Міністерство освіти і науки України

Інженерний інститут

Запорізького національного університету

Кафедра програмного забезпечення автоматизованих систем

**КУРСОВА РОБОТА**

**з курсу Операційні системи**

**тема:** Програма збору інформації про систему

Спеціальність 121 „Інженерія програмного забезпечення”

Виконав студент

групи ІПЗ-16-бд. Розов Б.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник роботи

Лимаренко Ю.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: доц. Лимаренко Ю.О.

Запоріжжя

2018 р.

# Реферат

* Об'єм роботи – 26 сторінок
* Кількість рисунків – 8
* Лістинги – 5
* Кількість використаних джерел – 3

Метою написання курсової роботи з предмету "Операційні системи" було створення утиліти, що дає можливість переглянути різну інформацію про систему, таку як: наявні девайси, встановлене програмне забезпечення та інше.

В результаті розробки був отриманий програмний продукт, створений на мові програмування C++ , що дозволяє виводити інформацію про систему

Ключові слова: програмний застосунок, інформація, обробник, компонент

# Зміст

[1. Реферат 2](#_Toc11880734)

[2. Зміст 3](#_Toc11880735)

[4. Вступ 5](#_Toc11880736)

[4.1. Глосарій 5](#_Toc11880737)

[4.2. Постановка задачі 6](#_Toc11880738)

[4.3. Огляд існуючих методів рішення 7](#_Toc11880739)

[4.3.1. Аналогічні рішення 7](#_Toc11880740)

[4.3.2. Висновок 10](#_Toc11880741)

[4.4. План робіт 10](#_Toc11880742)

[5. Вимоги до оточення 11](#_Toc11880743)

[5.1. Вимоги до апаратного забезпечення 11](#_Toc11880744)

[5.2. Вимоги до програмного забезпечення 11](#_Toc11880745)

[5.3. Вимоги до користувачів 11](#_Toc11880746)

[6. Функціональні вимоги 12](#_Toc11880747)

[6.1. Загальні вимоги 12](#_Toc11880748)

[7. Архітектура системи 13](#_Toc11880749)

[7.1. Бібліотека класів 15](#_Toc11880750)

[8. Специфікація даних 17](#_Toc11880751)

[8.1. Опис формату та/або структури даних 17](#_Toc11880752)

[9. Вимоги до інтерфейсу 18](#_Toc11880753)

[10. Інші вимоги 19](#_Toc11880754)

[10.1. Вимоги до надійності 19](#_Toc11880755)

[10.2. Вимоги до продуктивності 19](#_Toc11880756)

[11. Проект програмної системи 20](#_Toc11880757)

[11.1. Засоби реалізації 20](#_Toc11880758)

[11.2. Модулі і алгоритми 20](#_Toc11880759)

[11.3. Проект інтерфейсу 23](#_Toc11880760)

[11.4. Реалізація і тестування 24](#_Toc11880761)

[12. Висновки 25](#_Toc11880762)

[13. Список літератури 26](#_Toc11880763)

# Вступ

Існує декілька вбудованих програм що дозволяють переглянути необхідну користувачу інформацію про систему: Програми та компоненти, диспетчер задач, диспетчер приладів.

На відміну від вбудованих програмних засобів, розроблюваний програмний продукт дає можливість переглянути усю необхідну у більшості випадків інформацію використовуючи лише одну застосунок.

## Глосарій

**Common Language Runtime**, скорочено **CLR** — «загальномовне виконуюче середовище» — це компонент пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

**Windows Management Instrumentation** (**WMI**) в дослівному перекладі — інструментарій управління Windows. WMI — це одна з базових технологій для централізованого управління і стеження за роботою різних частин комп'ютерної інфраструктури під управлінням платформи Windows.

Тест продуктивності, або **бенчмарк** (англ. benchmark) — контрольне завдання, необхідне для визначення порівняльних характеристик продуктивності комп'ютерної системи. Існують програми для тесту продуктивності системи, що тестують час автономної роботи ноутбуків і КПК, радіус дії бездротової мережі, пропускну здатність каналів передачі даних, АЧХ звукового тракту та інші доступні для вимірювання характеристики, що не пов'язані напряму з продуктивністю.

## Постановка задачі

Система повинна відображати необхідний перелік інформацію про комп’ютер та програмного забезпечення користувача відповідно, а саме: інформація про компоненти системи, присутні в оперативній пам’яті процеси, встановлене програмне забезпечення, іформацію про внутрішні та зовнішні носії, девайси вводу та виводу, загальну інформацію про операційну систему.

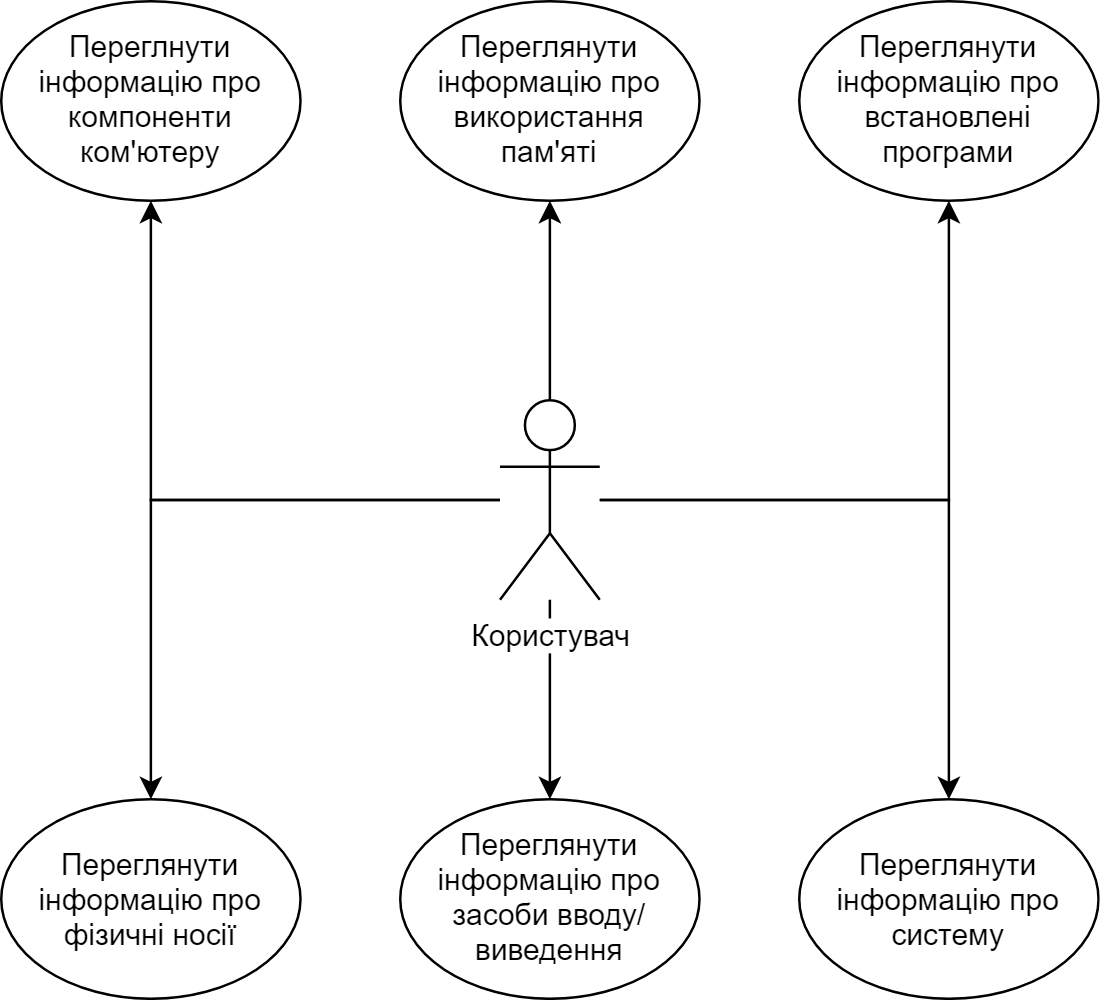


Рис. 1 Діаграма варіантів використання

## Огляд існуючих методів рішення

### Аналогічні рішення

**Aida64** – застосунок компанії FinalWire Ltd. для тестування та ідентифікації компонентів персонального комп'ютера під управлінням операційних систем Windows, що надає детальні відомості про апаратне та програмне забезпечення.

* Основні можливості - детальний аналіз системних пристроїв, моніторинг температури та енергії, різні бенчмарки для перевірки потужності процесору.
* Користувацький інтерфейс - складається переважно с верхнього меню, бокової ієрархічної навігації зліва та основної відображаїмої інформації справа.
* Підтримувані платформи - Windows, Anіdroid, iOS, Windows Phone, Tizen, Chrome OS and Sailfish OS operating systems
* 30 днів пробного періоду, ціна базової версії складає $39.9

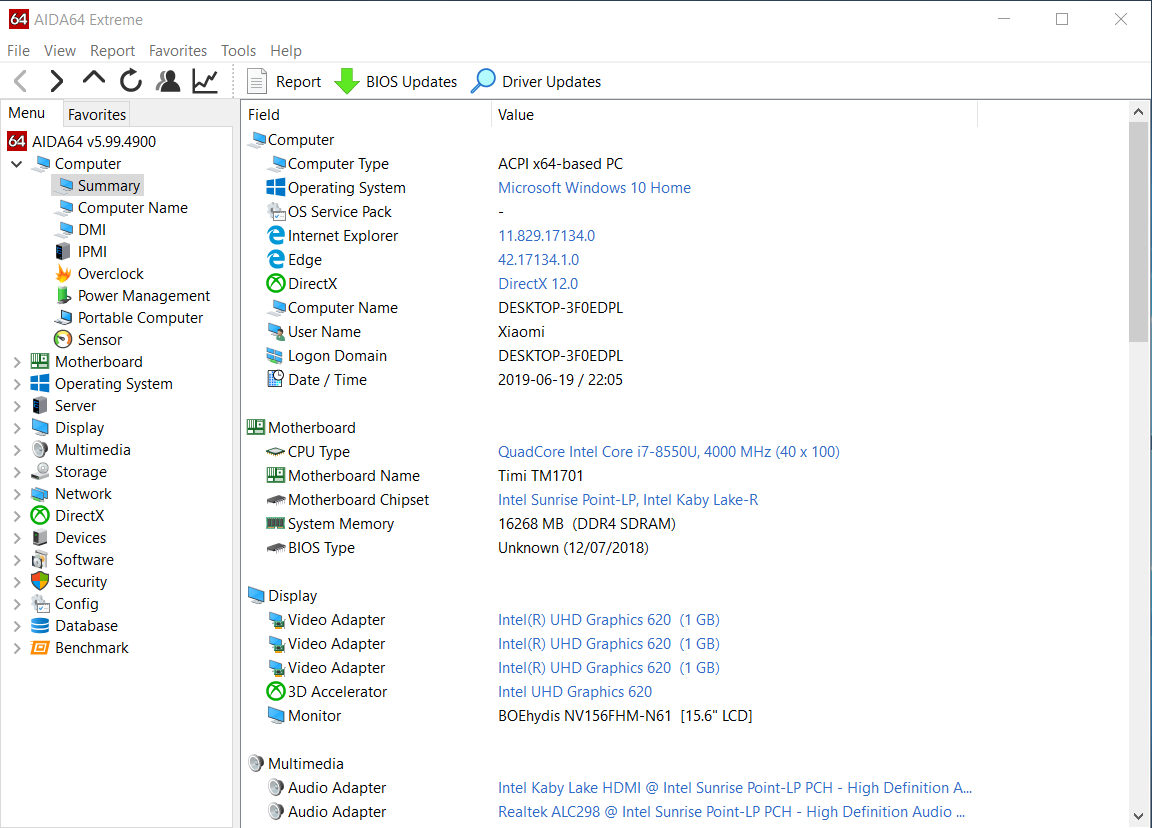


Рис. 2 Користувацький інтерфейс програми AIDA64

**CrystalDiskInfo –** безкоштовна утиліта з відкритим вихідним кодом, що розробляється програмістом з Японії Noriyuki Miyazaki. Призначена для діагностики роботи жорстких дисків ПК. У процесі роботи програми відображається загальна інформація про HDD, ведеться моніторинг значень S.M.A.R.T., а також здійснюється постійний контроль температури диска.

* Основні можливості - детальних моніторинг фізичних (вбудованих та зовнішніх) носіїв
* Користувацький інтерфейс - Меню зверху, перелік системних дисків, відображення стану та інформації про обраний диск.
* Підтримувані платформи - Операційна система Windows починаючи з версії XP
* Open source

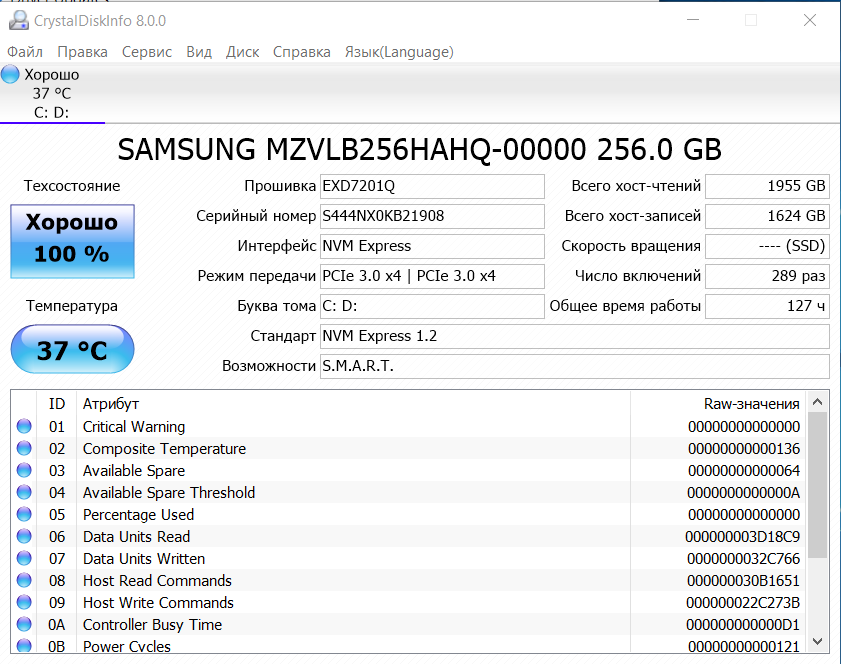


Рис. 3 Користувацький інтерфейс програми CrystalDiskInfo

**CCleaner —** безкоштовна утиліта із закритим вихідним кодом, яка надає користувачам потужний і простий у використанні інструмент для очищення і оптимізації 32- та 64-розрядних операційних систем Microsoft Windows. Утиліта була створена британською приватною фірмою Piriform Limited і написана на C++.

* Основні можливості – моніторинг встановлених програм с можливостю видалення, очистка реєстру та кешу браузерів.
* Користувацький інтерфейс – головне меню зліва та основна частина по центру
* Підтримувані платформи – Windows, Mac OS, Android
* Безкоштовна базова версія, вартість повної - 570.04 гривні

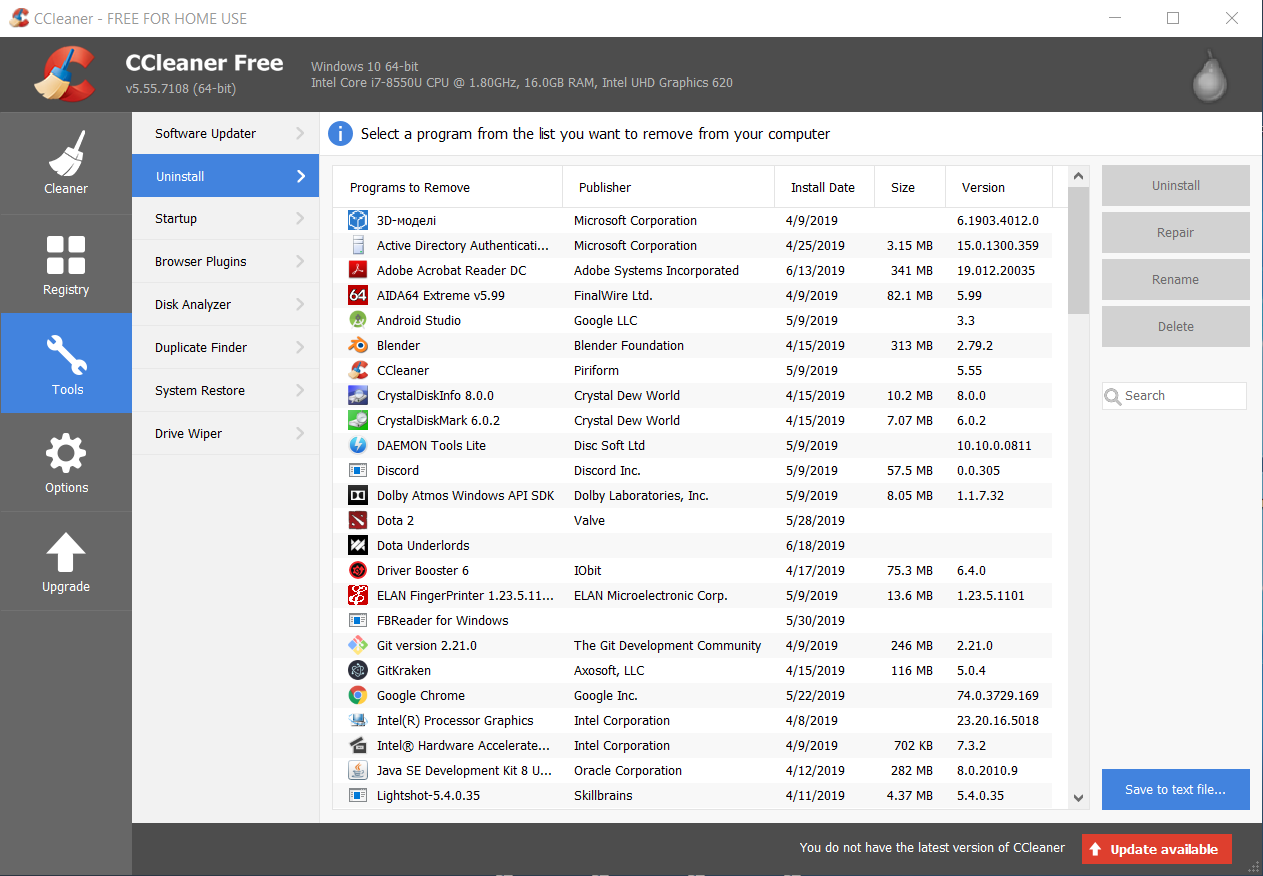


Рис. 4 Користувацький інтерфейс програми CCleaner

### Висновок

На даний момент існую декілька крупних програмних застосунків що надають можливість користування однією або двома необхідними за технічним завданням функціями, проте жодне з існуючих рішень не дає змоги одразу використовувати усі необхідні можливості.

## План робіт

01.02 – 01.03 Розроблення технічного завдання

02.03 – 10.03 Складання плану розробки

11.03 – 25.03 Початок розробки

26.03 Перший реліз (alpha-версія)

27.03 – 10.04 Подальша розробка

11.04 Другий реліз (beta-версія)

12.04 – 20.05 Розробка релізной версії програми

21.05 – 18.06 Тестування релізної версії

19.06 Презентація фінальної версії програмного продукту

# Вимоги до оточення

## Вимоги до апаратного забезпечення

Мінімальні вимоги до апаратного забезпечення:

* **Система** - Windows XP Sр3
* **Процесор** - тактова частотою не менше 300 MHz.
* **Оперативна пам’ять** - 128 Мб.
* **Жорсткий диск** - 100Мб

Рекомендовані вимоги до апаратного забезпечення:

* **Система** - Windows 7
* **Процесор** - 1 ГГц.
* **Оперативна пам’ять** - 1024Мб.
* **Жорсткий диск** - 100Мб
* **Графічний процесор** - 128 Мб пам'яті (адаптер з підтримкою DirectX 9), драйвери WDDM версії 1.0 і старше.

## Вимоги до програмного забезпечення

ОС Windows, починаючи з Windows 7 та вище.

## Вимоги до користувачів

Користувач повинен пройти базовий курс користування операційною системою Windows та розуміти використовуємо термінологію

# Функціональні вимоги

## Загальні вимоги

Система повинна:

1. Дозволяти користувачу:
   1. Переміщення між розділами
   2. Перегляд інформації у кожному розділі
2. Складатися с наступних розділів:
   1. System,
   2. Memory,
   3. Software,
   4. Drives,
   5. Summary,
   6. I/O Devices.
3. Містити наступні підрозділи у розділі System:
   1. Процесор,
   2. Оперативна пам'ять,
   3. Відеосистема,
   4. Логічні розділи,
   5. Шина PCI,
   6. Plug and Play-пристроях,
   7. BIOS і CMOS.

# Архітектура системи

Система складається з шести головних частин:

1. System,
2. Memory,
3. Software,
4. Drives,
5. I/O Devices
6. Summary

Після проведення аналізу вимог було вирішено використовувати наступну архітектуру:

1. Головний клас, що відповідає за обробку подій які відбуваються на екрані
2. Базовий клас, який буде відповідати за спільну, необхідну для отримання інформації, логіку
3. Дочірні класи, які відповідають за необхідну у кожному конкретному випадку логіку.

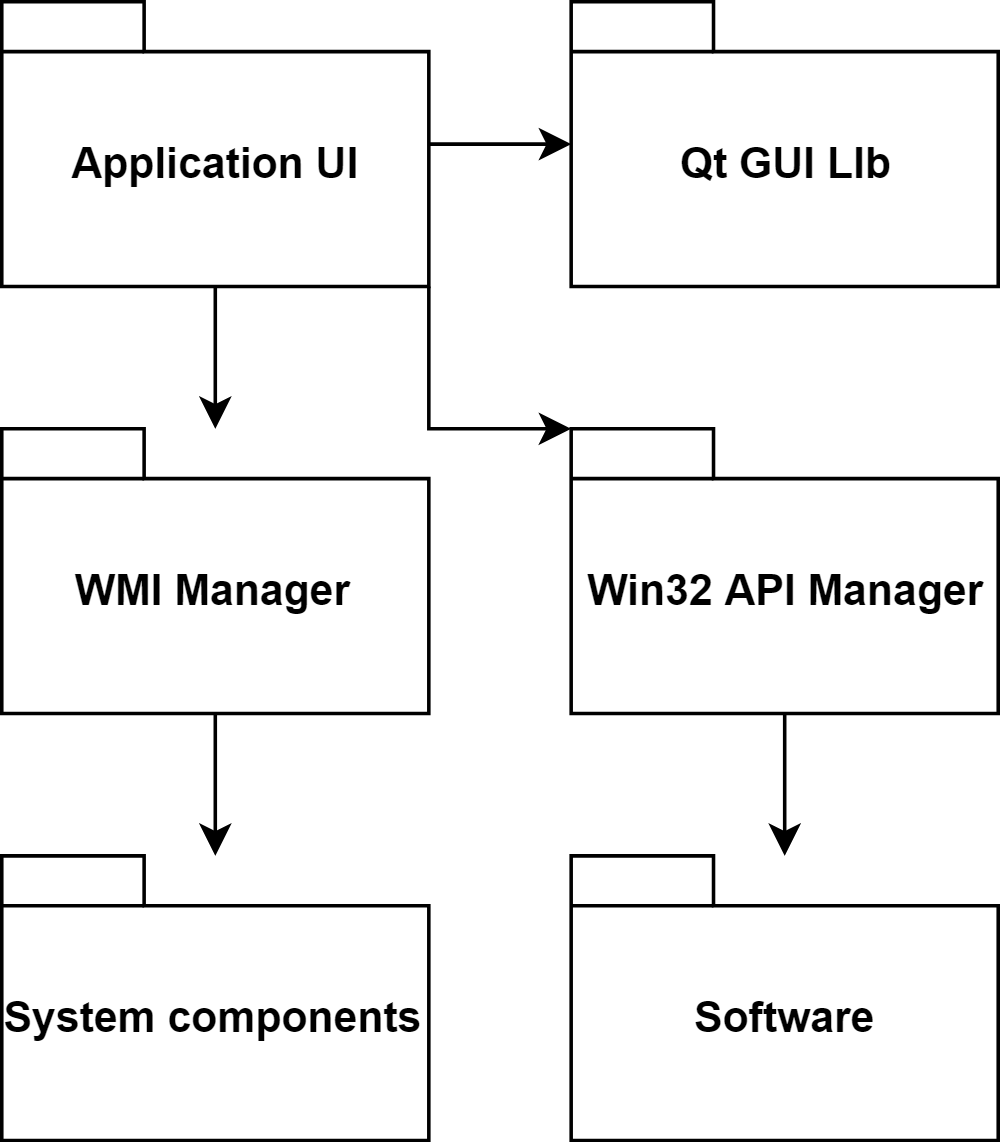


Рис. 5 Діаграма пакетів

## Бібліотека класів

**main** – точка входу до програми, виводить користувацький інтерфейс на екран.

**Boida27** – обробляє події які викликаються інтерфейсом користувача, виводить на екран інформацію отриману із класі які відповідають за генерування інформації для розділів.

**Manager** – базовий клас для обробників інформації розділів. Містить у собі методи отримання інформації про системні компоненти за допомогою CLR та WMI класів.

**SystemManager** – оброблює інформацію для розділу System та його підрозділів, має окремі методи для того щоб отримати інформацію про кожен з підрозділів.

**MemoryManager** – оброблює інформацію для розділу Memory.

**SoftwareManager** – оброблює інформацію для розділу Software.

**DrivesManager** – оброблює інформацію для розділу Drives.

**SummaryManager** – оброблює інформацію для розділу Summary.

**IOManager** – оброблює інформацію для розділу I/O Devices.

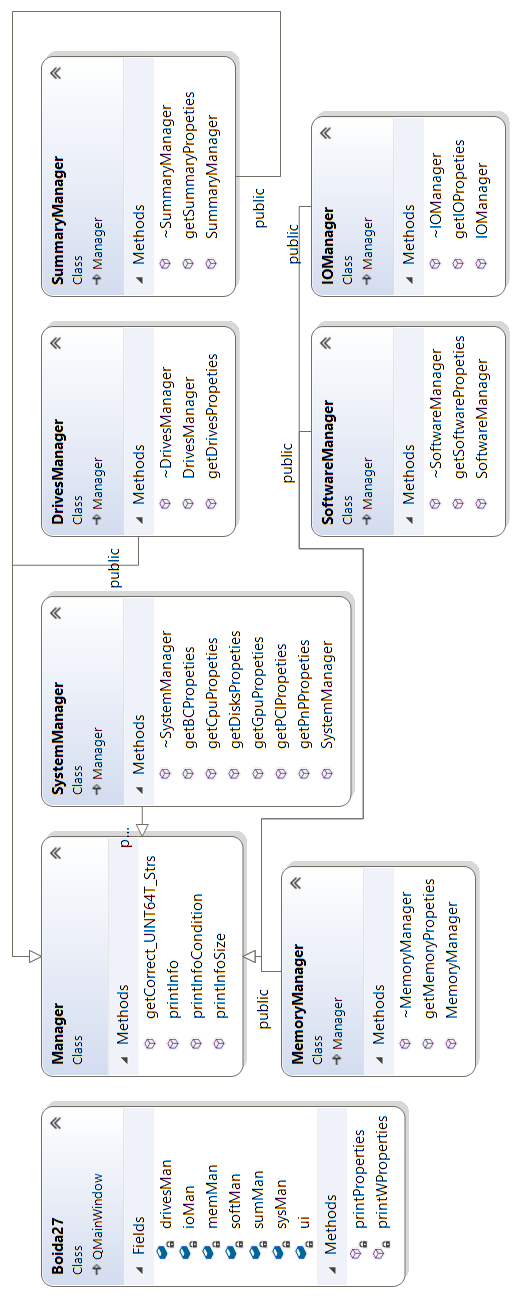


Рис.6 Діаграма класів

# Специфікація даних

## Опис формату та/або структури даних

Отримана за допомогою обробників інформація приводиться до вигляду списку пар текст(назва параметру)-текст(значення параметру) та передається у таблицю як значення для першого та другого стовпчика відповідно.

Лістинг 1. Приклад зберігання інформації

vector<pair<string, string>> SummaryManager::getSummaryPropeties() {

vector<pair<string, string>> gpuProperites;

for each (string str in printInfo("Win32\_VideoController", "Name"))

{

gpuProperites.push\_back(pair<string, string>("Name", str));

}

return gpuProperites;

}

Лістинг 2. Використання списку параметрів для заповнення таблиці

void Boida27::printProperties(vector<pair<string, string>> properties) {

int i = 0;

for each (pair<string, string> p in properties)

{

ui.propertiesTable->insertRow(i);

ui.propertiesTable->setItem(i, 0, new QTableWidgetItem(QString::fromStdString(p.first)));

ui.propertiesTable->setItem(i, 1, new QTableWidgetItem(QString::fromStdString(p.second)));

i++;

}

}

# Вимоги до інтерфейсу

Windows-інтерфейс що надає користувачу змогу перемикатися між різними розділами та переглядати наявну інформацію про кожен з них.

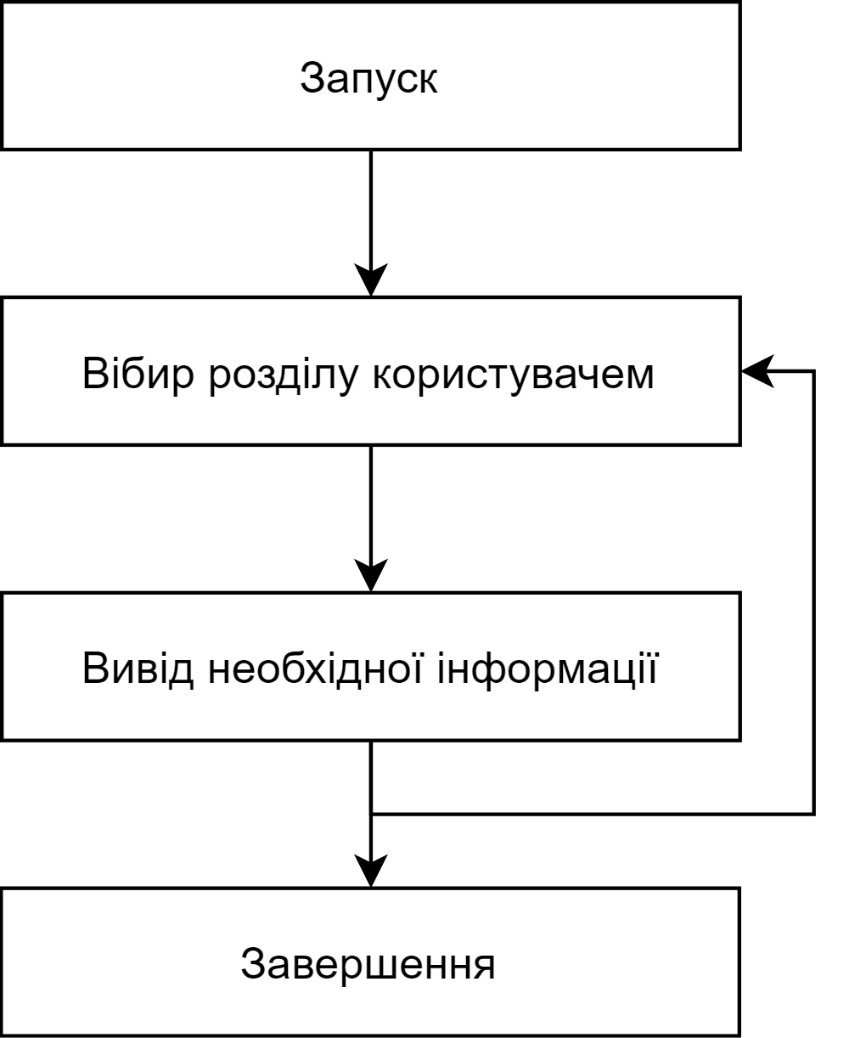


Рис.7 Діаграма станів

# Інші вимоги

## Вимоги до надійності

Програмний застосунок повинен відображати дійсну інформацію про стан системи, наявні компоненти та програмне забезпечення.

## Вимоги до продуктивності

Перемикання між розділами не повинне викликати затримку більш ніж декілька секунд у відображанні інформації розділу.

# Проект програмної системи

## Засоби реалізації

Розробка велась у середі розробки Visual Studio 2017. Графічний інтерфейс був збудований за допомогою бібліотеки VS Qt Tools та графічного редактору середи розробки Qt.

Мова програмування – С++

## Модулі і алгоритми

Проект складається з 7 модулів:

Модуль обробника графічного інтерфейсу – оброблює події які викликаються користувачем та видає відповідну інформацію.

Модуль Manager – використовуючи CLR звертається до WMI має змогу отримати інформацію про необхідний компонент системи та один із його параметрів.

Лістинг 3. Алгоритм отримання інформації про системні компоненти

vector<string> Manager::printInfo(String^ hardwareClass, String^ propertyName) {

vector<string> toReturn;

try {

ManagementObjectSearcher^ searcher = gcnew ManagementObjectSearcher("root\\CIMV2", "SELECT \* FROM " + hardwareClass);

ManagementObjectCollection^ collection = searcher->Get();

for each (ManagementObject^ object in collection)

{

if (object != nullptr && object[propertyName] != nullptr) {

string propStr = "";

auto prop = object[propertyName]->ToString();

if (prop != nullptr) {

propStr = msclr::interop::marshal\_as<string>(prop);

}

toReturn.push\_back(propStr);

}

}

}

catch (...) {

}

return toReturn;

}

Модуль System – за допомогою модулю Manager отримую інформацію про наступні компоненти системи: центральний процесор, оперативна пам’ять, відео система, логічні розділи, шині PCI, Plag-and-Play пристроях, BIOS та CMOS.

Лістинг 4. Використання Manager у модулі System

vector<pair<string, string>> SystemManager::getGpuPropeties() {

vector<pair<string, string>> gpuProperites;

for each (string str in printInfo("Win32\_VideoController", "Name"))

{

gpuProperites.push\_back(pair<string, string>("Name", str));

}

return gpuProperites;

}

Модуль Software – за допомогою Win API та відповідного ключа регіструю отримуємо інформацію про встановлене програмне забезпечення.

Модуль Drives – за допомогою модулю Manager отримує інформацію про фізичні носії системи.

Модуль Summary – за допомогою модулю Manager отримує інформацію про операційну систему

Модуль I/O Devices – за допомогою модулю Manager отримує інформацію про пристрої вводу-виведення системи.

Модуль Memory – за допомогою Win API спочатку отримуємо список поточних процесів та їх ID, потім завдяки ID отримуємо назву та виділену пам’ять кожного з процесів.

Лістинг 5. Алгоритм отримання інформації про процеси

vector<pair<string, string>> MemoryManager::getMemoryPropeties() {

vector<pair<string, string>> memoryProperites;

DWORD aProcesses[1024], cbNeeded, cProcesses;

unsigned int i;

if (!EnumProcesses(aProcesses, sizeof(aProcesses), &cbNeeded))

{

return memoryProperites;

}

// Calculate how many process identifiers were returned.

cProcesses = cbNeeded / sizeof(DWORD);

// Print the name and process identifier for each process.

for (i = 0; i < cProcesses; i++)

{

if (aProcesses[i] != 0)

{

PrintProcessNameAndID(&memoryProperites, aProcesses[i]);

}

}

std::sort(memoryProperites.begin(), memoryProperites.end(), [](auto &left, auto &right) {

return stoi(left.second) > stoi(right.second);

});

return memoryProperites;

}

## Проект інтерфейсу

Згідно до умов, у ході розробки програми було вирішено використовувати інтерфейс який складається з одного головного екрану у якому виконується навігація по розділам програми. Головний екран у свою чергу ділиться на навігаційне меню зліва та таблицю властивостей обраного розділу. Навігаційне меню складається с назви розділів, підрозділів та іконок що візуалізують зміст розділу та полегшують візуальне сприйняття застосунку.

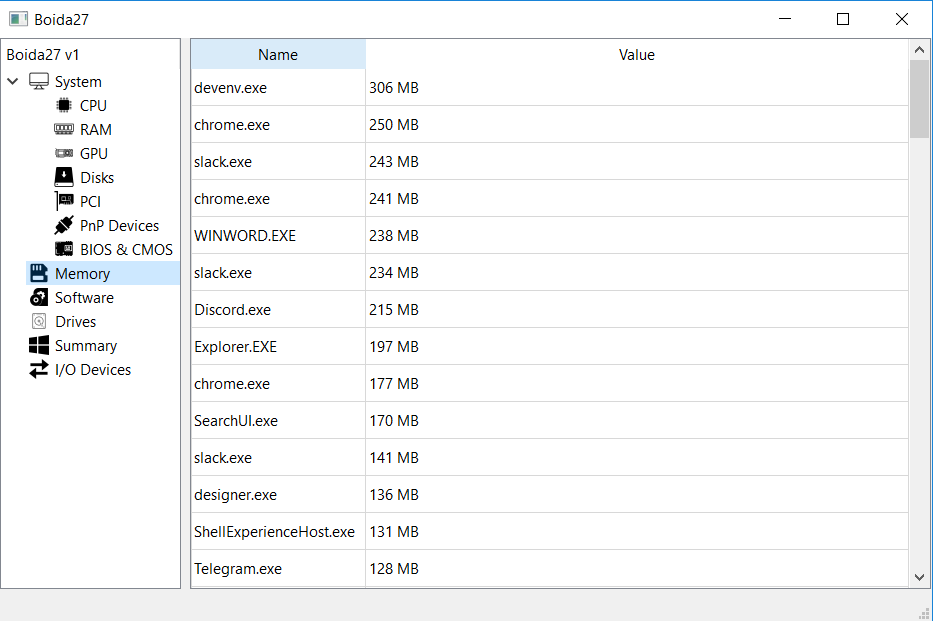


Рис.8 Інтерфейс застосунку

## Реалізація і тестування

Фізичні характеристики даної системи:

* Об’єм коду в строках – 831.
* Об’єм коду в КБ – 400.
* Кількість форм – 1.
* Витрати оперативної пам'яті МБ ~8.
* Затрати часу на обробку основних операцій ~ 0.2 сек.

Тестування проводилось по принципу «чорного ящика» після розробки кожної нової версії. Під час тесту перевірялись кожен з можливих збоїв в кожній функції.

# Висновки

Таким чином, у процесі виконання курсової роботи мною було розроблено програмний застосунок що має наступний функціонал: виведення системних та програмних компонентів, процесів та інформацію про операційну систему.

Була спроектована об’єктно-орієнтована архітектура продукту, побудована діаграма класів та діаграма послідовностей. Були отримані навички праці із Visual Studio IDE, Qt IDE, C++, WMI, Win32 API.

# Список літератури

1. Медвєдєв В.І. Особливості об’єктно-орієнтованого програмування на C++/CLI, C# и Java.—М.:Школа, 2010.
2. Гайсарян С.С. Объектно-ориентированное программирование. М.: ЦИТ, 2002.
3. Гордєєв А.В. Операційні системи.2-е видання. - СПб: Петербург, 2005;