму можна писати загальні та гнучкі алгоритми, які можуть працювати з різними типами об'єктів, без необхідності знати їх конкретний тип. Це сприяє повторному використанню коду та забезпечує більш гнучку та розширювану архітектуру програми [18].

В даній автоматизованій системі принцип поліморфізму був використаний при додаванні ланок на робочу область.

На (рис 2.18.) продемонстровано код, який додає ланку на робочу область. Так як всі класи ланок є спадкоємцями класу JComponent, то їх можна додавати на робочу панель, як звичайний компонент з бібліотеки Swing. Коли буде викликаний метод для відображення елемента, буде викликана реалізація об’єкта, на який вказує посилання “component2D”.

```
@Override
public boolean importData(TransferHandler.TransferSupport support) {
    boolean accept = false;
    if (canImport(support)) {
        try {
            Transferable t = support.getTransferable();
            Object value = t.getTransferData(SUPPORTED_DATE_FLAVOR);
            if (value instanceof String) {
                JComponent panel = (JComponent) support.getComponent();
                if (panel instanceof JPanel) {
                    ComponentType type = ComponentType.valueOf((String) value);
                    Point dropPoint = support.getDropLocation().getDropPoint();
                    dropPoint.setLocation( x: dropPoint.x - 50,  y: dropPoint.y - 20);
                    Component2D component2D = ComponentFactory.createComponent(type, dropPoint);
                    panel.add(component2D);
                    panel.repaint();
                }
            }
        } catch (Exception exp) {
            exp.printStackTrace();
        }
    }
    return accept;
}
```

Рис 2.18. Приклад використання поліморфізму

Це дозволяє легко розширювати програму і як наслідок, можна створити новий тип ланки, успадкувати його від класу Component2D, перевизначити метод “paint”, додати в нього логіку для малювання відповідної ланки, а також додати виклик

метода “paint” супер класу і стане можливим додати новий компонент в програму не змінюючи код, який вже є.

2.6. Тестування програми

Тестування програмного забезпечення - це процес перевірки, який включає порівняння фактичної роботи програми з очікуваним результатом. Цей процес включає в себе створення відповідного набору тестів, їх виконання та аналіз результатів. Тестування програмного забезпечення є важливою складовою контролю якості і включає такі етапи, як: планування, проектування, виконання тестів і аналіз отриманих даних.

Якість програмного забезпечення (Software Quality) - це сукупність характеристик програмного забезпечення, які стосуються його здатності задовольняти встановлені і припущені потреби.

Верифікація (verification) - це процес оцінки системи або її компонентів з метою визначення, чи відповідають результати поточного етапу розробки умовам, сформованим на початку цього етапу [IEEE]. Іншими словами, перевіряється, чи досягнуті наші цілі, терміни, завдання з розробки проекту, визначені на початку поточної фази.

Головні цілі тестування включають:

- Забезпечення надійності та коректності роботи тестового додатку у різних умовах.
- Переконання, що тестовий додаток відповідає всім вимогам та специфікаціям.
- Забезпечення актуальної інформації про поточний стан продукту.

Етапи тестування:

1. Аналіз продукту
2. Взаємодія з вимогами

3. Розробка стратегії тестування та планування процедур контролю якості
4. Створення тестової документації
5. Тестування прототипу
6. Основне тестування
7. Стабілізація

Рівні тестування — це різні етапи та підходи до тестування програмного забезпечення, які спрямовані на виявлення помилок та перевірку відповідності програмних продуктів вимогам.

Основні рівні тестування включають:

1. Модульне тестування (Unit Testing): перевірка окремих модулів або компонентів програми для переконання в їх правильному функціонуванні.
2. Інтеграційне тестування (Integration Testing): перевірка взаємодії між різними модулями або компонентами програми для виявлення помилок, які можуть виникнути при їх злитті.
3. Системне тестування (System Testing): перевірка системи в цілому згідно з її вимогами та функціональними вимогами для виявлення помилок та оцінки якості роботи.
4. Приймальне тестування (Acceptance Testing): виконання тестів користувачами або представниками клієнта з метою переконатися, що програма відповідає їх вимогам та готова до прийняття.
5. Функціональне тестування (Functional Testing): перевірка функціональних можливостей програми, її реакції на різні вхідні дані та очікувані результати.
6. Навантажувальне тестування (Load Testing): тестування продуктивності та стабільності системи при підвищеному навантаженні для оцінки її роботи в реальних умовах високого трафіку [19].

Дана автоматизована система було протестована в ручну. Процес ручного тестування для desktop-додатку включає наступні кроки для тестування функціональності переносу об'єкту за допомогою функції «drag and drop» з лівого меню елементів на робочу область:

1. Підготовка тестового середовища: встановлення додатку та забезпечення необхідних умов для проведення тестування.
2. Аналіз тест-кейсу: ретельне ознайомлення з тест-кейсом, що описує кроки тестування та очікувані результати.
3. Підготовка тестових даних: створення або завантаження елементів в ліве меню, які можуть бути перетягнуті на робочу область.
4. Виконання тест-кейсу:
   a. Запуск додатку та навігація до відповідної сторінки або модуля, де доступний «drag and drop».
   b. Знаходження об'єктів у лівому меню, які можуть бути перетягнуті та перетягування одного з них на робочу область за допомогою «drag and drop».
   c. Перевірка, чи відбулось коректне перенесення об'єкту на робочу область та його відображення.
   d. Перевірка, чи відбулась зміна стану об'єкту на робочій області після переносу.
   e. Перевірка, чи відповідає результат перенесення об'єкту очікуваному результату, відповідно до тест-кейсу.
5. Запис результатів тестування: фіксування результатів кожного кроку тест-кейсу, включаючи будь-які виявлені помилки або проблеми.
6. Відновлювальне тестування: у випадку виявлення помилок, перевірка, чи можна відновити та повторити проблему, якщо застосувати перенос об'єкту за допомогою «drag and drop» ще раз або в інших умовах.
7. Документування результатів: запис результатів тестування, включаючи опис виявлених помилок, проблем або недоліків, якщо такі є.
8. Звіт про тестування: підготовка звіту, що містить опис проведеного тестування, включаючи виконані кроки, результати та виявлені проблеми.

2.7. Огляд функціоналу

Розроблена автоматизована система дозволяє будувати механізм з готових ланок, а також легко перебудовувати механізм перетягуючи або видаляючи будь-яку ланку.

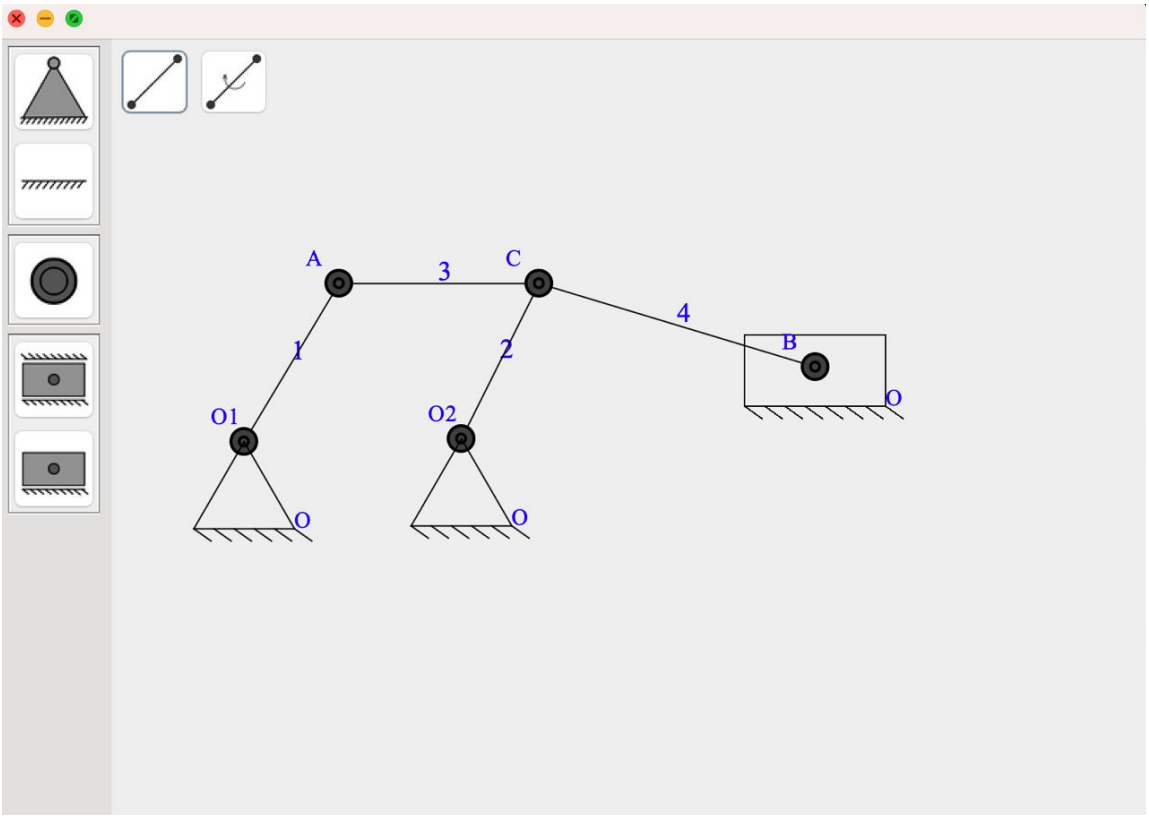


Рис 2.19. Інтерфейс автоматизованої системи для побудови плоского важільного механізму

На (рис. 2.19.) зображено інтерфейс автоматизованої системи для побудови плоского важільного механізму. Зліва розташовано меню з елементами (з підтримкою прокручування у разі, якщо ланок буде більше ніж висота меню). Кожен елемент в лівому меню можна перетягнути на робочу область, яка знаходиться справа. На робочій області побудовано механізм за допомогою перетягування елементів з лівого меню та з'єднання, які поєднанні між собою за допомогою кнопок у верхньому меню. Кожна стійка позначається літерою «О». Кожен шарнір стійки позначається літерою

«О» та порядковим номером, який відповідає порядку додавання стійки на робочу область. Всі рухливі шарніри позначаються англійськими літерами в алфавітному порядку.

На (рис. 2.20.) зображено процес перетягування ланки з меню елементів на робочу область. Для того щоб перетягнути елемент, потрібно навести курсор мишки на ланку в меню, натиснути на ліву клавішу та перетягувати елемент в будь-яке місце робочої області.

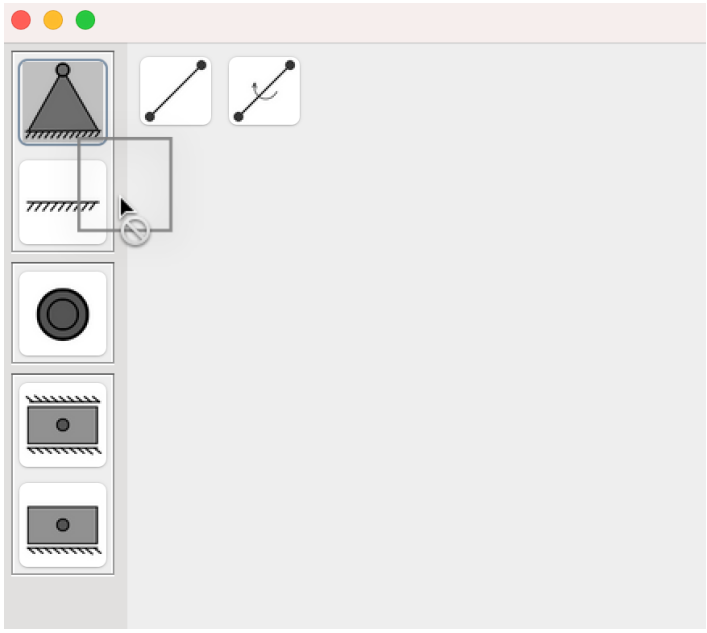


Рис 2.20. Демонстрація перетягування ланки на робочу область

Робоча область представляє собою прямокутник з сірим фоном. В робочій області можуть бути розташовані ланки механізму. Кожну ланку можна обрати для подальшого перетягування. Якщо натиснути лівою кнопкою миші на ланку, то навколо неї з’явиться прямокутних з штрих пунктирної лінії, а також кнопкою видалення в верхньому правому куті, як на (рис 2.21.).

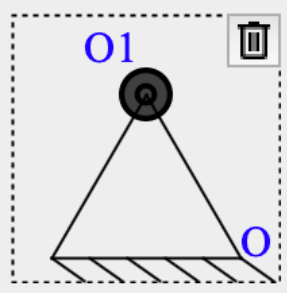


Рис 2.21. Демонстрація елемента, який був обраний користувачем

Кожну ланку можна видалити з робочої області, якщо натиснути на клавішу видалення.

В робочій області можна обрати декілька шарнірів для подальшого з’єднання, щоб обрати шарнір, треба натиснути на нього лівою клавішею миші.

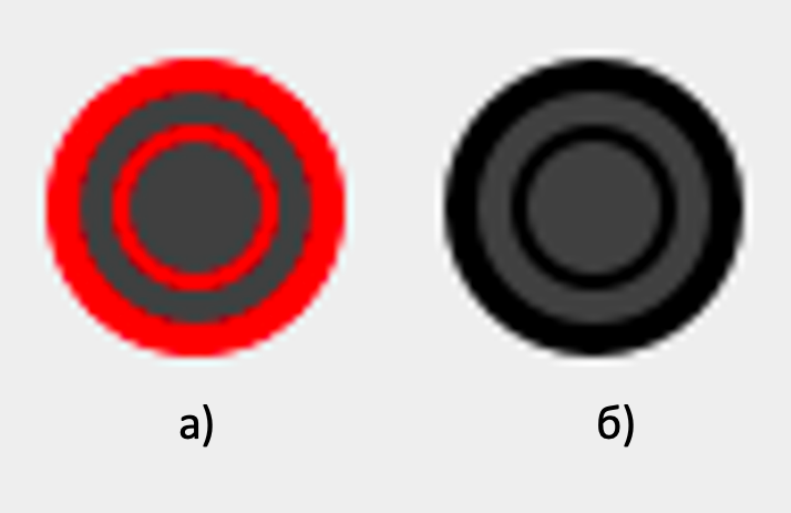


Рис 2.22. Зображення шарнірів: а) вигляд обраного користувачем шарніру, б) звичайний вигляд шарніру

Шарніри можна з’єднувати між собою, для цього потрібно обрати два шарніри та натиснути на одну з кнопок розташованих на верхній панелі.

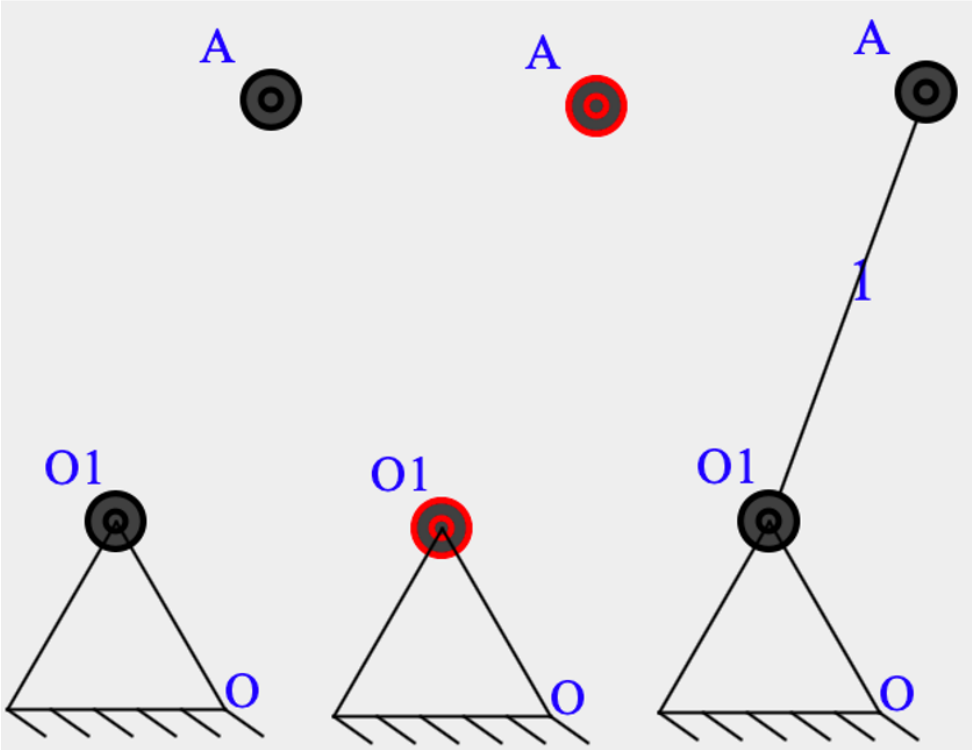


Рис 2.23. З'єднання двох шарнірів між собою

Якщо обрати менше або більше двох шарнірів, у користувача з’явиться вікно попередження, як на (рис. 2.24.).

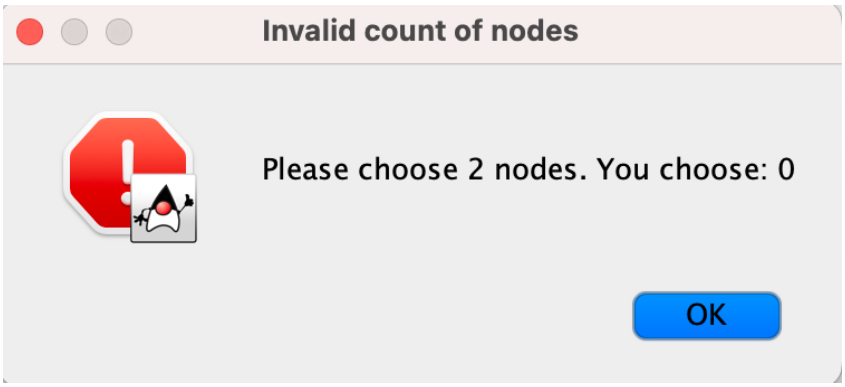


Рис 2.24. Вікно попередження про невірну кількість обраних шарнірів

У разі перетягування користувачем одної з ланок на робочій області, з’єднання між шарнірами будуть перебудовуватись автоматично.

Програмна архітектура даної автоматизованої системи вдало спроектована, вона дає можливість легко розширювати програму, без втручання у вже написаний код.

Мова програмування Java не розрахована для вирішення подібних задач, але через те, що вона має графічні бібліотеки, такі як Swing, це дає змогу вирішувати такі задачі. За допомогою мови Java легко будувати правильну архітектуру програми з застосуванням всіх принципів ООП.

Вибір середовища розробки IntelliJ IDEA прискорив розробку додатку, так як він має зручний та інтуїтивний інтерфейс. IntelliJ IDEA вказує на блоки коду, в яких потенційно може виникнути помилка, що підвищує якість коду.

Інтерфейс програми був побудований за допомогою бібліотеки Swing, яка має різні стилі компонентів, що робить інтерфейс програми сучасним та інтуїтивно зрозумілим.

ВИСНОВОК

На сьогоднішній день, автоматизація задач теорії машин та механізмів обмежена вузьким колом розроблених додатків, переважно це веб-додатки для розрахунків. Огляд наукових праць за темою дипломної роботи показав, що існуючі рішення, окрім обмеженого застосування, теж мають ряд недоліків. Наприклад, вони можуть бути неефективними у використанні для широкого спектру сценаріїв або не забезпечувати достатню гнучкість. Тому, з'явилася необхідність у розробці автоматизованої системи, яка здатна вирішувати ці обмеження та недоліки, і забезпечувати більш гнучкий та універсальний підхід до автоматизації задач теорії машин та механізмів.

У роботі було проведено огляд різних технологій та інструментів розроблення для вирішення завдання автоматизації побудови плоского важільного механізму. Після аналізу було обрано мову програмування Java через свою широку підтримку, багатий набір бібліотек та фреймворків, а також високу надійність та переносимість.

У парадигмі об'єктно-орієнтованого програмування було розроблено автоматизовану систему для побудови плоского важільного механізму. Ця система використовує принципи спадкування, поліморфізму та інкапсуляції для створення класів та методів. Завдяки цьому підходу, розроблено систему, яка є гнучкою та розширюваною, що дозволяє зручно впроваджувати нові функціональності та змінювати параметри механізму без значних зусиль.

У рамках реалізації була розроблена функціональність побудови плоского важільного механізму за допомогою комбінації створених об'єктів-ланок. Цей підхід дозволяє складати механізми з окремих елементів і змінювати їх конфігурацію. Комбінація об'єктів-ланок забезпечує високу точність та надійність побудови механізму.

Тож, інженерам та студентам більше не потрібно будувати механізми класичними методами. Вона дає можливість вносити конструктивні зміни в механізм,

не перебудовуючи весь механізм. Також автоматизована система забезпечує точність та надійність проектування плоского важільного механізму.

Розроблена автоматизована система для побудови плоского важільного механізму виявилася ефективним інструментом, який спрощує та прискорює процес розробки механізму. Ця система містить у собі набір функцій та інструментів, що дозволяють автоматизувати процес проектування механізму.

І на останок, існування такої системи в сучасному світі має безліч переваг. По-перше, вона допомагає збільшити продуктивність та швидкість розробки, що є важливим в умовах швидкоплинного технологічного розвитку. По-друге, сприяє зниженню витрат, оскільки дозволяє ефективніше використовувати ресурси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теорії механізмів і машин. Конспект лекцій (напрямку: 6.050502 «Інженерна механіка», 6.050503 «Машинобудування», 6.070106 «Автомобільний транспорт») / укл: Романюк О.Д. – Дніпродзержинськ: ДДТУ, 2016. – 112с.

2. П.П. Анципорович [та ін.] Структура механізмів: навчально-методичний посібник до лабораторних робіт з дисципліни “Теорія механізмів, машин та маніпуляторів”, 2012. 4-те видання. С. 4-9

3. Синтез механізмів з нижчими парами. URL: https://moodle.znu.edu.ua/pluginfile.php/475434/mod_resource/content/1/3-3.pdf (дата звернення: 15.05.23)

4. Лекція 3. Кінематичний аналіз механізмів. URL: http://www.teormach.ua/lect3.htm (дата звернення: 15.05.23)

5. Autodesk Invertor. URL: https://www.autodesk.com/products/inventor/overview?term=1-YEAR&tab=subscription (дата звернення: 19.05.23)

6. Simens NX. URL: https://plm.sw.siemens.com/en-US/nx/ (дата звернення: 19.05.23)

7. CATIA Dassault Systemes. URL: https://www.3ds.com/en/produkty-i-uslugi/catia/ (дата звернення: 19.05.23)

8. PTC Creo. URL:https://www.ptc.com/en/products/creo (дата звернення: 19.05.23)

9. Agile product design with Cloud-Native Cad & PDM. URL: https://www.onshape.com/en/ (дата звернення: 19.05.23)

10. Fusion 360 Overview. URL: https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription (дата звернення: 19.05.23)

11. SimScale cloud-native simulation platform. URL: https://www.simscale.com/ (19.05.23)

12. Моделювання та кінематичний аналіз кривошипно-шатунного механізму, н.с. Ащепкова, вісник НТУ "ХПІ" , 30.11.2014, № 62 (1104), с 4-12

13. Кінематичний аналіз плоского важільного механізму програмними засобами, Олександр Ступак, Сергій Цибульник, Матеріали IV Всеукраїнської науково-технічної конференції теоретичні та прикладні аспекти радіотехніки, приладобудування і комп'ютерних технологій 2019, с 241-244

14. What is Java technology and why do I need it? URL: https://www.java.com/en/download/help/whatis_java.html (дата звернення: 24.05.23)

15. IntelliJ IDEA, URL: https://www.jetbrains.com/idea/ (дата звернення: 27.05.23)

16. About the JFC and Swing, URL: https://docs.oracle.com/javase/tutorial/uiswing/start/about.html (дата звернення: 24.05.23)

17. Java Swing Tutorial, URL: https://www.javatpoint.com/java-swing (дата звернення: 27.05.23)

18. Cay S. Horstmann Core Java Volume 1 – Fundamentals, 11th edition, May 15 2018 – Prentice Hall

19. Тестування. Фундаментальна теорія, URL: https://dou.ua/forums/topic/13389/ (дата звернення: 27.05.23)