ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ «ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ» ІМЕНІ ІВАНА ПУЛЮЯ

Циклова комісія програмних систем і комплексів

**КУРСОВА РОБОТА**

з дисципліни:

**«Об’єктно-орієнтоване програмування»**

на тему: **«Розробка програмного забезпечення «Конструктор ПК»»**

Студента  3  курсу групи КН-321 спеціальності 122 «Комп’ютерні науки»

Олійник Д.С.

(прізвище та ініціали)

Керівник: викладач   Слободян  Р.О.         
Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_ Р.О. Слободян

(підпис)

\_\_\_\_\_\_\_\_\_\_ Р.Ю. Болюбаш

(підпис)

м. Тернопіль – 2021

ЗМІСТ

[ВСТУП 5](#_Toc89526147)

[1 ТЕХНІЧНЕ ЗАВДАННЯ 6](#_Toc89526148)

[1.1 Найменування та область застосування 6](#_Toc89526149)

[1.2 Підстави для розробки 6](#_Toc89526150)

[1.3 Призначення розробки 6](#_Toc89526151)

[1.4 Вимоги до програми чи програмного виробу 7](#_Toc89526152)

[1.5 Вимоги до програмної документації 9](#_Toc89526153)

[1.6 Техніко-економічні показники 9](#_Toc89526154)

[1.7 Стадії та етапи розробки 10](#_Toc89526155)

[1.8 Порядок контролю та прийому 11](#_Toc89526156)

[2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ 12](#_Toc89526157)

[2.1 Розробка загальної структури і варіантів використання програми 12](#_Toc89526158)

[2.2 Розробка системи класів 16](#_Toc89526159)

[2.3 Розробка методів 17](#_Toc89526160)

[2.4 Проєктування і опис інтерфейсу користувачаff 17](#_Toc89526161)

[2.5 Опис файлової структури програми 17](#_Toc89526162)

[2.6 Опис структури бази даних програми 18](#_Toc89526163)

[3 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ 19](#_Toc89526164)

[ВИСНОВКИ 20](#_Toc89526165)

[ПЕРЕЛІК ПОСИЛАНЬ 21](#_Toc89526166)

[Додаток А Лістинг файлу «program.pro» 22](#_Toc89526167)

[Додаток Б Лістинг файлу «program.h» 23](#_Toc89526168)

[Додаток В Лістинг файлу «program.cpp» 24](#_Toc89526169)

[Додаток Г Компакт-диск із програмним продуктом 25](#_Toc89526170)

ВСТУП

Текст вступу

# ТЕХНІЧНЕ ЗАВДАННЯ

## Найменування та область застосування

Найменування програми – «Конструктор ПК».

Робоча назва – «PC Constructor».

Область застосування програми – конструювання збірок персональних комп’ютерів з різних складових, з перевіркою їх сумісності, розрахуванням потужності і ціни збірки.

## Підстави для розробки

Підставою для проведення розробки являється індивідуальне завдання на курсову роботу з дисципліни «Об’єктно-орієнтоване програмування».

Найменування теми курсової роботи – «Розробка програмного забезпечення «Конструктор ПК»».

Замовник – керівник курсової роботи, викладач Слободян Руслан Олесьович.

Виконавець ­– студент групи КН-321 Олійник Денис Сергійович.

## Призначення розробки

Призначенням програми є спрощення і вдосконалення процесу відбору комплектуючих для ПК, конструювання власних збірок, моделювання сумісності комплектуючих. Припускається, що основним сегментом користувачів є будь-які люди або компанії, перед якими стоїть задача зібрати ефективний, сумісний і робочий ПК з мінімальними витратами часу та грошей.

Також передбачається зручне зберігання необхідних для функціонування сучасного ПК комплектуючих та наданні швидкому доступі до кожного потрібного компоненту з можливістю перегляду його зовнішнього вигляду та характеристик.

## Вимоги до програми чи програмного виробу

### Вимоги до функціональних характеристик

Виконання поставленої задачі вимагає, щоб конструктор ПК забезпечував зручне та зрозуміле представлення великої маси комплектуючих комп’ютера та їх групування. Також обов’язковим є послідовне конструювання збірки з дотриманням сумісності компонентів, як-от:

* сокет у материнської плати та центрального процесора;
* тип оперативної пам’яті у материнської плати та безпосередньо мікросхеми оперативної пам’яті;
* тип інтерфейсу обміну даними у материнської плати та накопичувача;
* можливість блоку живлення забезпечити усі потреби і споживання інших компонентів збірки.

Програма повинна застерігати користувача від випадкової втрати даних, наприклад, збірки, і вважати конструкцію закінченою лише тоді, коли наявні обов’язкові компоненти:

* материнська плата;
* центральний процесор;
* оперативна пам’ять;
* хоча б один накопичувач;
* якщо центральний процесор не містить графічного ядра, то хоча б одна відеокарта;
* блок живлення.

Вхідні дані повинні опрацьовуватись зрозумілим та раціонально змодельованим графічним інтерфейсом, а вихідні зберігатися у найбільш оптимальному представленні організації даних, як-от база даних.

Програма повинна надавати користувачу можливість самому вносити у базу даних комплектуючі, які його цікавлять, редагувати їх, видаляти зайві.

### Вимоги до часових характеристик

Оскільки програма повинна надавати доступ до функціоналу використовуючи графічний інтерфейс, то запуск програми мусить відбуватися за часом до 1 секунди, а відгук інтерфейсу бути майже моментальним, щоб не створювалося відчуття «провисання» програми.

### Вимоги до надійності

Конструктор повинен застерігати користувача від введення неправильних вхідних даних, їх випадкового видалення, забезпечувати їх надійне зберігання і виключати можливість втрати даних. Весь функціонал забезпечення надійності повинен супроводжуватися зрозумілими для користувача повідомленнями, які буде виводити графічний інтерфейс.

### Умови до експлуатації

Для експлуатації даної програми вистачить базових навичок користування персональним комп’ютером, але для ефективного використання конструктора за призначенням потребується знання архітектури ПК, характеристик його комплектуючих для конструювання збірки та внесення нових компонентів у базу.

### Вимоги до складу і параметрів технічних засобів

В склад технічних засобів входять:

* монітор з діагоналлю не менше 13" та роздільністю не менше 1280 на 1024 пікселів, рекомендується широкий формат, на кшталт 16:9 або 16:10;
* комп’ютерні клавіатура та мишка;
* персональний комп’ютер з операційною системою, не старішою за Microsoft Windows 7 або Linux/MacOS (з перезбиранням конструктора на цільових ОС) з такими мінімальними характеристиками:
  + процесор з частотою ядра не менше 2ГГц;
  + оперативна пам’ять з не менше ніж 1ГБ пам’яті;
  + накопичувач з обсягом не менше 20ГБ.

### Вимоги до інформаційної і програмної сумісності

Вихідні коди програми повинні бути реалізовані мовою програмування С++ зі стандартом не старішим за стандарт ISO C++11, а в якості середовища розробки повинне використовуватись середовище, яке підтримує С++11 та володіє можливістю інтегрування фреймворку Qt не старіше версії 5.12.

Системні програмні засоби та утиліти повинні забезпечуватись операційною системою, яка відповідає вимогам технічних засобів.

Вимоги до захисту інформації і програми не пред’являються.

## Вимоги до програмної документації

По закінченню розробки програмного забезпечення потрібно підготувати таку документацію:

* інструкція інсталяції програми;
* загальні відомості про можливості програми;
* інструкція з експлуатації.

## Техніко-економічні показники

Розрахунок економічної ефективності і вартості розробки програмного продукту не проводиться.

Приблизне число використань програми в рік визначити не представляється можливим і є індивідуальним для кожного користувача в залежності від його потреб.

## Стадії та етапи розробки

Розробка конструктора є комплексною і підходить під визначення каскадної моделі розробки. Весь процес розробки програмного забезпечення буде містити такі стадії:

* Аналіз.

Перший етап являє собою безпосередній аналіз поставленої задачі та її предметної області. Формується загальне уявлення, як програма повинна працювати та взаємодіяти з користувачем, як саме вона повинна задовольняти вимогам.

* Проєктування.

На даному етапі формуються макети, вже реальна реалізація того, як приблизно зовнішньо має виглядати графічний інтерфейс програми, розробляються класи, моделюються зв’язки між ними. Застосовуються такі засоби ООП, як спадкування, інтерфейс, поліморфізм, створюється ієрархія класів. Визначається тип підключення об’єктів класів – статичний чи динамічний. Формується структура бази даних та загальний вигляд її таблиць.

* Реалізація.

Логіка та функціонал послідовно описуються мовою програмування, для їх реалізації застосовуються різні засоби С++ та фреймворку Qt. Написаний код на даному етапі повинен поверхнево відповідати вимогам та загальній парадигмі ООП, важливим принципам та/або паттернам проектування, але все ж допускаються виключення, приорітет робиться на реалізації функціоналу.

* Рефакторинг.

Програмний код переглядається та оптимізовується, де це потрібно – переписується. Усі іменування об’єктів, змінних та інших ідентифікаторів даних за потреби перейменовуються так, щоб код «пояснював сам себе». За потреби додаються коментарі. Відбувається усунення дубльованого коду, спрощення невиправдано ускладнених блоків. В останню чергу переглядається та за потреби редагується зовнішьої частини програми – вікон графічного інтерфейсу, їх дизайну.

* Тестування.

Останній етап передбачає проходження розробленим ПЗ тестування і налагодження: на вхід подаються найбільш різноманітні вхідні дані, оглядається реакція програми на них, обробка некоректних дій користувача. Аналізується, наскільки програма відповідає вимогам і реалізовує потрібний функціонал. За потреби до програмного коду вносяться корективи. Для пошуку помилкової поведінки програми використовуються попередження компілятора, зневаджувач.

Після проходження етапу тестування оформлюється пояснювальна записка.

## Порядок контролю та прийому

Прийом розробленого програмного забезпечення повинен відбуватися на об’єкті Замовника в терміни, які зазначені в індивідуальному завданні.

Для прийому роботи Виконавець повинен представити:

* діючу програму, яка повністю відповідає даному технічному завданню;
* вихідний програмний код, записаний разом із програмою на носій інформації.

Прийом програмного забезпечення повинен відбуватися перед комісією з двох чоловік (один з яких - Замовник) у такій послідовності:

* доповідь Виконавця про виконану роботу;
* демонстрація Виконавцем роботи програми;
* контрольні випробування роботи програми;
* відповіді на запитання і зауваження комісії.

# РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

## Розробка загальної структури і варіантів використання програми

Даний та наступні підрозділи послідовно описують розробку ПЗ згідно каскадної моделі розробки та попередньо зазначених етапів.

Перш за все стояла задача дослідити предметну область програми, а саме комп’ютерну інженерію або архітектуру. Було набуто нові знання і закріплено вже існуючі з галузі будови ПК, після чого визначалися ключові питання, відповіді на яких дають змогу моделювати вже безпосередньо взаємодію програми та користувача:

* хто буде виступати в ролі суб’єктів програми, скільки їх буде та яка буде їхня роль, як програма буде з ними взаємодіяти;
* хто буде виступати в ролі користувачів та скільки їх буде;
* який функціонал повинна надавати програма користувачу та у який спосіб;

Послідовно на ці питання було дано відповіді: в ролі діючих суб’єктів (надалі - користувачів) буде виступати будь-яка людина чи група людей, оскільки цільова категорія користувачів є широкою, як було вказано в розділі 1.3. Методом взаємодії програми та користувача був обраний інтерфейс як найбільш зручний засіб, який надає користувачу можливість керування програмою без випадкового внесення змін в конструктор та аналізом, виправленням некоректного користувацького введення.

Також, суб’єктом програми буде операційна система, яка і повинна забезпечити функціонування програми на комп’ютері. Засобом взаємодії програми та ОС виступає мова програмування та/або фреймворк, які дозволяють використовувати системний функціонал та керувати ним для вирішення задачі.

Як і було вказано, в ролі користувачів може виступати будь хто, і оскільки задача не передбачає створення контролю облікових записів (наприклад, адміністратор - користувач) та їх контролю, то з цього випливає, що користувачів може бути декілька і програма повинна забезпечувати багатокористувацький доступ до функціоналу.

Програма повинна давати користувачу такі можливості згідно технічного завдання:

* створення збірки ПК, її редагування, видалення, перегляд;
* створення декількох збірок ПК та навігація між ними;
* внесення, видалення, редагування, перегляд комплектуючих;
* перевірка сумісності обраних комплектуючих збірки;
* запобігання некоректним діям користувача;
* надійну та безперебійну роботу програми;
* вираховування потужності та ціни кожної збірки;

В свою чергу, користувач буде виконувати такі дії для взаємодії з конструктором, і деякі з них можна розбити на підпункти, як на рисунку 2.1:

* запуск і вимкнення програми;
* створення, редагування, видалення збірок;
* створення, редагування, видалення комплектуючих;
* розрахування потужності та ціни збірки;

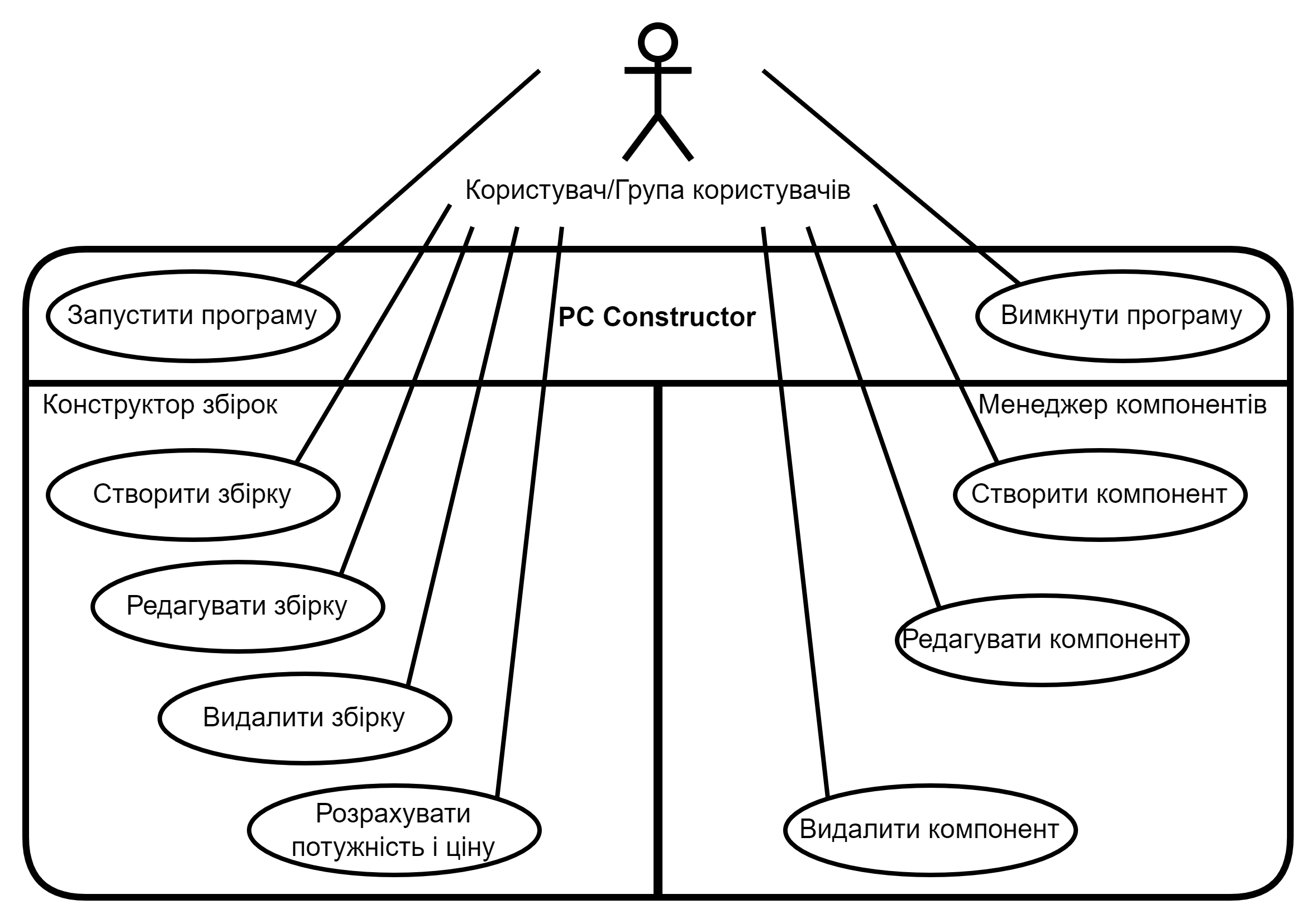


Рисунок 2.1 – Діаграма варіантів використання програми PC Constructor

Кожен з вказаних варіантів потрібно описати, щоб розуміти, як їх розробляти засобами мови програмування.

***PC Constructor:***

* ***Запустити програму.*** Точка входження у програму. Повинен вивестись інтерфейс Конструктора з можливістю переходу у наступні варіанти або в Менеджер компонентів і його використання.
* ***Вимкнути програму.*** Програма закінчує своє виконання, зберігаючи всі дані та звільняючи зайняту оперативну пам’ять.

***Конструктор збірок:***

* ***Створити збірку.*** Користувачу пропонується створити збірку і назвати її. Надалі йдуть різні сценарії: якщо користувач вводить назву і погоджується створити збірку і вона створюється, то створена збірка стає активною і з нею можна працювати. Якщо користувач відмовляється, або назва є некоректною, або така збірка вже існує, то збірка не створюється і програма повертається до початкового вікна.
* ***Редагувати збірку.*** При виборі збірки з вже існуючих або після створення збірки така збірка стає активною. Інтерфейс надає елементи для додавання, зміни, видалення компонентів. Сценарії: якщо змінити збірку, то така збірка стає активною, а попередня залишається в так званому «сплячому стані», щоб можна було повернутися для роботи з нею. Послідовно додаються компоненти, якщо якийсь з них не є сумісним, то користувач отримує помилку, якщо він вибере сумісний компонент, то зможе далі збирати ПК. Збірку можна закрити і тоді активної збірки не буде. Виводиться інформація про готовність збірки. Недоступно, якщо немає активної збірки.
* ***Видалити збірку.*** Програма питає користувача, чи точно він хоче видалити збірку. Якщо він погоджується, то збірка видаляється і інформація про неї стирається, а активною збіркою стає остання «спляча» збірка, або, якщо таких немає, то жодна. Якщо відмовляється, то збірка не видаляється і можна продовжувати роботу. Недоступно, якщо немає активної збірки.
* ***Розрахувати потужність і ціну.*** Виводиться інформація про потужність і ціну активної збірки з компонентами, які вже є обраними. Недоступно, якщо немає активної збірки.

***Менеджер компонентів:***

* ***Створити компонент.*** Попередньо обирається активна група компонентів. Створюється компонент, дані про якого вносить у базу сам користувач і може завантажити фотографію по бажанню. Якщо компонент з таким ім’ям вже існує, то компонент не створиться.
* ***Редагувати компонент.*** Користувач може змінювати характеристики компонента, проте, якщо він змінить його найменування і воно буде повторювати вже існуюче у базі іншого компонента, то зміна не внесеться. Недоступно, якщо нема активної групи компонентів.
* ***Видалити компонент.*** З’являється вікно, яке питає, чи точно користувач хоче видалити компонент. Якщо він погоджується, компонент видаляється, в іншому випадку залишається. Недоступно, якщо нема активної групи компонентів.

На даному етапі також доцільним є сформувати UML-діаграми дій цих сценаріїв для того, щоб більш конкретизувати їх поведінку та спросити їх реалізацію.

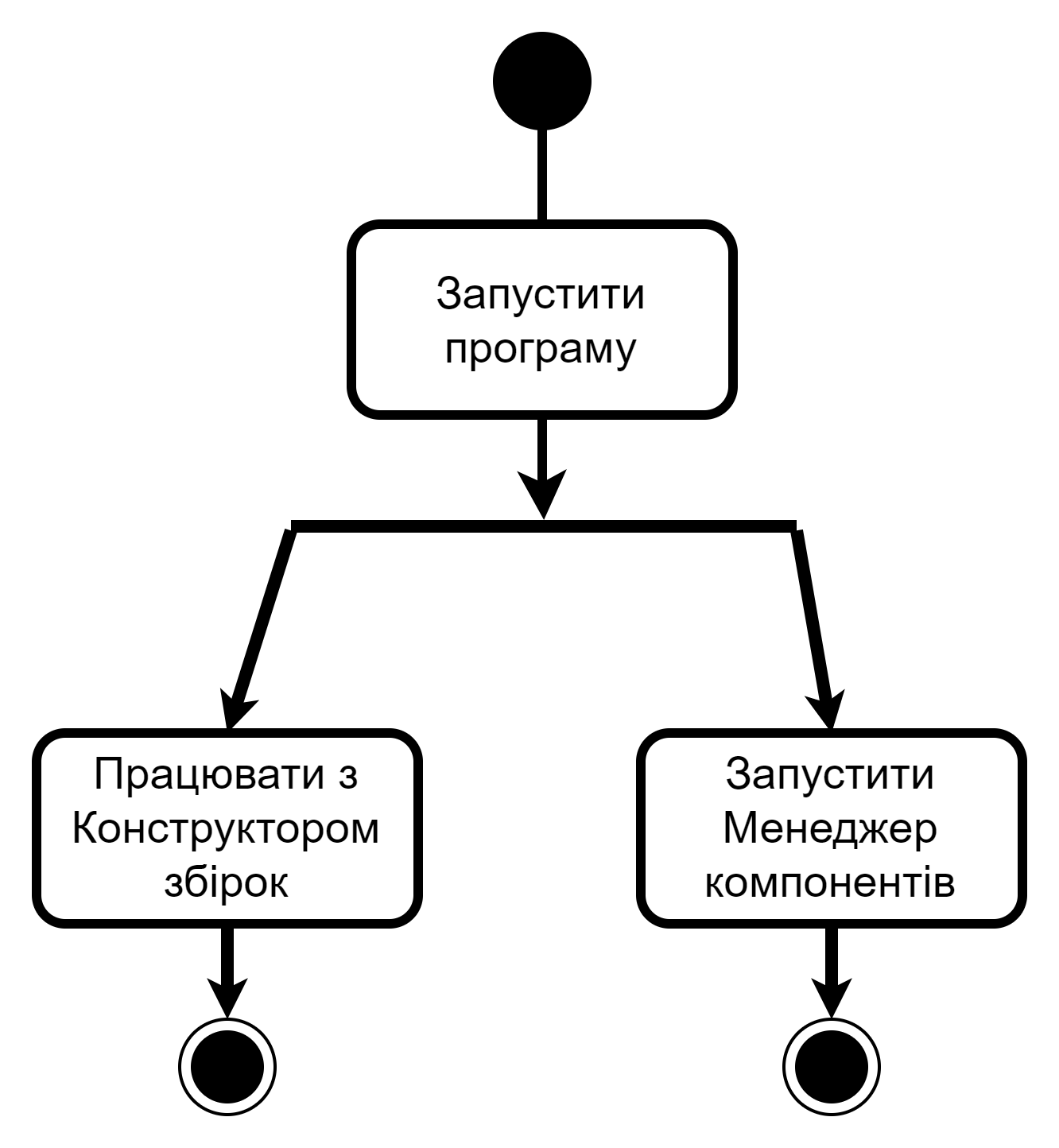


Рисунок 2.2 – Діграма дій при запуску програми

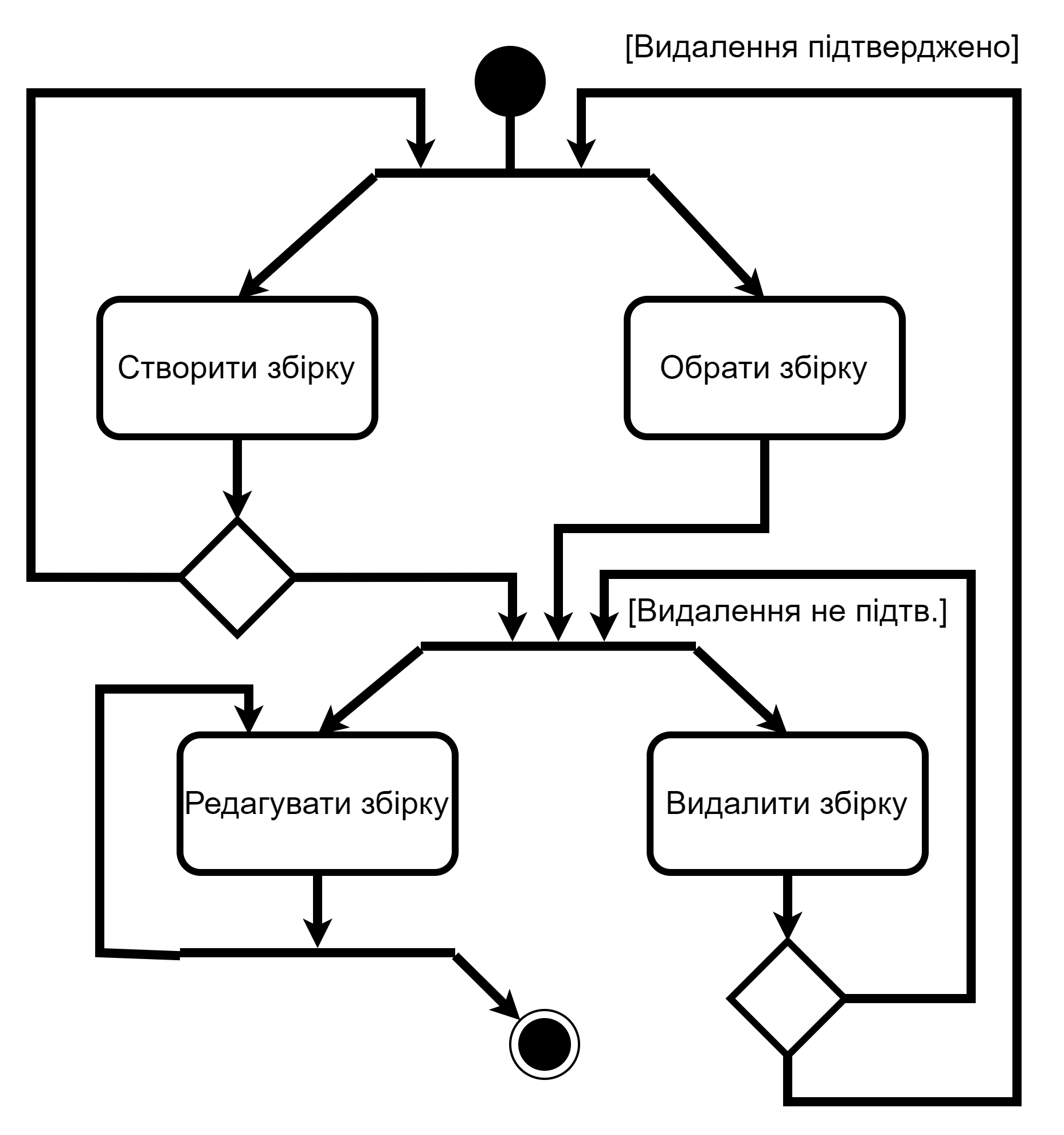


Рисунок 2.3 – Діаграма дій в Конструкторі збірок

## Розробка системи класів

Визначивши загальну модель поведінки програми, будується UML-діаграма класів, яка зображуватиме об’єктно-орієнтовану будову програми та зв’язки між її класами.

Згідно розробленої структури, записуємо ключові ідентифікатори, які будуть використовуватися в програмі та взаємодіяти між собою:

* Головне вікно конструктора;
* База даних;
* Менеджер компонентів;
* Діалогове вікно створення збірки;
* Активне ім’я, ідентифікатор збірки;
* Модель таблиці;
* Віджет компонентів;
* Список компонентів;
* Конфлікт компонентів;
* Активне ім’я, тип, ідентифікатор компоненту;
* Віджет-переглядач характеристик компоненту;
* Повідомлення про помилки.

Діаграма будується за допомогою програми StarUML, вказуються зв’язки між класами: наслідування, агрегація, композиція, залежність. Записуються усі атрибути та методи, їх модифікатор доступу, чи є той чи інший метод статичним, віртуальним.

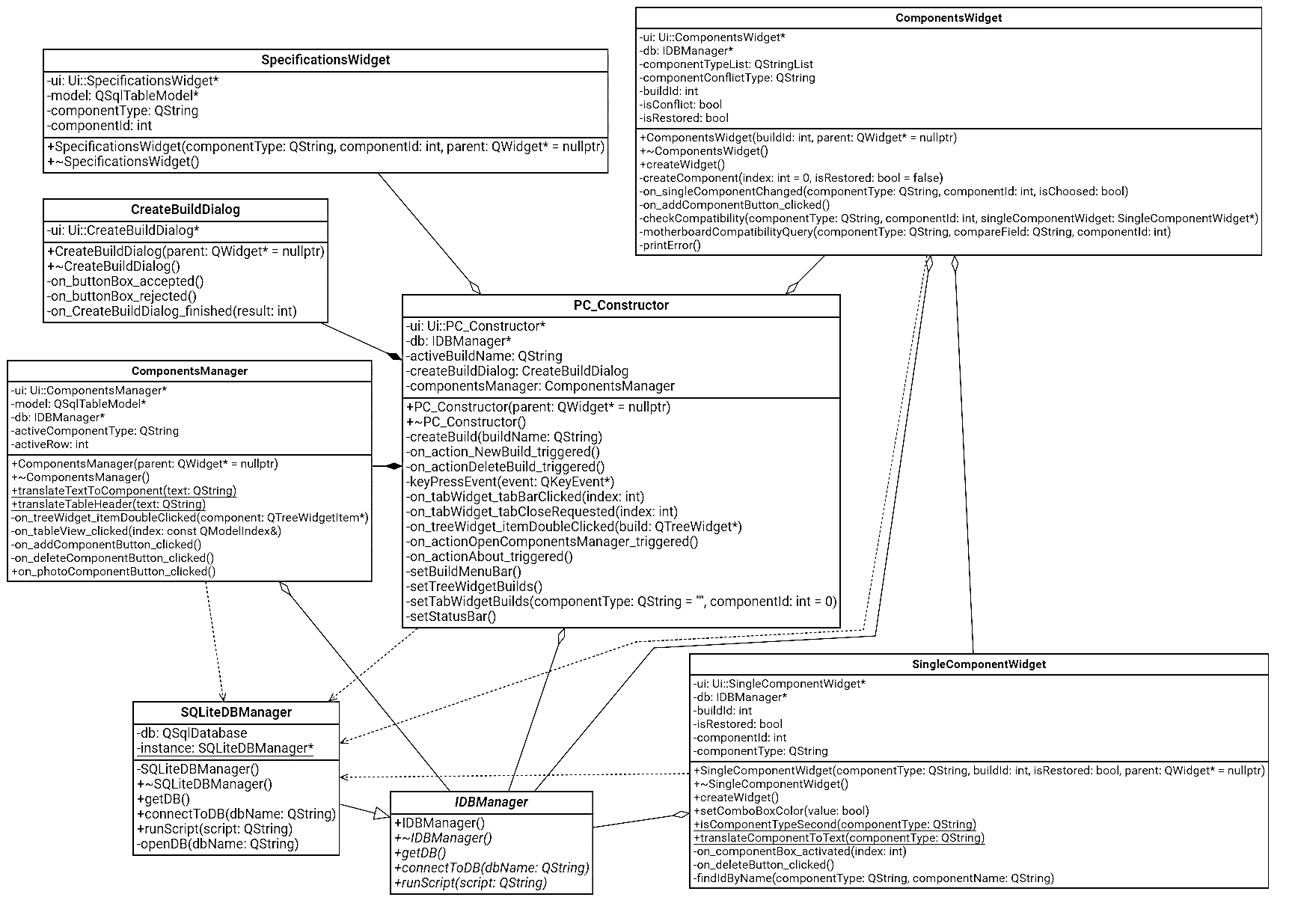


Рисунок 2.4 – UML-діаграма класів програми PC Constructor

Згідно моделі та спроектованих сценаріїв взаємодії користувача і програми, роботи програми, можна виділити, що визначені раніше головні іменники програми сформувалися в одну монолітну структуру і є взаємозв’язаними: наприклад, всі класи, які працюють з базами даних, використовують як оболонку для доступу до бази даних інтерфейс IDBManager, який, в свою чергу, має свою реалізацію у виді похідного класу SQLiteDBManager, що видно на рисунку 2.4. Оскільки інтерфейс – лише сигнал для програми, що є деяка абстракція, яка дає доступ до бази даних, проте існує й інша реалізація, в нашому випадку похідний клас, динамічний об’єкт якого і буде записуватись у вказівник типу IDBManager. Це є зручним, якщо, наприклад, потрібно буде замінити СКБД SQLite на, наприклад, MySQL: достатньо буде лише створити окрему реалізацію інтерфейсу та підключити її один-два рази, не шукаючи слідів іншого менеджеру по всій програмі.

Наглядно видно, що класи реалізовують свій функціонал модульно і кожен має свою зону відповідальності: наприклад, ComponentsManager лише надає можливість працювати безпосередньо з базою даних, а ComponentsWidget і SpecificationsWidget – переглядати та редагувати збірки. І лише в головному класі PC\_Constructor вони організовуються у єдину систему.

На основі діаграм, наведених на рис. 2.2, 2.3 і 2.4 вже представляється можливим написання коду програми та реалізації задумки, вирішення задачі.

Створюється проект в інтегрованому середовищі розробки Qt Creator версії 5.0.1, обирається робочим фреймворк Qt версії 6.1 та стандарт мови C++17. Відповідно до визначених моделей створюються С++-класи у вигляді відповідних .h (заголовних, надалі - хедерів) та .cpp (реалізації) файлів.

Послідовно у хедери вносяться оголошення класів, їх атрибутів і методів, параметрів, вказується тип значень, який повертає кожен метод.

## Розробка методів

У файлах реалізації створюються відповідно до оголошень тіла методів, а також файл main.cpp, який буде містити головну функцію main() та точку входу для програми, в ролі якої виступає об’єкт класу фреймворку QApplication і нашого класу PC\_Constructor:

#include <QApplication>

#include "pc\_constructor.h" // Хедер головного вікна програми

int main(int argc, char \*argv[])

{

QApplication application(argc, argv);

PC\_Constructor mainWindow;

mainWindow.show();

return application.exec();

}

Лістинг 2.1 – main.cpp

При запуску

## Проєктування і опис інтерфейсу користувача

## Опис файлової структури програми

## Опис структури бази даних програми

# ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

ВИСНОВКИ

Текст

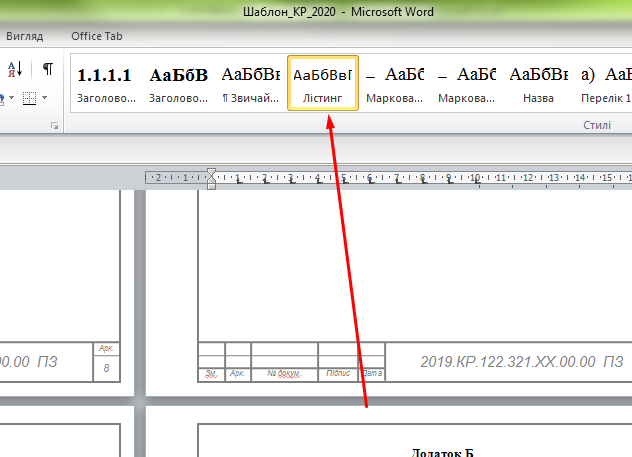
ПЕРЕЛІК ПОСИЛАНЬ

1. Грицюк Ю.І. Рак Т.Є. Об’єктно-орієнтоване програмування мовою С++ : навч. посіб. Львів : Вид-во Львівського ДУ БЖД, 2011. 404 с.
2. Вайсфельд М. Объектно-ориентированное мышление. СПб. : Питер, 2014. 304 с.
3. Build with Qt. URL: https://www1.qt.io/built-with-qt (дата звернення: 15.12.2019).
4. EVILEG : вебсайт. URL: https://evileg.com/uk/knowledge/qt (дата звернення: 15.12.2019).

Додаток А  
Лістинг файлу «program.pro»

Додаток Б  
Лістинг файлу «program.h»

/////////////// Лістинги програм оформляти цим шрифтом!!!! ///////////////



/////////////////////підключення бібліотек//////////////////////

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <graphics.h>

#include <dos.h>

#include <math.h>

#include <dir.h>

/////////////// Лістинги програм оформляти цим шрифтом!!!! ///////////////

. . .

/////////////////////Оголошення класу ... /////////////////////

Додаток В  
Лістинг файлу «program.cpp»

Додаток Г  
Компакт-диск із програмним продуктом