ВСТУП

За останні двадцять років платформа Java набула великої популярності у користувачів. Це стало можливим завдяки великим можливостям, які закладені в її основу - мову програмування Java та віртуальну машину Java (JVM). Програмна платформа Java - ряд програмних продуктів та специфікацій компанії Sun Microsystems, раніше незалежної компанії, а нині підрозділу корпорації Oracle, які спільно надають систему для розробки прикладного програмного забезпечення та вбудовування її в будь-яке крос-платформенне програмне забезпечення[1]. Java використовується в самих різних комп'ютерних платформах: від вбудованих пристроїв і мобільних телефонів у нижньому ціновому сегменті до корпоративних серверів і суперкомп'ютерів у вищому ціновому сегменті[2]. Java-аплети використовуються для поліпшення функціональності і підвищення безпеки при перегляді всесвітньої павутини[3].

Програмний код, написаний на Java, віртуальна машина Java перетворює в байт-код Java[4].

Існує чотири варіанти платформ Java: Java Card, Java ME, Java SE, Java EE[5].

Java Card - технологія, яка дозволяє невеликим Java-програмам надійно працювати на смарт-картах та інших пристроях з малим об'ємом пам'яті[6].

Java ME - включає в себе кілька різних наборів бібліотек (відомих як профілі) для пристроїв з обмеженим об'ємом місця для зберігання, невеликим розміром дисплея і батареї[7]. Використовується для розробки програм для мобільних пристроїв, КПК, TV-ресиверів і принтерів[8].

Java SE - для використання на ПК, ноутбуках, та невеликих серверах[9].

Java EE - Java SE плюс API, корисне для багатопотокових клієнт-серверних бізнес-застосувань[A].

У курсовій роботі розроблюється програма для платформи Java SE[B].

1 Порядок виконання курсової роботи

1.1. Загальні положення

Курсова робота виконується студентами як підсумкове завдання та відіграє роль підсумкового контролю з дисципліниC. Згідно вимог, курсова робота, яка присвячена розробці програмного забезпечення, повинна складатися з таких розділів:

1. Загальний розділ
2. Розробка програмного забезпечення
3. Спеціальний розділ

Висновки

Перелік посилань

Додатки

1.2. Опис розділів

Розділ «Загальний розділ»

1.2.1. Опис та аналіз предметної області

Опис предметної області виконується з метою формулювання технічного завдання на розробку програмного забезпеченняD. Опис повинен містити такі етапи:

1. Визначення мети та призначення розробки, виявлення предметної області проектуE.
2. Збирання інформації по темі роботи, аналіз існуючих у даній галузі рішеньF. Головною метою даного етапу є виявлення основних характерних рис, технологій, архітектури та дизайну проекту10.
3. Вибір та обґрунтування критеріїв якості програмного забезпечення, формування вимог і обмежень11.
4. Визначення рішень, технологій (або навіть програмних бібліотек), які присутні у готових продуктах та можуть бути застосовані в даній розробці12.
5. Визначення напрямків досліджень для реалізації механізмів, що не є очевидними та не можуть бути повторно використаними чи запозиченими13.

Після аналізу предметної області визначаються можливості використання певних технологій проектування14. На цьому етапі треба також враховувати

архітектурні та функціональні вимоги до проекту, кількість учасників, перспективи подальшого розвитку, вимоги до дизайну15. Якщо проект, що розроблюється, буде частиною іншого проекту, треба передбачити механізми інтеграції з існуючими частинами проекту16. Для цього може знадобитися визначити технологію та інструментальні засоби розробки, а саме - мову програмування, бібліотеки та каркаси розробки17.

1.2.2. Постановка задачі на розробку програмного забезпечення

На основі результатів, отриманих внаслідок аналізу предметної області, виконується розробка технічного завдання (постановка задачі)18. У технічному завданні потрібно чітко та однозначно сформулювати галузь застосування, призначення та мету проекту; підстави для розробки, вимоги до архітектури, технологій та інструментів розробки проекту; вимоги до функціональних, технічних і експлуатаційних характеристик об'єкту розробки; визначення структури вхідних і вихідних даних; зв'язок проекту з іншими продуктами, характер використання результатів; вимоги до документації, основні етапи і терміни розробки; методи тестування19.

Розділ «Розробка програмного забезпечення»

Виконання проекту передбачає наступні етапи: *ескізний проект,* на якому приймаються загальні рішення щодо обрання інструментів розробки, побудови моделей і визначення засобів взаємодії продукту з оточенням (користувачем, обладнанням); *технічний проект,* на якому приймаються спеціальні рішення з деталізації алгоритмів, документування алгоритмів та інтерфейсів; *робочий проект,* на якому проводиться кодування програми1A.

На основі технічного завдання й аналізу вимог до технічних і експлуатаційних характеристик об'єкта розробки необхідно визначити технологію проектування1B.

Після вибору технології проектування треба визначити інструментальні засоби створення проекту1C. До них можна віднести мову програмування, транслятор, бібліотеки та середовище розробки, що відповідають сучасним вимогам до мови програмування, наприклад, якщо обрано мову Java, компілятор повинен підтримувати сучасну версію JDK - Java Development Kit (на 2016-2017 рік це - JDK8)1D.

За останні роки для будь-якої мови програмування створено велику кількість бібліотек1E. Навіть стандартна бібліотека мови Java складається з декількох тисяч класів, що дозволяють вирішувати найрізноманітніші завдання1F. Щодо середовища програмування, воно повинне бути інтуїтивно зрозумілим, зручним та швидким у роботі20. Фактично, стандартними середовищами для мови програмування Java є NetBeans IDE - вільне безкоштовне середовище, та IntelliJ IDEA - багатофункціональне комерційне середовище, проте безкоштовне для використання в освітніх закладах21.

Оскільки мова Java є об'єктно-орієнтованою, обов'язковим елементом проекту є побудова діаграми ієрархії об'єктів програми22. На цьому етапі виконується побудова ієрархії об'єктів, що відображає склад і структуру необхідних для реалізації проекту абстракцій23. Ієрархія об'єктів повинна бути описана з визначенням специфікацій для всіх об'єктів та проілюстрована графічно (рис24. 1)25. Окремим об'єктам на графічному зображені повинні відповідати окремі прямокутники, у яких указуються назви відпо-відних об'єктів і назви (прототипи) інтерфейсних функцій26. Зв'язки підпорядкування між об'єктами зображуються у вигляді ліній або стрілок, які вказують напрямок поширення керуючих впливів27. Інтерфейси, що реалізують додаткові засоби взаємодії, графічно повинні бути зображені відмінними від інтерфейсів підпорядкування засобами з обов'язковим вказанням напрямку поширення керуючих впливів28.

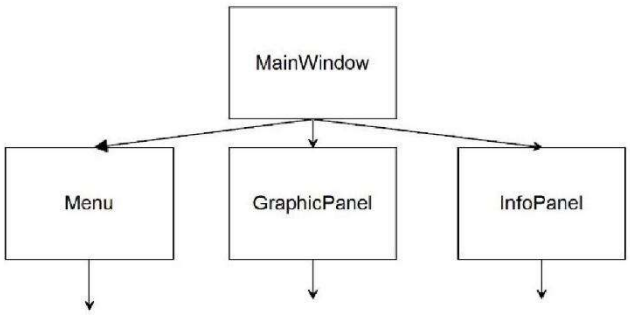


Рис29. 12A. Ієрархія об'єктів програми

На основі ієрархії об'єктів, описаної в попередньому розділі, розробляється ієрархія успадкування класів проекту2B. Опис ієрархії успадкування повинен включати специфікації класів та інтерфейсів2C. Результати проектування класів повинні бути подані графічно з виділенням абстрактних (графічно зображуються у вигляді прямокутників з переривчастими контурами) і кінцевих (графічно зображуються у вигляді прямокутників із суцільними контурами) класів2D. У

прямокутниках необхідно вказати назви відповідних класів2E. Приклад діаграми - на рис2F. 230.

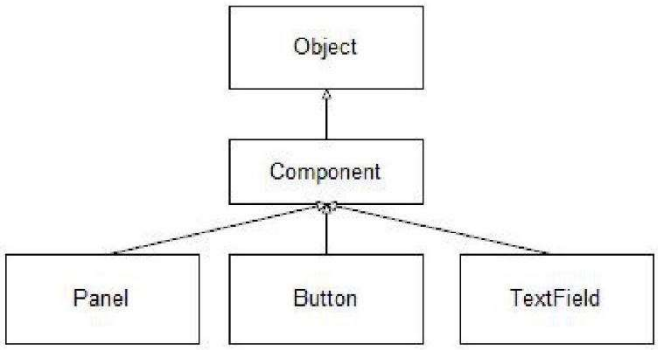


Рис31. 232. Ієрархія успадкування класів

Після опису ієрархії успадкування класів необхідно провести опис класів, їхніх полів та методів33. Цей розділ повинен містити деталізацію алгоритмів і технічну документацію на складові частини програми з докладним описом інтерфейсів, властивостей і поведінки класів, принципів організації даних та алгоритмів функцій з обґрунтуванням прийнятих рішень34. Важливі для проекту алгоритми потрібно ілюструвати за допомогою блок-схем або діаграм взаємодії (UML)35. Якщо в розробці було використано готові програмні рішення інших розробників (бібліотеки, фреймворки та ін36.), необхідно навести короткі описи цих рішень з обґрунтуванням доцільності їхнього вибору37.

1.2.3. Розробка програмного забезпечення

На основі рішень, прийнятих у попередніх розділах можна приступати до реалізації проектних рішень та тестування програми, що буде створена38. Цей розділ роботи повинен містити опис основних етапів реалізації програми, а також використаних на кожному з цих етапів методик тестування39. Необхідно також враховувати, що розробка програми проводиться на комп'ютері розробника в умовах, що можуть достатньо сильно відрізнятись від умов експлуатації3A. Тому, для остаточного (приймального) етапу тестування, необхідно вказати умови, в яких буде працювати закінчений програмний продукт3B. Якщо при тестуванні окремих елементів програми використовувались додаткові програми або тестові фреймворки, то необхідно навести код цих програм або дати посилання на опис тестового фреймворку3C.

Основне призначення аналізу отриманих результатів - визначення, в якій мірі

результати виконаної роботи відповідають початковим вимогам до програмного продукту і поточного стану справ у предметній області3D. Також цей аналіз може бути використаний для формування на його основі планів і перспективних напрямків розвитку проекту або його частин3E. Виконання аналізу може бути необхідним з таких причин: у роботі над проектом функціональність або властивості окремих його частин були виконані частково або з обмеженнями; в результаті роботи над проектом було з'ясовано, що його можна розширити та доповнити новими функціями, або поліпшити якість виконання існуючих функцій3F. Також не виключається ситуація, що під час роботи над проектом у предметній області відбулись важливі для проекту зміни40.

1.2.3. Розділ 3 «Спеціальний розділ»

Можливі такі три підрозділи (за вибором).

В підрозділі 3.1 «Інструкція з інсталяції програмного забезпечення» визначається мінімальна конфігурація комплексу технічних засобів, в разі необхідності - потрібні додаткові пристрої, технічні характеристики принтера; вказуються можливі конфігурації механічних засобів, які потрібні для роботи програмного комплексу в різних умовах застосування. В разі необхідності вказується ОС, в якій даний програмний комплекс може працювати, середовища програмування, що були використані під час розробки, описуються файли настроювання, створені для забезпечення функціонування комплексу (інсталяційні програми, архіватори, розпаковщики), мова програмування, якою написані програми комплексу, технологічний тип представлення програм (початковий, об’єктний, завантажувальний, архівний).

В підрозділі 3.2 «Інструкція з використання тестових наборів» необхідно передбачити використання набору тестів.

Типи тестів:

- аварійні тести;
- вироджені тести;
- комплексні тести;
- основні тести;
- стикові тести;
- структурне тестування;

- функціональне тестування.

В підрозділі 3.3 «Інструкція з експлуатації програмного комплексу» визначається спосіб звертання до комплексу, вказується рекомендації щодо шляхів розміщення командних файлів комплексу та інформаційних файлів системи.

Допускається наведення відомостей про використання оперативної пам'яті та обсягів завантажувальних модулів в кілобайтах.

1.2.4. Висновки

Заголовок цього розділу – слово "Висновки" надруковане окремим рядком великими буквами. Йому порядковий номер не присвоюється.

Цей розділ повинен включати у себе оцінку результатів курсової роботи, у тому числі їх відповідність вимогам завдання.

У розділі наводиться коротка викладка показників, отриманих при виконанні роботи; вказуються напрями подальшої роботи над темою або мотивується недоцільність продовження роботи; вказуються статті, авторські свідоцтва (заявки), доповіді і повідомлення, опубліковані, підготовлені до друку і прочитані у процесі роботи.

У висновках студент вказує, яку частину роботи він виконав самостійно і які нові завдання розв'язані ним у процесі виконання.

1.2.5. Перелік посилань

У перелік посилань включають усі джерела, використані студентом під час виконання проекту.

Перелік посилань складають в алфавітному порядку або за порядком використання літератури у пояснювальній записці.

У тексті записки повинна вказуватися вся література, що включена до списку.

1.2.6. Додатки

В кожному конкретному випадку перелік графічної частини визначається керівником дипломного проекту та студентом відповідно до теми проекту41. Графічна частина курсової роботи містити наступні креслення:

- структурні або функціональні схеми ПЗ, що розробляється чи вдосконалюється, або окремих його частин;
- блок-схеми алгоритмів роботи ПЗ, його функцій, методів чи окремих скриптів;

- ER-діаграму бази даних ПЗ;
- UML-діаграму діяльності користувачів ПЗ;
- UML-діаграму класів ПЗ[42].

Тексти програм повинні починатися з найменування програмних модулів (класів, файлів)[43]. Кожен з них повинен починатися з нової сторінки[44]. Кожен клас повинен мати початковий коментар, що описує призначення класу, його взаємодію з іншими класами та інші додаткові відомості, що необхідні для розуміння роботи класу[45]. При написанні програм бажано враховувати необхідність друку кодів програмних модулів та, за можливістю, не припускати розривання рядків коду засобами друку текстів[46].

Якщо програма має графічний інтерфейс користувача, це повинно бути відображено у додатках із використанням знімків екранних форм - "скріншотів"[47].

2 Графічна бібліотека Swing

2.1. Обґрунтування використання графічної бібліотеки

Оскільки завданням курсової роботи є розробка програми, що виконує побудову одного з видів діаграм (кругова, стовпчикова, гістограма), то зробити це буде неможливо, якщо не використати одну з існуючих графічних бібліотек48.

2.2. Огляд існуючих графічних бібліотек для Java

Сучасні програми потребують графічного інтерфейсу користувача (GUI)49. Користувачі відвиклі працювати через консоль: вони керують програмою і вводять вхідні дані за допомогою так званих елементів управління (в програмуванні їх також називають візуальними компонентами), до яких відносяться кнопки, текстові поля, списки, що випадають, і т4A. д4B.

Кожна з сучасних мов програмування надає безліч бібліотек для роботи зі стандартним набором елементів управління4C. Нагадаємо, що під бібліотекою в програмуванні мають на увазі набір готових класів та інтерфейсів, призначених для вирішення певного кола завдань4D.

У мові Java є три бібліотеки візуальних компонентів для створення графічного інтерфейсу користувача4E. Найбільш рання з них називається AWT4F. Вважається, що при її проектуванні був допущений ряд недоліків, внаслідок яких з нею досить складно працювати50. Бібліотека Swing розроблена на базі AWT і замінює більшість її компонентів своїми, спроектованими більш ретельно і зручно51. Третя, найновіша бібліотека базується на можливостях створення інтерфейсів користувача у новій мові програмування JavaFX52.

Кожна бібліотека надає набір класів для роботи з кнопками, списками, вікнами, меню і т53. д54., але ці класи спроектовані по-різному: вони мають різний набір методів з різними параметрами, тому "перевести" програму з однієї бібліотеки на іншу (наприклад , з метою збільшення швидкодії) не так-то просто55. Це майже як перейти з однієї мови програмування на іншу: всі мови вміють робити одне і те ж, але у кожної з них свій синтаксис, своя програмна структура і свої численні хитрощі56.

З цієї причини замість того, щоб робити огляд усіх трьох бібліотек, основну увагу буде приділено одній з них - бібліотеці Swing57. Повноцінний графічний інтерфейс може бути розроблений з її допомогою58.

2.3. MVC архітектура контейнерів і компонентів Swing

Перед тим, як розглянути основні класи бібліотеки Swing, варто зауважити, що в основу бібліотеки покладено концепцію організації компонентів, що отримала назву "Model-View-Controller" (скорочена назва MVC)[59].

2.3.1. *Історія виникнення*

Концепцію MVC було описано Т[5A].Ринскаугом в 1979, який працював тоді з мовою програмування Smalltalk в Xerox PARC[5B]. Оригінальна реалізація описана в статті "Розробка застосувань в Smalltalk-80: Як використовувати Model-View-Controller"[5C]. Тоді було створено версію MVC для бібліотеки класів Smalltalk-80[5D].

В оригінальній концепції була описана сама ідея і роль кожного з елементів: моделі, представлення та контролера[5E]. Але зв'язки між ними були описані без конкретизації[5F]. Крім того, розрізняли дві основні модифікації:

- пасивна модель - модель не має жодних способів впливати на представлення або контролер, і використовується ними як джерело даних для відображення[60]. Всі зміни моделі відслідковуються контролером і він же відповідає за відображення представлення, якщо це необхідно[61]. Така модель частіше використовується в структурному програмуванні, оскільки в цьому випадку модель - це просто структура даних, без методів їх обробки;
- активна модель - модель оповіщає представлення про те, що в ній відбулися зміни, а представлення, що зацікавлені в оповіщенні, підписуються на ці повідомлення[62]. Це дозволяє зберегти незалежність моделі як від контролера, так і від представлення[63].

Класичною реалізацією концепції MVC прийнято вважати версію саме з активною моделлю[64].

З розвитком об'єктно-орієнтованого програмування і поняття про шаблони проектування був створений ряд модифікацій концепції MVC, які при реалізації у різних авторів можуть відрізнятися від оригінальної[65].

2.3.2. *Призначення*

Основна мета застосування цієї концепції полягає в розділенні бізнес- логіки (моделі) від її візуалізації (представлення)[66]. За рахунок такого поділу підвищується можливість повторного використання класів програми[67]. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах або з різних точок зору Зокрема, виконуються

наступні завдання:

1. До однієї моделі можна приєднати кілька представлень, при цьому вони ніяким чином не впливають на реалізацію моделі68. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми та кругової діаграми69.

2. Не торкаючись реалізації представлень, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних з клавіатури, використання джойстику), для цього досить використати (а можливо і створити) інший контролер6A.

3. Існує багато розробників, які спеціалізуються тільки в одній з областей: або розробляють графічний інтерфейс (інтерфейсні програмісти), або розробляють бізнес-логіку (логічні програмісти)6B. Тому можливо досягнути того, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть інформовані про те, яке представлення буде використовуватися6C.

2.3.3. Концепція

Концепція MVC дозволяє розділити дані, їхнє представлення та обробку дій користувача на три окремих компоненти:

- модель (англ6D. *Model)*6E. Модель надає інформацію: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан6F. Не містить відомостей, як цю інформацію можна візуалізувати;
- представлення, іноді його називають "вид" (англ70. Hew)71. Відповідає за відображення інформації (візуалізацію)72. Часто як представлення виступає вікно або панель з графічними елементами;
- контролер (англ73. *Controller)*74. Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель і представлення для реалізації необхідної реакції75.

Завжди треба мати на увазі, що як представлення, так і контролер залежать від моделі76. Однак модель не залежить ні від представлення, ні від контролера77. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуального представлення, а також створювати кілька різних представлень для однієї моделі78.

Для реалізації схеми Model-View-Controller використовуються традиційні

шаблони проектування, основні з яких "Observer", "Strategy", "Composite"[79].

Найбільш типова реалізація відокремлює представлення від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи механізм "підписка - сповіщення"[7A]. При кожній зміні внутрішніх даних в моделі, вона оповіщає всі залежні від неї представлення, і представлення оновлюються[7B]. Для цього використовується шаблон "Observer"[7C]. При обробці реакції користувача представлення вибирає, в залежності від потрібної реакції, потрібний контролер, який забезпечить той чи інший зв'язок з моделлю[7D]. Для цього використовується шаблон "Strategy", або замість цього може бути модифікація з використанням шаблону "Command"[7E]. Якщо представлення має складну ієрархічну структуру, то для можливості однотипної поведінки підоб'єктами такого представлення може використовуватися шаблон "Composite"[7F]. Крім того, можуть використовуватися й інші шаблони проектування, наприклад, "Factory method", який дозволить задати за замовчуванням тип контролера для відповідного представлення[80].

2.3.4. Вікно JFrame

Кожна GUI-програма запускається у вікні і по ходу роботи може відкривати кілька додаткових вікон[81].

В бібліотеці Swing описаний клас JFrame, що представляє собою вікно з рамкою і рядком заголовка (з кнопками "Згорнути", "На весь екран" і "Закрити")[82]. Воно може змінювати розміри і переміщатися по екрану[83].

Конструктор `JFrame ()` без параметрів створює порожнє вікно[84]. Конструктор `JFrame(String title)` створює порожнє вікно з заголовком title[85].

Щоб написати найпростішу програму, що виводить на екран порожнє вікно, нам буде потрібно ще три методи:

`setSize (int width, int height)` - встановлює розміри вікна[86]. Якщо не задати розміри, вікно буде мати нульову висоту незалежно від того, що в ньому знаходиться, і користувачеві після запуску доведеться розтягувати вікно вручну Розміри вікна включають не тільки "робочу" область, а й кордони і рядок заголовка[87].

`setDefaultCloseOperation (int operation)` - дозволяє вказати дію, яку необхідно виконати, коли користувач закриває вікно натисканням на хрестик[88]. Зазвичай в програмі є одне або кілька вікон при закритті яких програма припиняє роботу[89]. Для того, щоб запрограмувати це поведінка, слід як параметр operation

передати константу EXIT_ON_CLOSE, описану в класі JFrame8A.

setVisible (boolean visible) - коли вікно створюється, воно за замовчуванням невидиме8B. Щоб відобразити вікно на екрані, викликається даний метод з параметром true8C. Якщо викликати його з параметром false, вікно знову стане невидимим8D.

Тепер розглянемо програму, яка створює вікно, виводить його на екран і завершує роботу після того, як користувач закриває вікно8E.

```
import javax.swing.*;
public class MyClass {
public static void main (String [] args) { JFrame myWindow = new JFrarne("npo6He вікно”) ; myWindow.setDefaultCloseCperation(JFrarre.EXIT CN CLOSE); myWindow.setSize(400, 300); myWindow.setVisible(true);
}
}
```

Зверніть увагу, для роботи з більшістю класів бібліотеки Swing знадобиться імпортувати класи пакету java.swing.

Як правило, перед відображенням вікна, необхідно здійснити набагато більше дій, ніж в цій простій програмці8F. Необхідно створити безліч елементів управління, налаштувати їх зовнішній вигляд, розмістити в потрібних місцях вікна90. Крім того, в програмі може бути багато вікон і налаштовувати їх все в методі main () незручно і неправильно, оскільки порушує принцип інкапсуляції: тримати разом дані і команди, які їх обробляють91. Логічніше було б, щоб кожне вікно займалося своїми розмірами і вмістом самостійно92. Тому класична структура програми з вікнами виглядає наступним чином:

Файл SimpleWindow.java:

```
public class SimpleWindow extends JFrame { SimpleWindow(){
super("Не вікно");
setDefaultCloseOperation(EXIT_ON_CLOSE); setSize(250, 100);
}
}
```

Файл Program.java:

```
public class Program {
public static void main (String [] args) { JFrame myWindow = new SimpleWindow(); myWindow.setVisible(true);
}
}
```

З прикладу видно, що вікно описується в окремому класі, що є спадкоємцем JFrame і настроює свій зовнішній вигляд і поведінку в конструкторі (першою командою викликається конструктор суперкласу)93. Метод main() міститься в іншому класі, відповідальному за управління ходом програми94. Кожен з цих класів дуже простий, кожен займається своєю справою, тому в них легко розбиратися і легко супроводжувати (тобто удосконалювати при необхідності)95.

Примітка96. *Зверніть увагу, що метод setVisible() не викликається в класі SimpleWindow, що цілком логічно: за тим, де яка кнопка розташована та які розміри воно повинно мати, стежить саме вікно, а от приймати рішення про те, яке вікно в який момент виводиться на екран - обов'язок керуючого класу програми97.*

2.3.5. ***Панель вмісту вікна***

Напряму в вікні елементи управління не розміщуються98. Для цього служить панель вмісту, що займає весь простір вікна99. Звернутися до цієї панелі можна методом getContentPane() класу JFrame9A. За допомогою методу add(Component component) можна додати на неї будь-який елемент управління9B.

Спочатку розглянемо тільки один елемент управління - кнопку (не вдаючись у подробиці її побудови)9C. Кнопка описується класом JButton і створюється конструктором з параметром типу String - написом9D.

Додамо кнопку в панель вмісту вікна командами:

```
JButton newButton = new JButton();
getContentPane() .add (newButton);
```

В результаті отримаємо вікно з кнопкою9E. Кнопка займає всю припустиму площу вікна9F. Такий ефект взагалі не є прийнятним, тому необхідно використовувати різні способи розміщення елементів на панелі A0.

У деяких старомодних мовах програмування необхідно було вказувати координати і розміри кожного компонента вікнаA1. Це працювало добре, якщо було відомо роздільну здатність екрану кожного користувачаA2. У Java використовується абстракція менеджерів компоновки (розміщення) (Layout Managers), які дозволяють розміщувати компоненти на екрані, не знаючи точних позицій компонентівA3. Менеджери компоновки гарантують, що та частина інтерфейсу, за яку вони відповідають, буде виглядати правильно незалежно від розмірів вікна і роздільної здатності екрануA4.

Swing надає такі менеджери компоновки:

- FlowLayout
- GridLayout
- BoxLayout
- BorderLayout
- CardLayout
- GridBagLayout
- GroupLayout

Щоб використовувати будь-який з цих менеджерів, необхідно створити його екземпляр і потім призначити цей об'єкт якомусь контейнеру (container), наприклад панеліA5.

Використання менеджерів компоновки розлянемо на прикладі створення програми, що розв'язує квадратне рівняння $ax^2+bx+c=0$

FlowLayout - послідовне розташування

За цією схемою компоненти розміщуються у вікні (або іншому контейнері) рядок за рядкомA6. Наприклад, текстові мітки, іконки, текстові поля і кнопки будуть додаватися в перший умовний рядок, поки в ньому є місцеA7. Коли перший рядок

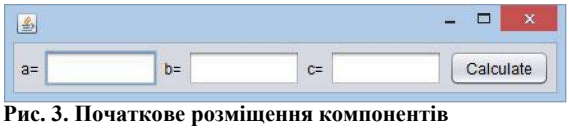


Рис. 3. Початкове розміщення компонентів

заповниться, компоненти, що залишилися будуть додаватися до наступного рядка і так даліA8. Якщо користувач змінить розмір вікна, картина може змінитисяA9.

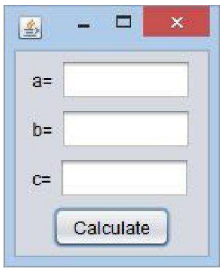


Рис. 4. Автоматичне переміщення компонентів

Якщо потягнути за кут вікна, його розмір зміниться, при цьому FlowLayout переміститься елементи вікна відповідно до зміни розмірівАА.

FlowLayout - менеджер компоновки "за замовчуванням", тому, якщо головна панель вікна використовує його, це навіть не обов'язково вказувати:

```
    private void initComponents() {
    jPanell = new javax.swing.JPanel() jLabell = new
javax.swing.JLabel() jTextFieldl = new javax.swing.JTextField
() jLabel2 = new javax.swing.JLabel(); jTextField2 = new
javax.swing.JTextField () jLabel3 = new javax.swing.JLabel();
    jTextField3 = new javax.swing.JTextField(); jButtonl = new
javax.swing.JButton();
    setDefaultCloseOperation(
    javax.swing.WindowConstants.EXIT ON CLOSE);
    // перед тим, як додавати елементи вікна, // треба
встановити
    // менеджер компоновки,
    // але для FlowLayout - це необов’язково
    j Labell.setText(”a=”) ; jPanell.add(jLabell) ;
    jTextFieldl.setColumns(7); jPanell.add(jTextFieldl) ;
    j Label2.setText(”b=”); jPanel1.add(j Label2) ;
    jTextField2.setColumns(7); jPanel1.add(jTextField2) ;
    j Label3.setText(”c=”); jPanel1.add(jLabel3) ;
    jTextField3.setColumns(7); jPanel1.add(jTextField3) ;
    jButtonl.setText("Calculate”) ; jPanell.add(jButtonl) ;
    getContentPane().add(
    j Panell, j ava.awt.BorderLayout.CENTER) ; pack();
    }
```

GridLayout - табличне розташування

Клас java.awt.GridLayout дозволяє організувати компоненти, як рядки і стовпці в таблиціАВ. Компоненти будуть додаватися в комірки умовної таблиці. Якщо розмір

вікна буде збільшений, комірки стануть більше, але положення компонентів відносно один одного залишиться тим самимAC. У нашій програмі сім компонентів - три текстові мітки, три текстових поля і кнопка. Ми можемо розмістити їх в таблиці з чотирма рядками і двома колонками (одна комірка залишиться порожньою) AD:

```
GridLayout layout = new GridLayout(4,2);
```

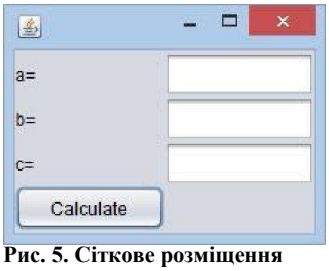


Рис. 5. Сіткове розміщення

При збільшенні вікна відносне положення зберігається, а розміри елементів змінюються:

Також можна задати відстань між комірками по вертикалі і горизонталі, наприклад в десять пікселів:

```
GridLayout layout = new GridLayout(4, 2, 10, 10);
```

Ще одна важлива річ, яку варто пам'ятати про табличний менеджер компоновки

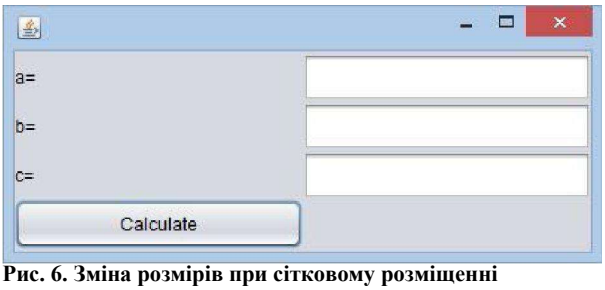


Рис. 6. Зміна розмірів при сітковому розміщенні

- на ньому всі комірки мають однакову довжину і ширину.

BorderLayout - розміщення по областях

Клас `java.awt.BorderLayout` поділяє вікно на Південну, Західну, Північну, Східну і Центральну області. Північна область знаходиться нагорі вікна, Південна - знизу, Західна - ліворуч, а Східна - праворучAE.

При додаванні елементу у вікно або панель з таким розміщенням, треба явно вказувати, в яку з п'яти зазначених областей буде розміщуватися компонент. Наприклад, `panel1.add("North", txtDisplay);`

BoxLayout - розташування по горизонталі або вертикалі

Клас java.swing.BoxLayout розміщує кілька компонентів вікна горизонтально (по осі абсцис) або вертикально (по осі ординат)AF. На відміну від менеджера FlowLayout, коли вікно з BoxLayout змінює свій розмір, його елементи керування не зміщуються зі своїх позиційB0. BoxLayout дозволяє елементам вікна мати різні розміри (чого не дозволяє GridLayout)B1. Наступні два рядки коду задають BoxLayout з вертикальним вирівнюванням на JPanelB2.

```
JPanel p1= new JPanel();
setLayout(new BoxLayout(p1,BoxLayout.Y AXIS));
```

Щоб зробити цей код коротшим, у ньому не створювалась окрема змінна для зберігання посилання на об'єкт BoxLayout, замість цього було створено екземпляр цього класу і відразу ж передано його, як аргумент методу setLayout()B3.

GridBagLayout - більш гнучке табличне розташування

GridBagLayout - більш розширена схема розміщенняB4. Вона дозволяє задавати розмір комірки, рівним декільком клітинам таблиціB5. GridBagLayout має допоміжний клас, який називається GridBagConstraints (обмеження на клітини таблиці)B6. Ці обмеження не що інше, як атрибути комірок, які необхідно задавати для кожної комірки таблиці окремоB7. Всі обмеження мають бути задані до того, як у комірку поміщаються компонентиB8. Наприклад, один з атрибутів GridBagConstraints називається gridWidthB9. Він дозволяє задати ширину якоїсь однієї клітинки, рівній ширині кількох іншихBA. Під час роботи з GridBagLayout необхідно спочатку створити екземпляр класу GridBagConstraints, і потім задати значення для його властивостейBB. Після того як це зроблено, можна додавати об'єкт в клітинку контейнераBC.

Комбінування схем розміщення

Якщо розглянути інші менеджери компоновки, можна зрозуміти, що жоден з них (крім GroupLayout) не дозволяє створити вікна, що нагадуватимуть своїм виглядом, наприклад стандартний калькулятор в Microsoft WindowsBD. Як випливає із документації, GroupLayout створювався для використання у спеціалізованих середовищах розробкиBE. Наприклад, середовище NetBeans дозволяє будувати інтерфейси користувача із використанням цього менеджера компоновки за допомогою спеціалізованого дизайнера користувальницького інтерфейсуBF. Якщо подивитись на код програми, що згенерований за допомогою дизайнера інтерфейсу в NetBeans, можна побачити, що він досить складний і заплутанийC0. Його майже неможливо редагувати вручнуC1. Тому, якщо інтерфейс користувача програми повинен бути простим, використовуйте один із традиційних менеджерів компоновки, якщо ж потрібно представити складний інтерфейс - використання дизайнера інтерфейсу та GroupLayout має сенсC2.

Таким чином, після розгляду контейнерів та схем компоновки, розглянемо

докладніше елементи графічного інтерфейсу - компоненти.

2.4. Опис компонентів бібліотеки Swing

Бібліотека Swing надає величезний набір компонентів для розробки застосувань з графічним інтерфейсом. Більшість компонентів Swing - легкі компоненти, які проектовані таким чином, що використовують Java- код та без необхідності не використовують відповідні класи базової операційної системи.

Клас JComponent є базовим класом для всіх компонентів Swing. Компонент, який може містити інші компоненти, називається контейнером. Бібліотека Swing надає два типи контейнерів: контейнери верхнього рівня, та всі інші. Контейнер верхнього рівня не може міститись в інший контейнер, і він може бути відображений безпосередньо на робочому столі. Об'єкт класу JFrame представляє контейнер верхнього рівня.

Об'єкт класу JButton представляє собою кнопку. Кнопка також відома як "командна кнопка".

Користувач натискає або клацає мишкою по компоненту JButton для того, щоб ініціювати виконання визначеної дії. Кнопка може відображати текст, значок або обидва.

Об'єкт класу JPanel являє собою контейнер, який може містити інші компоненти. Як правило, об'єкт класу JPanel використовується для групування компонентів, пов'язаних між собою. JPanel НЕ є контейнером верхнього рівня.

Об'єкт класу JLabel являє собою компонент мітки, що відображає текст, значок або обидва. Як правило, текст в JLabel описує інший компонент.

Бібліотека Swing надає кілька текстових компонентів, які дозволяють відображати і редагувати різні типи тексту.

Об'єкт класу JTextField використовується для роботи з одним рядком звичайного тексту.

Об'єкт класу JTextArea використовуватися для роботи з багаторядковим звичайним текстом.

Об'єкт класу JPasswordField використовується для роботи з одним рядком тексту, в якому фактичні символи в тексті замінені луна-символами.

Об'єкт класу JFormattedTextField працює з одним рядком звичайного тексту де Ви можете вказати форматування тексту, наприклад, таке як відображення дати в

"ДД/ММ/УУУУ"DA.

Об'єкт класу JEditorPane дозволяє працювати зі стилізованим текстом, наприклад, в HTML і RTF форматахDB.

Об'єкт класу JTextPane працює з стилізованим документом з впровадженими зображеннями і компонентамиDC.

Також, для кожного текстового компонента можна додати вхідний верифікатор для перевірки тексту, введеного користувачемDD. Екземпляр класу InputVerifier діє як вхідний верифікаторDE. Ви можете встановити вхідний верифікатор для текстового компонента за допомогою методу setInputVerifier () об'єкта, успадкованого від JComponentDF.

Бібліотека Swing надає безліч компонентів, які дозволяють вибрати один або кілька елементів зі списку елементівE0. Такі компоненти є об'єктами класів JToggleButton, JCheckBox, JRadioButton, JComboBox, і JListclassesE1. JToggleButton може бути в натиснутому, або ненатиснуто- му станіE2. JCheckBoxcan використовується для представлення вибору типу "так/ні"E3. Іноді група компонентів CheckBox використовується, щоб дозволити користувачеві вибрати нуль або більше варіантівE4.

Група компонентів JRadioButton використовується для представлення користувачам набору взаємовиключних варіантівE5.

Об'єкти класу JComboBox використовуються, щоб надати користувачеві вибір одного з набору варіантів, де користувач, можливо, може ввести нове значенняE6. JComboBox займає менше місця на екрані в порівнянні з іншими варіантами, які можна створити використовуючи інші компоненти, тому що в ньому приховані всі варіанти, і користувач повинен відкрити список варіантів, перш ніж він може зробити вибірE7.

Через об'єкт класу JList користувач може вибрати нуль або декілька варіантів зі списку виборуE8. Всі варіанти в JList видимі користувачеві E9.

Компонент JSpinner поєднує переваги JFormattedTextField та JComboBoxEA. Це дозволяє задавати список варіантів, які встановлюються в JComboBox, і в той же час, можна застосувати формат до відображуваної величиниEB. Таким чином він показує тільки одне значення зі списку варіантівEC. Цей компонент також дозволяє ввести нове значенняED.

Об'єкт класу JScrollBar використовується, щоб забезпечити можливість прокрутки для перегляду компоненту, який більше за розміром, ніж доступний вільний простірEE. JScrollBar може бути встановлений вертикально або горизонтальноEF. Скролінг здійснюється шляхом перетягування "ручки" вздовж доріжки JScroUBarF0. Для його правильної роботи потрібно написати логіку для забезпечення можливості прокрутки за допомогою компонента JScroUBarF1.

JScrollPane - контейнер, який використовується, щоб обернути компонент, більший за розміром, ніж простір доступного вільного місцяF2.

Об'єкт класу JScrollPane забезпечує можливість автоматичної прокрутки в горизонтальному і вертикальному напрямкахF3.

Об'єкт JProgressBar використовується для відображення ходу виконання завданняF4. Він може мати горизонтальну або вертикальну орієнтаціюF5. він має три значення, пов'язані з ним: поточне значення, мінімальне значення, максимальне значенняF6. Якщо прогрес завдання не відомо, кажуть, що компонент JProgressBar знаходиться в невизначеному станіF7.

Об'єкт класу JSlider призначений для графічного вибору певного значення з діапазону значень між двома цілими числами, зсунувши ручку вздовж доріжкиF8.

Компонент JSeparator - зручний компонент для того, щоб додати роздільник між двома компонентами або двома групами компонентівF9. Як правило, JSeparator використовується в меню, щоб відокремити групи пов'язаних елементів менюFA. Як правило, це виглядає як горизонтальна або вертикальна суцільна лініяFB.

Компонент "меню" використовується для забезпечення список дій для користувача в компактній форміFC.

Об'єкт класу JMenuBar представляє собою панель менюFD. Об'єкти класів JMenu, JMenuItem, JCheckBoxMenuItem, та JRadioButtonMenuItem представляють собою різноманітні варіанти пунктів менюFE.

JToolBar - панель кнопок представляє собою групу невеликих кнопок, які забезпечують найбільш часто використовувані дії для користувача в JFrameFF. Як правило, панель кнопок використовується разом з меню100.

Об'єкт класу JTable використовується для відображення і редагування даних в табличній формі101. Він представляє дані у вигляді рядків і стовпців102. Кожна колонка має заголовок стовпця103. Рядки і стовпці використовують індекси,

починаючи з 0104.

Об'єкти класу JTree використовуються для відображення ієрархічних даних у вигляді дерева105. Кожен елемент в JTree називається вузлом106. Вузол що має дочірні вузли, називається вузлом розгалуження107. Вузол, який не має дочірніх вузлів, називається листовим вузлом108. Також вузол називається батьківським вузлом для його дочірніх вузлів109. Перший вузол в JTree, що немає батьківського вузла, називається кореневим вузлом10A.

Об'єкт класу JTabbedPane діє як контейнер для інших компонентів Swing, розташовуючи їх у вигляді вкладок10B. Він може відображати вкладки, використовуючи назву, іконку, або обох10C. При використанні такого компонента, видно лише зміст тільки однієї вкладки10D. Використовуючи JTabbedPane можна поділити простір між декількома вкладками10E.

Об'єкти класу JSplitPane - розгалужувачі, які можуть бути використані, щоб розділити простір між двома компонентами10F. Стрічка розділювача може відображатися горизонтально або вертикально110. Коли вільного місця менше, ніж простір, необхідне для відображення двох компонентів, користувач може переміщати планку розгалужувача вгору/вниз або вліво/вправо, таким чином один компонент отримує більше місця, ніж інший111. Якщо є достатньо місця, обидва компоненти можуть бути показані повністю112.

Об'єкти класу JDialog у бібліотеці Swing - контейнери верхнього рівня113. Вони використовуються як тимчасовий контейнер верхнього рівня (або як спливаюче вікно) для надання допомоги в роботі з основним вікном, щоб привернути увагу користувача, або для реалізації входу користувача114. Об'єкт класу JOptionPane забезпечує багато статичних методів, щоб показати різні типи діалогів для користувача за допомогою екземпляра класу JDialog115.

Об'єкт класу JFileChooser допомагає користувачеві вибрати файл / каталог з файлової системи за допомогою вбудованого діалогу116.

Об'єкт JColorChooser дозволяє користувачеві вибрати колір графічно за допомогою вбудованого діалогу117.

Бібліотека Swing дозволяє встановити кольори фону і переднього плану компонента118. Для цього використовується об'єкт класу java119.awt11A.Color11B. Він представляє собою певний колір11C. Ви можете вказати колір за допомогою

червоного, зеленого, синього та альфа-компонентів або використовуючи компоненти відтінку, насиченості і яскравості11D. Об'єкт класу Color - незмінний11E. Він забезпечує декілька констант, які представляють часто використовувані кольори, наприклад, Color11F.RED, Color120. BLUE представляють червоний та синій кольори121.

В Swing, ви можете намалювати рамку навколо компонентів122. Рамка представлена екземпляром класу що реалізує інтерфейс Border123. Є різні типи рамок124. Клас BorderFactory забезпечує фабричні методи для створення всіх типів рамок125.

Swing дозволяє встановити шрифт для тексту, відображуваного в компонентах126. Для цього використовується клас java127.awt128.Font - він представляє шрифт у програмі Java129.

Бібліотека Swing дозволяє малювати різні типи форм (круги, прямокутники, лінії, полігони і т12A.д12B.) з використанням об'єкта, що належить до класу Graphics12C.

Як правило, ви використовуєте JPanel у якості полотна для малювання фігур12D. Swing надає два способи, щоб перемалювати компоненти у вікні або на панелі: асинхронно і синхронно12E. Виклик метода repaint() малює компонент асинхронно, проте виклик методу paintImmediately() призведе до негайного відображення компонент12F.

Перемалювання деталей може бути виконане на екрані або поза екраном130. Якщо виконувати малювання безпосередньо на екрані, це може призвести до мерехтіння131. Проте об'єкт Image може бути створений поза кадром, із використанням буфера, а уже після того буфер можна скопіювати "в один постріл" на екран, при цьому уникнути мерехтіння132. Такий механізм малювання зображення називається подвійною буферизацією і забезпечує кращий результат для користувача, надаючи ефект гладкого малювання133.

4 ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ

4.1 Обсяг пояснювальної записки

Обсяг пояснювальної записки курсової роботи становить не менше 35 аркушів. До цього обсягу не включаються додатки.

Рекомендується дотримуватися такого співвідношення між розділами:

- загальний - 25%;
- розрахунковий - 30%;
- спеціальний - 25%.

4.2 Загальні вимоги

Загальними вимогами до пояснювальної записки є логічна послідовність викладання матеріалу, стисліcть, чіткість і конкретність викладання теоретичних і практичних результатів роботи, доказовість висновків і обґрунтованість рекомендацій.

У тексті пояснювальної записки не вживати звороти із займенниками першої особи, наприклад: « Я вважаю ...» , « Ми вважаємо ...» тощо. Потрібно вести виклад, не вживаючи займенників, наприклад: « Вважаємо …» , « ... знаходимо ...» тощо.

В науково-технічній літературі прийняті невизначено-особова і безособова форма викладення, підкреслюючи об’єктивний характер явищ і процесів.

Наприклад, «обираю програму», «описую алгоритм» писати не доцільно.

Правильно писати, в залежності від часу: «обирається програма» або «описано алгоритм» і т. д.

Пояснювальна записка виконується на одній стороні білого аркушу паперу формату А4 (210 х 297 мм) відповідно до вимог ГОСТ 2.105-95 та ДСТУ 3008-95, українською мовою із застосуванням друкуючих і графічних пристроїв виведення (ГОСТ 2.004 - 88).

Усі текстові документи (за винятком титульного аркуша і завдання) повинні мати рамки за формами 9 і 9а ГОСТ 2.106-96. Відстань від рамки до початку і кінця рядків тексту повинна бути не менше 3 мм, а від верхнього або нижнього рядка тексту до лінії рамки не менше 10 мм. Тому доцільно використовувати такі розміри берегів тексту: верхній – 15 мм, нижній – 30 мм, правий – 10 мм, лівий – 25мм. Вирівнювання

основного тексту - по ширині.

При оформленні пояснювальної записки дотримуватись наступних параметрів: шрифт – Times New Roman, розмір шрифту – 14 пунктів, колір – чорний, міжрядковий інтервал - 1,5, абзацний відступ першого рядка - 1,25 см.

4.3 Основні написи пояснювальної записки

Зміст, розміщення та розміри граф основних написів повинні відповідати формам 2 і 2а (ГОСТ 2.104-2006). Для першого аркуша пояснювальної записки із змістом основний напис повинен бути форми 2, а на інших аркушах змісту і пояснювальної записки – форми 2а. Кожний конструкторський документ згідно з ГОСТ 2.102-68, ГОСТ 2.601-74 та ГОСТ 2.201-80 повинен мати назву та позначення. Для навчальних проектів рекомендується така структура позначень:

2017.КР. 0501. ХХХ. ХХ 00.00 ХХ

1 група 2 група З група 4 група 5 група

Перша група – рік виконання та код виду документа, що розробляється.

Друга група — код напряму: для спеціальності 5.05010101 “Обслуговування програмних систем і комплексів” – 0501.

Третя група – номер групи: ОПК-421- 421.

Четверта група - порядковий номер студента в наказі про затвердження теми.

П'ята група - шифр документа, що входить до складу проекту.

Приклад: 2017.КР.0501.421.01.00.00 ПЗ

4.4 Структурні елементи та розділи

Структурні елементи записки, такі як «ЗМІСТ» , «ВСТУП», « ВИСНОВКИ», «ПЕРЕЛІК ПОСИЛАНЬ», створюють розділи записки і їхні найменування служать заголовками окремих розділів. Такі розділи не мають нумерації.

Заголовки структурних елементів записки і заголовки розділів необхідно розташовувати посередині рядка і друкувати прописними літерами без крапки наприкінці, не підкреслюючи. Розділи пояснювальної записки можуть поділятися на підрозділи, пункти та підпункти, які необхідно починати з абзацного відступу і друкувати малими літерами, крім першої прописної з напівжирним зображенням не підкреслюючи і без крапки наприкінці, заголовки 4-го рівня і нижчих рівнів звичайним. Якщо заголовок складається з двох або більше речень, їх розділяють крапкою.

Переноси слів у заголовку розділу не допускаються. Між заголовком і наступним або попереднім текстом необхідно пропустити один пустий рядок або встановити інтервал перед абзацом заголовка розміром 24 пт, після –18 пт.

Не припускається розміщувати найменування розділу, підрозділу, а також пункту в нижній частині сторінки, якщо після нього розташований тільки один рядок тексту.

4.5 Нумерація сторінок

Нумерувати сторінки необхідно арабськими цифрами, дотримуючись наскрізної нумерації по всьому тексті. Нумерацію на титульному аркуші і завданні до курсової роботи не проставляється, але враховується. Номер сторінки проставляють у графі 7 (форма 2а ГОСТ 2.104-2006).

4.6 Нумерація розділів та підрозділів

Розділи, підрозділи, пункти, підпункти пояснювальної записки нумеруються арабськими цифрами. Розділи пояснювальної записки повинні мати порядкову нумерацію в межах суті пояснювальної записки і позначатися арабськими цифрами без крапки, наприклад, 1, 2, 3. Підрозділи повинні мати порядкову нумерацію в межах кожного розділу. Номери підрозділів складаються з номерів розділів і підрозділів, що розділяються крапкою, наприклад, 1.1, 1.2, 1.3. Номер пункту вміщує номер розділу, підрозділу і пункту, які розділені крапками, наприклад, 3.2.1 – перший пункт другого підрозділу третього розділу.

4.7 Переліки

За необхідністю можуть бути використані переліки. Перед переліком ставлять двокрапку. Перед кожною позицією переліку слід ставити малу літеру української абетки з дужкою, або, не нумеруючи – дефіс (перший рівень деталізації). Для подальшої деталізації переліку треба використовувати арабські цифри з дужкою (другий рівень деталізації). Переліки першого рівня деталізації виконують з абзацного відступу, другого рівня – з відступом відносно місця розташування

переліків першого рівня.

4.8 Ілюстрації

В пояснювальні записці для пояснення тексту, що викладається, повинна бути достатня кількість ілюстрацій – креслень, рисунків, графіків, схем, діаграм, фотознімки чи знімків екрану. Ілюстрації потрібно розташовувати в пояснювальній записці безпосередньо після тексту, у якому вони вперше згадуються, або на наступній сторінці.

На всі ілюстрації мають бути посилання в пояснювальній записці таким чином: « ... на рис. 2.3 ...» або « .. на рисунку 3.4....» , якщо посилання є повторним – « див. рисунок 1.3» . Креслення, рисунки, графіки, схеми, діаграми, розміщені у звіті, мають відповідати вимогам стандартів ЄСКД та ЄСПД.

Ілюстрації повинні мати назву, що розташовують під ілюстрацією. При необхідності під ілюстрацією ще розташовують пояснювальні дані. Ілюстрація позначається словом « Рисунок» , що разом із назвою поміщають після пояснювальних даних, наприклад, « Рисунок 2.1 – Схема алгоритму» .

Ілюстрації нумеруються арабськими цифрами в межах розділу, за винятком ілюстрацій, наведених у додатках. Номер ілюстрації складається з номера розділу і порядкового номера ілюстрації, розділених крапкою. Якщо ілюстрація не вміщується на одній сторінці, можна переносити її на інші сторінки, при цьому назву ілюстрації розташовують на першій сторінці, пояснювальні дані – на кожній сторінці, і під ними вказують: «Рисунок <*номер рисунка*>, аркуш <*номер аркуша*>».

4.9 Таблиці

Таблиці необхідно розташовувати в пояснювальній записці безпосередньо після тексту, у якому вони вперше згадуються, або на наступній сторінці. На всі таблиці повинні бути посилання в пояснювальній записці. Таблиця повинна мати назву, яку друкують малими літерами (крім першої прописної) і розміщують над таблицею. Якщо рядки або графи таблиці виходять за формат сторінки, таблицю

поділяють на частини, розташовуючи одну частину під іншою, або поруч, або переносячи частини таблиці на наступну сторінку.

Назву таблиці вказують один раз зліва над першою частиною таблиці, над іншими частинами пишуть: «Продовження таблиці *<номер таблиці>»* з вказівкою номера таблиці. Таблиці слід нумерувати арабськими цифрами в межах розділу, за винятком таблиць, наведених у додатках. Номер таблиці складається з номера розділу і порядкового номера таблиці, розділених крапкою, наприклад: « Таблиця 2.3 - Назва таблиці» .

У верхній (чи лівій) частині таблиці розміщують заголовок таблиці, в якому вказують назви граф. Діагональний поділ комірок таблиці не допускається. Назви граф пишуть малими буквами, починаючи перше слово графи з великої, у підграфах назви повністю пишуть малими буквами, в кінці крапку не ставлять. Текст в таблицях пишуть з одинарним інтервалом.

Якщо всі параметри величин, які наведені в таблиці, мають одну й ту ж одиницю фізичної величини, то наприкінці назви таблиці після коми розміщують її скорочене позначення (мм). Якщо ж параметри мають різні одиниці фізичних величин, то позначення одиниць записують в заголовках граф після коми (Довжина, мм).

Дані, що наводяться в таблиці, можуть бути словесними і числовими. Слова записують в графах з однієї позиції. Якщо текст складається з одного і більше слів, то при повторенні його замінюють словами “те ж”. При розділенні таблиці горизонтальними лініями - ніякої заміни не виконують.

Числа записують посередині графи так, щоб їх однакові розряди по всій графі були точно один під одним, за винятком випадку, коли вказують інтервал. Інтервал вказують від меншого числа до більшого з тире між ними: 12 – 35 або 122 – 450. Дробові числа наводять у вигляді десяткових дробів, з однаковою кількістю знаків після коми в одній графі. Якщо цифрові чи інші дані в таблиці не наводяться, то ставиться прочерк.

Якщо таблиця не поміщається на одній сторінці, її переносять на наступну, а під заголовком на початку таблиці нумерують стовпці і цю цифрову нумерацію стовпців розміщують в заголовку продовженої таблиці.

4.10 Формули і рівняння

Формули і рівняння розташовують у пояснювальній записці безпосередньо після тексту, у якому вони вперше згадуються, посередині сторінки. Вище і нижче кожної формули повинно бути залишено не менше одного вільного рядка.

Формули і рівняння слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком формул і рівнянь, наведених у додатках. Номер формули або рівняння складається з номера розділу і порядкового номера формули або рівняння, розділених крапкою. Номер формули або рівняння вказують на рівні формули або рівняння в дужках у крайньому правому положенні на рядку.

Пояснення символів і числові коефіцієнти, що входять до формули або рівняння, варто робити безпосередньо після формули або рівняння в тій послідовності, у якій вони надані у формулі або рівнянні. Пояснення значення кожного символу або числового коефіцієнта необхідно давати з нового рядка. Перший рядок пояснення починають з абзацу словом « де» без двокрапки.

Переносити формули або рівняння на інший рядок припускається тільки на знаках виконуваних операцій, причому знак операції на початку такого рядка повторюють. При переносі формули або рівняння на знаку операції множення застосовують знак « х» . Формули, що прямують одна за одною, і не розділені текстом, відокремлюють комою.

4.11 Посилання

Посилання в тексті на джерела необхідно наводити в послідовності їх згадування в записці, вказуючи порядковий номер, виділений двома квадратними дужками, наприклад, « … згідно [4 - 7]…» . Посилання може містити номер тому (якщо він є) і в необхідних випадках номер сторінки, наприклад: [10, т.2, с.85], [2, с. 37].

При посиланні на стандарти і технічні умови вказують лише їх позначення, наприклад, ГОСТ 2.105-95.

4.12 Скорочення

У тексті пояснювальної записки припускається робити тільки загальновживані скорочення (наприклад « і т.д., і т.п., та ін.») або поширені абревіатури. Якщо в пояснювальній записці прийнята специфічна термінологія, або вживаються малопоширені скорочення, нові символи, позначення і ін., їх перелік необхідно надати в окремому списку. Перелік повинен розташовуватися стовпчиком, у якому слова за абеткою наводяться скорочено, а справа - їхнє докладне тлумачення. Якщо спеціальні символи, терміни, скорочення, і позначення наводяться менше трьох разів, перелік їх не складається, а їхнє тлумачення роблять у тексті при першому згадуванні.

4.13 Перелік посилань

Перелік використаної літератури повинен містити лише ті літературні джерела, що використані при виконані курсової роботи, і на які є посилання в тексті пояснювальної записки.

Перелік посилань оформлюється як розділ пояснювальної записки з нової сторінки відповідно до вимог ДСТУ ГОСТ 7.1:2006.

Всі джерела нумерують наскрізно арабськими цифрами в алфавітному порядку.

Про кожен документ подаються такі відомості:

а) Якщо автор один, два або три: прізвище і ініціали першого автора, назва книжки, коса риска, ініціали і прізвища через кому перераховуються в порядку, у якому вони вказані в книжці, крапка, тире, місто видання книжки, двокрапка, назва видавництва, кома, рік випуску, крапка, тире, кількість сторінок, крапка.

Приклади:

Лубківський Р.М. Громове дерево : вибр. твори /Р.М.Лубківський - К. : Український письменник, 2006. - 525 с.

Erdmann K. Regierungsorganisation und Verwaltungsaufbau [Text] / K. Erdmann, W. Schafer, E. Mundhenke. — Heidelberg : D.v. Decker’s Verl., 1996. — 114 p.

б) Якщо чотири автора і більше: назва книжки, коса риска, ініціали і прізвища через кому перераховуються в порядку, у якому вони вказані в книжці, крапка, тире, місто видання книжки, двокрапка, назва видавництва, кома, рік випуску, крапка, тире, кількість сторінок, крапка.

Приклад:

Ремонт машин / О.І. Сідашенко, О.А. Науменко, А.Я. Поліський та Лубківський Р.М. – К.: Урожай, 1994. – 400 с.

Бібліографічний опис роблять мовою документа.

Перелік посилань на електронні ресурси потрібно оформляти наступним чином:

Cisco Catalyst 3750-24FS Switch [Електронний ресурс] – Режим доступу до ресурсу: http://www.cisco.com/c/en/us/support/switches/catalyst-3750-24fs-switch/model.html. – Дата доступу: 15.01.2017. – Заголовок з екрану.

Статистичні дані ЄСПЛ Analysis of statistics 2016 [Електронний ресурс] – Режим доступу до ресурсу: http://www.echr.coe.int/Documents/ Stats_analysis_2016_ENG.pdf. – Дата доступу: 01.02.2017.

Примітка: Слова: «– Заголовок з екрану» пишуться, якщо назва читається в заголовку вікна браузера, якщо НЕ читається, то не пишеться нічого.

4.14 Додатки

Додатки необхідно оформляти як продовження пояснювальної записки на наступних її сторінках, або в окремій частині, розташовуючи додатки в порядку появи посилань на них у тексті пояснювальної записки.

Кожний додаток повинен починатися з нової сторінки і мати заголовок, надрукований вгорі малими літерами з першої прописної, симетрично щодо тексту сторінки. Посередині рядка над заголовком малими літерами з першої прописної повинно бути надруковано слово «Додаток» і прописна буква, що позначає додаток. Додатки слід позначати послідовно прописними буквами українського алфавіту, за винятком букв Ґ, Є, З, І, Ї, Й, О, Ч, Ь. Один додаток позначається як додаток А.

Додатки повинні мати загальну з іншою частиною пояснювальної записки наскрізну нумерацію. Текст кожного додатка, при необхідності, може бути розділений на підрозділи і пункти, що нумеруються арабськими цифрами в межах кожного додатка. Всі додатки повинні бути перелічені у змісті пояснювальної записки із зазначенням їх позначень та заголовків.

Ілюстрації, таблиці, формули та рівняння, що є у тексті додатка, слід нумерувати в межах кожного додатка, наприклад, рисунок Г.3 – третій рисунок додатка Г; таблиця А.2 – друга таблиця додатка А; формула (А.1) – перша формула

додатка А.

Висновки

Для виконання курсової роботи необхідно провести аналіз предметної області, виконати постановку завдання, створити програму, що реалізує поставлені мети, провести аналіз її роботи. Для цього доцільно використовувати сучасне середовище програмування, наприклад NetBeans IDE від Oracle, або IntelliJ IDEA від JetBrains. При розробці програми доцільно дотримуватися шаблону проектування MVC, що дозволяє раціонально розподілити завдання між класами програми.

ДОДАТОК А

Зразок титульного аркуша курсової роботи

Міністерство освіти і науки України

Технічний коледж Тернопільського національного технічного університету імені Івана Пулюя

(повне найменування вищого навчального закладу)

відділення електронних апаратів

(назва відділення)

циклова комісія програмних систем і комплексів

(повна назва циклової комісії)

Курсова робота

з дисципліни :

«РОЗРОБКА КЛІЄНТ-СЕРВЕРНИХ ЗАСТОСУВАНЬ»

на тему:

РОЗРОБКА КЛІЄНТ-СЕРВЕРНОГО ЗАСТОСУВАННЯ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТІВ КОЛЕДЖУ»

Виконав: студент(ка) 4 курсу, групи ОПК-421

Попович О. В.

(прізвище та ініціали)

Напрям підготовки:
6.050101 «Комп'ютерні науки»
Спеціальність:
5.05010101 «Обслуговування програмних систем і комплексів»

Керівник Ляпандра А.С.

Національна шкала: _____

Кількість балів: _____

Оцінка: ECTS: _____

Члени комісії:

_____ Ляпандра А.С.

(підпис) (прізвище та ініціали)

_____ _____

(підпис) (прізвище та ініціали)

Тернопіль – 2017 р.

ДОДАТОК Б
Зразок завдання до курсової роботи

Міністерство освіти і науки України

Технічний коледж Тернопільського національного технічного університету імені Івана Пулюя

(повне найменування вищого навчального закладу)

відділення електронних апаратів

(назва відділення)

циклова комісія програмних систем і комплексів

(повна назва циклової комісії)

Напрям підготовки: 6.050101 «Комп'ютерні науки»

Спеціальність:5.05010101 «Обслуговування програмних систем і комплексів»

ЗАВДАННЯ

на курсову роботу

з дисципліни :

«РОЗРОБКА КЛІЄНТ-СЕРВЕРНИХ ЗАСТОСУВАНЬ»

студента(ки) 4 курсу, групи ОПК-421

Попович Олена Володимирівна

(прізвище, ім'я, по батькові)

Тема роботи:
«Розробка клієнт-серверного застосування системи тестування знань студентів коледжу»

Вихідні дані:

Зміст пояснювальної записки

1.Загальний розділ проекту

2 Розробка програмного забезпечення

3 Спеціальний розділ

Висновки

Додаткові вказівки:
Виконання проекту (з виготовленням макета, стенда та ін.) _____
_____
_____
Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсової роботи	Строк виконання етапів проекту	Примітка
1	Отримання і аналіз технічного завдання	___.___.2017	
2	Збір і узагальнення інформації	___.___.2017	
3	Написання першого розділу	___.___.2017	
4	Розробка програмного забезпечення	___.___.2017	
5	Написання спеціального розділу	___.___.2017	
6	Виконання графічної частини	___.___.2017	
7	Оформлення курсової роботи	___.___.2017	
8	Захист курсової роботи	___.___.2017	

Дата видачі завдання "____" _____2017р.

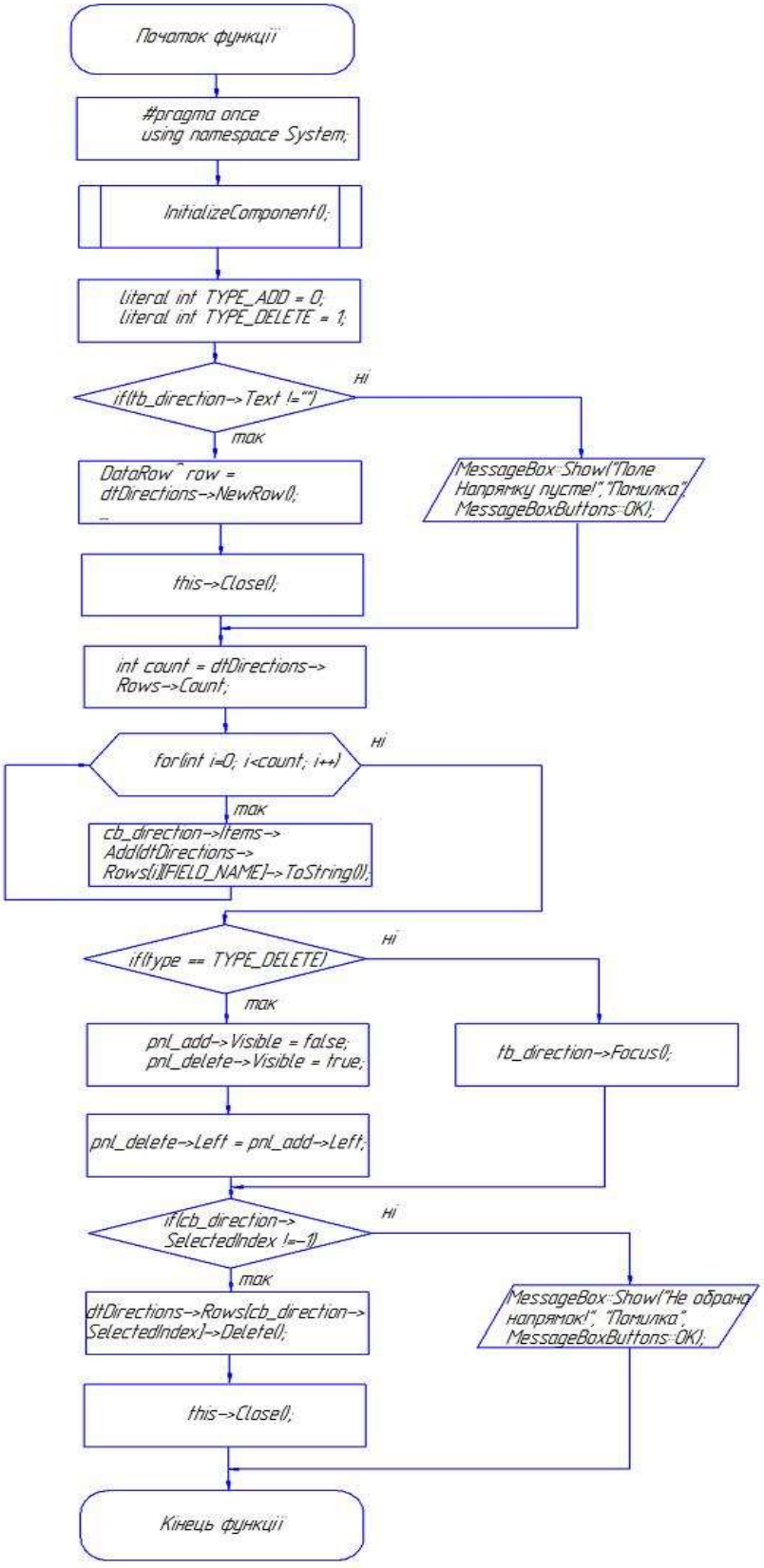
Керівник курсової роботи: _____ _Ляпандра А.С._
(підпис) (прізвище та ініціали)

Протокол №___ від "___" _____2017р.

Голова ЦК _____ Марціяш Г.Я.
(підпис) (прізвище та ініціали)

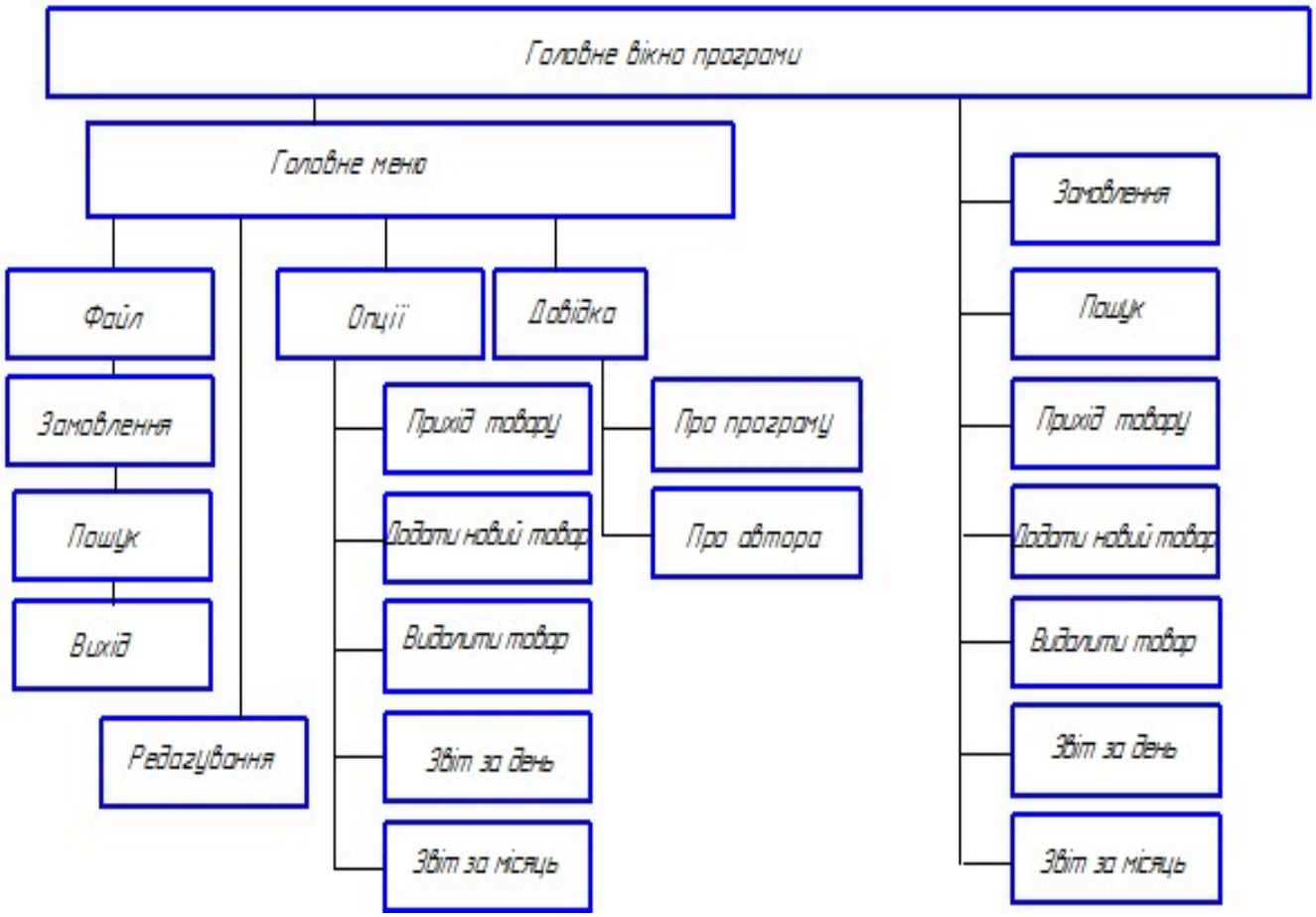
ДОДАТОК В

Зразок блок-схеми алгоритму функції



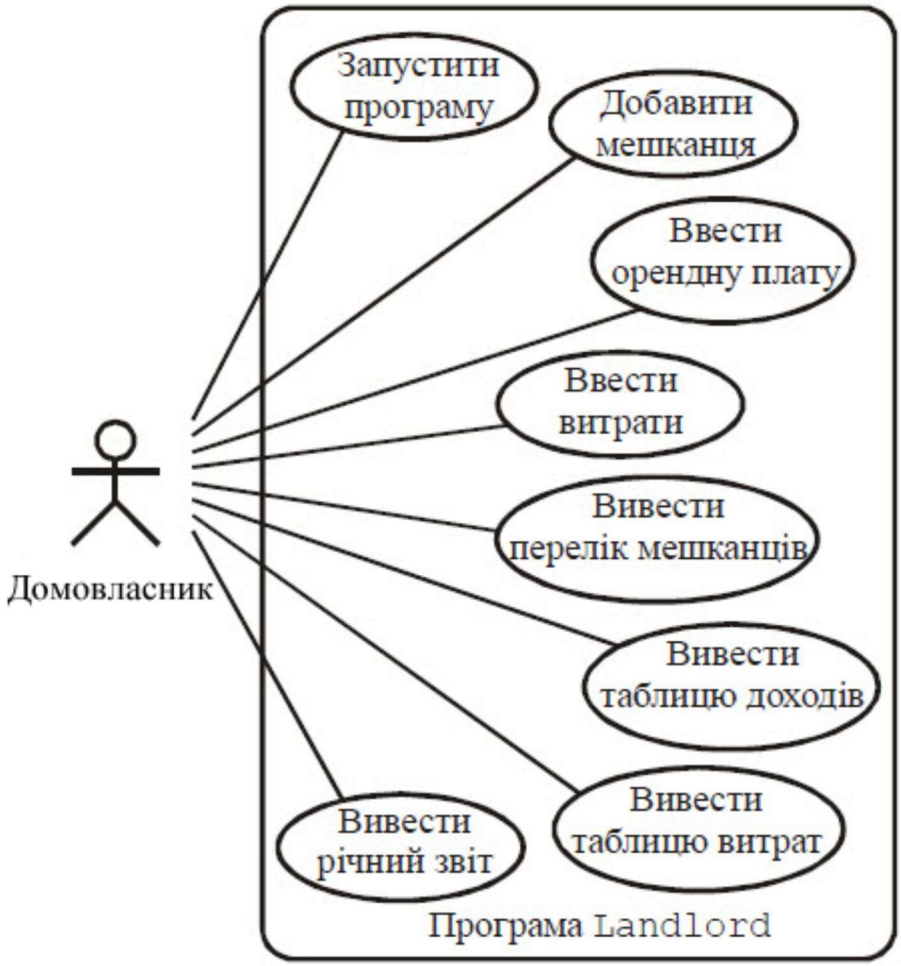
ДОДАТОК Г

Зразок структурної схеми програми



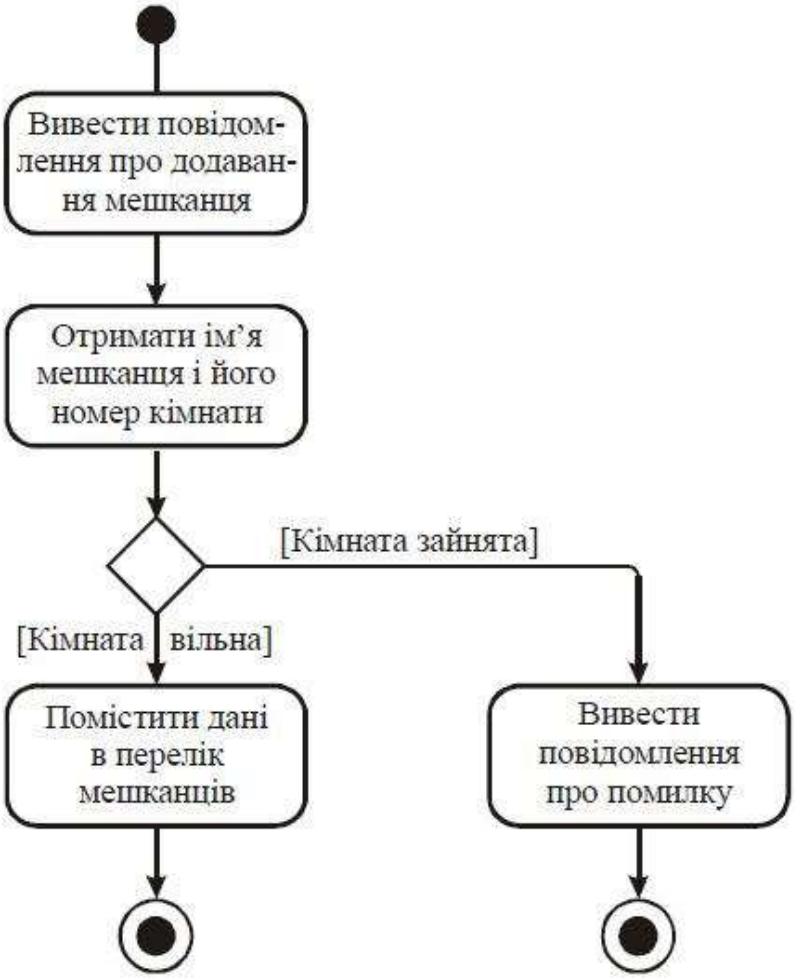
ДОДАТОК Д

Зразок UML-діаграми варіантів використання програми



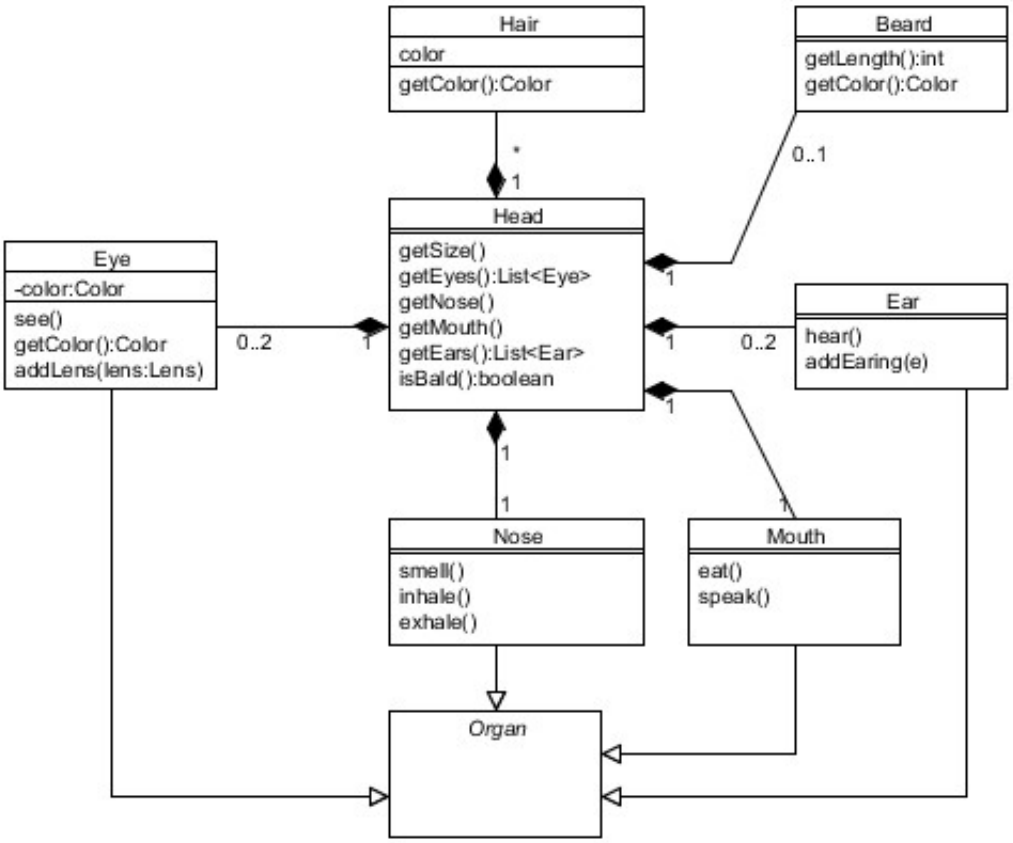
ДОДАТОК Е

Зразок UML-діаграми послідовності дій



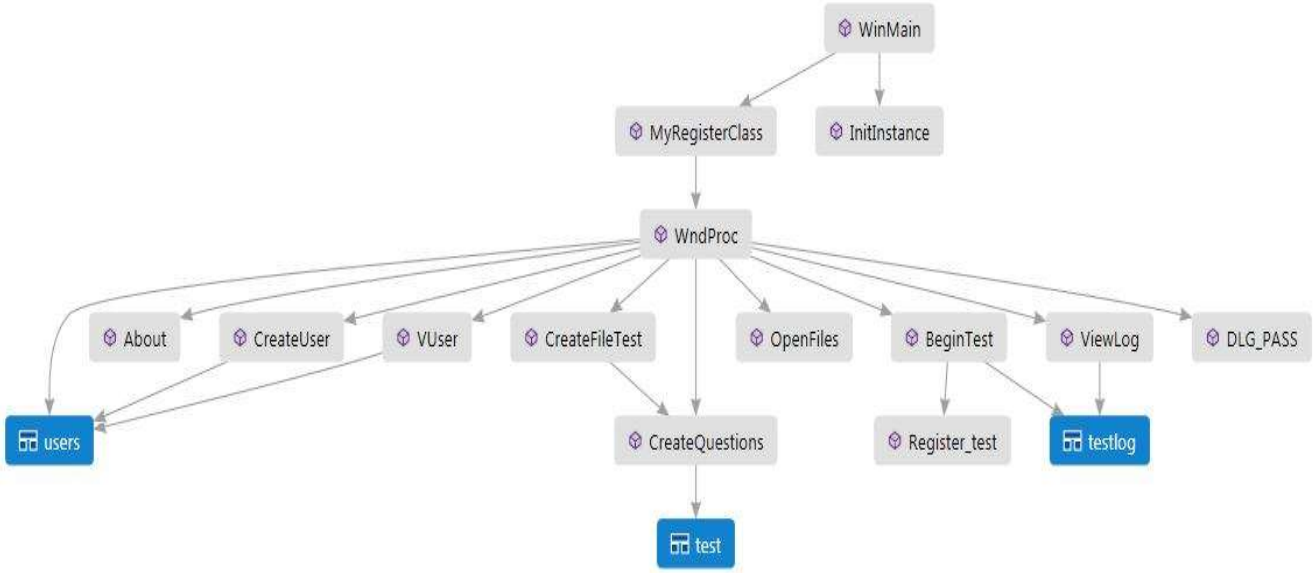
ДОДАТОК Ж

Зразок UML-діаграми класів програми



ДОДАТОК И

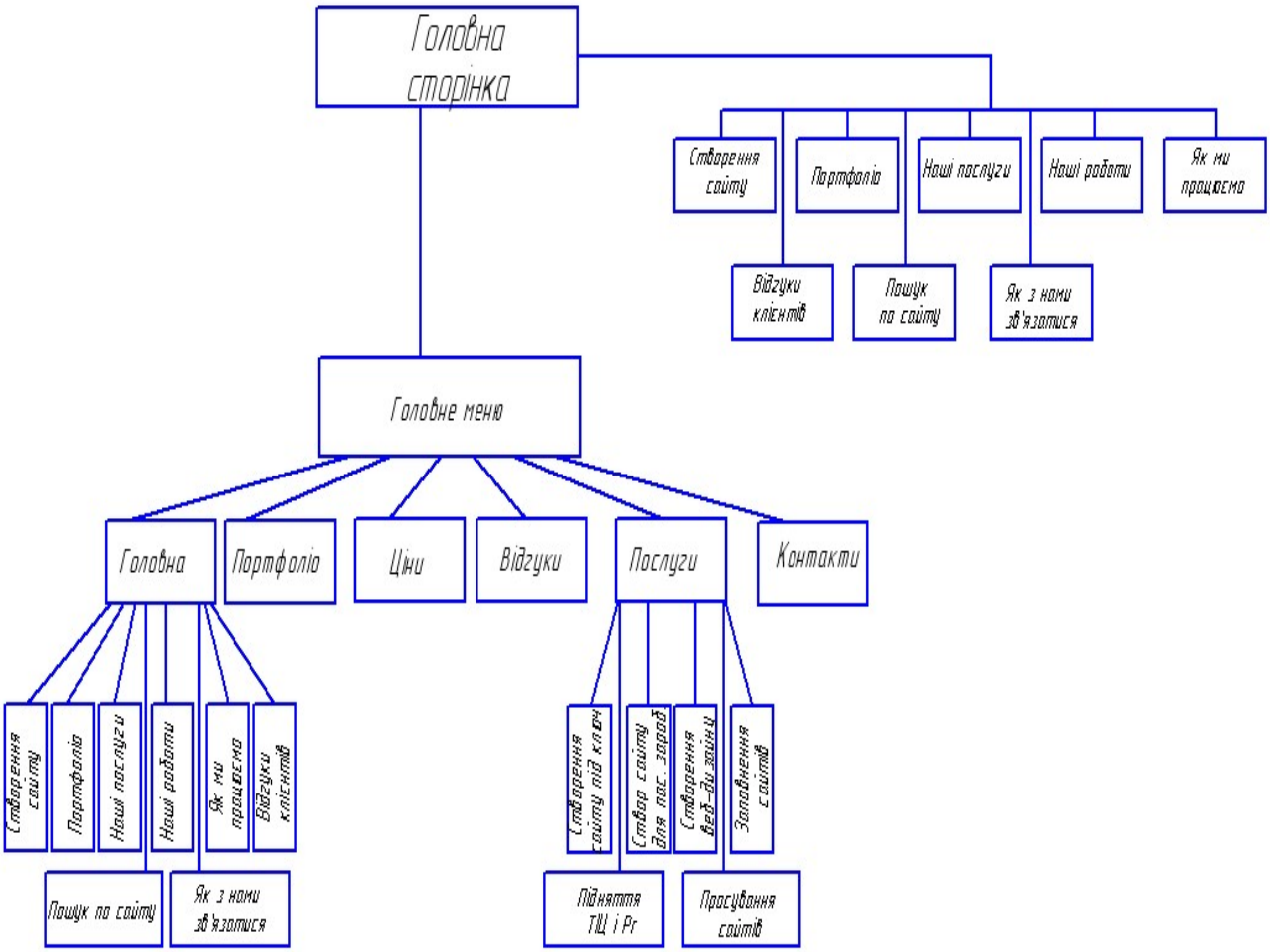
Зразок UML-діаграми взаємозвязків функцій



№ п.п	Назва	Призначення
1	WinMain	Головна функція
2	MyRegisterClass()	Реєстрація класу вікна
3	WndProc()	Обробка повідомлень головного вікна
4	About()	Обробка повідомлень вікна Про програму
5	…	…

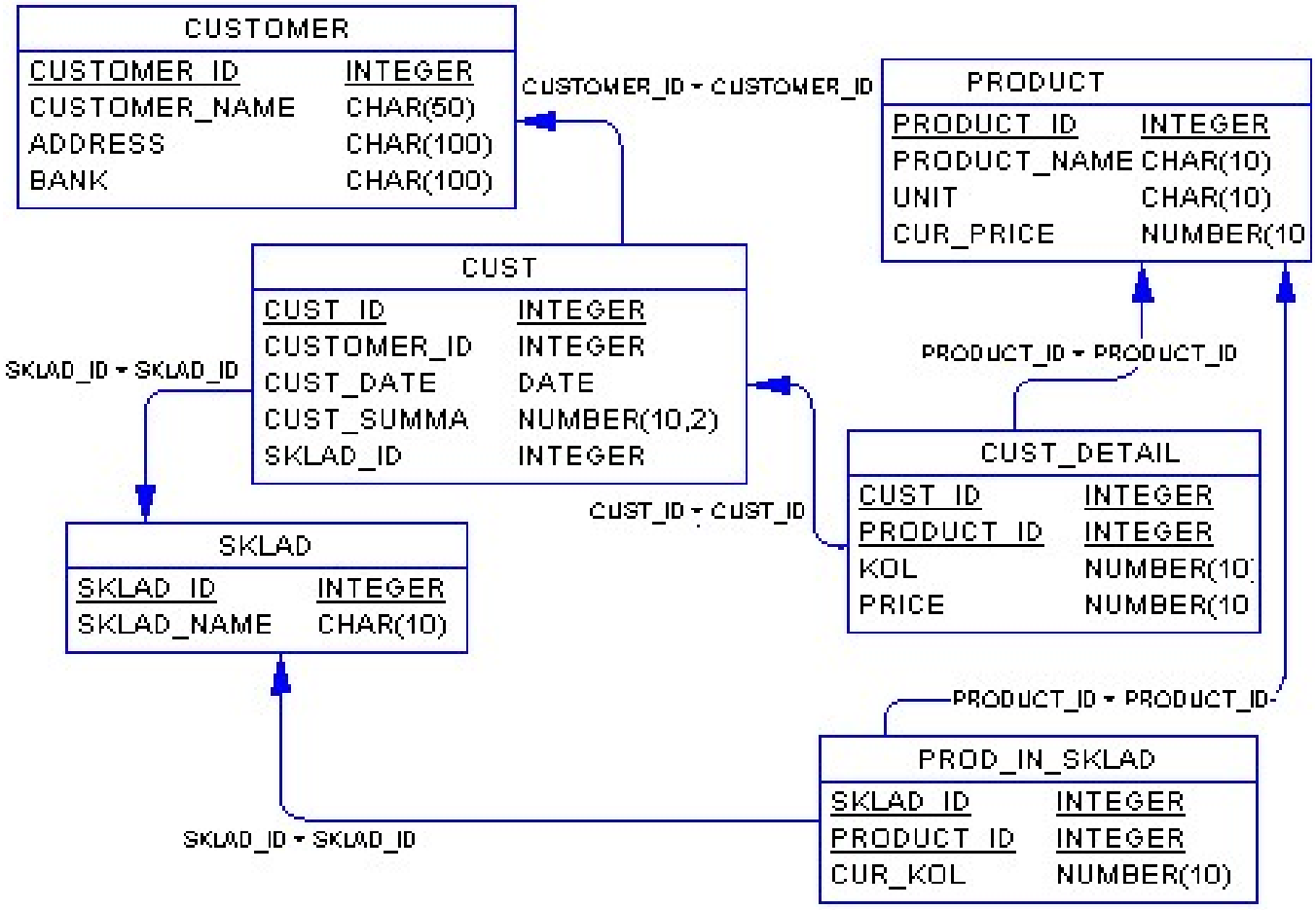
ДОДАТОК К

Зразок структурної схеми головної сторінки сайту



ДОДАТОК Л

Зразок ER-діаграми бази даних



Додаток М

Перелік тем курсових робіт з дисципліни ”Розробка клієнт-серверних застосувань”

1. Розробка клієнт-серверного застосування обліку абонентів оператора зв’язку.
2. Розробка клієнт-серверного застосування автоматичної системи управління мікрокліматом.
3. Розробка клієнт-серверного застосування видачі дозвільних документів посольством країни.
4. Розробка клієнт-серверного застосування інформаційного забезпечення автоматизованого робочого місця менеджера по персоналу.
5. Розробка клієнт-серверного застосування контролю витрат сировини.
6. Розробка клієнт-серверного застосування моніторигу елементів автоматичних систем протипожежного захисту будівлі.
7. Розробка клієнт-серверного застосування інформаційної системи калькуляції собівартості продукції на підприємстві
8. Розробка клієнт-серверного застосування інформаційної системи контролю якості підготовки фахівців програмної інженерії.
9. Розробка клієнт-серверного застосування інформаційної підсистеми планування та контролю навчального процесу циклової комісії.
10. Розробка клієнт-серверного застосування інформаційної системи транспортної логістики підприємства.
11. Розробка клієнт-серверного застосування інформаційної системи диспетчерування навчального процесу.
12. Розробка клієнт-серверного застосування категоризації інформації з обмеженим доступом.
13. Розробка клієнт-серверного застосування моніторингу комплектуючих комп’ютерної техніки.
14. Розробка клієнт-серверного застосування аналізу контингенту строкової та контрактної служби військового комісаріату.
15. Розробка клієнт-серверного застосування контролю виконання логічних операцій операційного пристрою комп’ютера.
16. Розробка клієнт-серверного застосування монторингу літальних апаратів.
17. Розробка клієнт-серверного застосування відслідковування наявності мікропроцесорних пристроїв.
18. Розробка клієнт-серверного застосування відділу науково-технічної інформації та патентів.
19. Розробка клієнт-серверного застосування системи тестування знань студентів коледжу.
20. Розробка клієнт-серверного застосування обліку замовлень в ІТ-компанії.
21. Розробка клієнт-серверного застосування обліку навчальної літератури в бібліотеці.
22. Розробка клієнт-серверного застосування підприємства з реалізації комп‘ютерної техніки.
23. Розробка клієнт-серверного застосування планування навантаження викладачів.
24. Розробка клієнт-серверного застосування сервісного обслуговування пристроїв радіозв’язку.