МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

Є. Ю. БЕРКУНСЬКИЙ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання курсових робіт з дисципліни "Мова програмування Java"

Рекомендовано Методичною радою НУК

Електронне видання
комбінованого використання на DVD-ROM

МИКОЛАЇВ • НУК • 2015

УДК 004.43(076)
ББК 32.973-01я73
Б 48

Укладач Є. Ю. Беркунський, старш. викладач

Рецензент К. В. Кошкін, д-р техн. наук, професор

Беркунський Є. Ю.

Б 48 Методичні вказівки до виконання курсових робіт з дисципліни "Мова програмування Java" / Є. Ю. Беркунський. – Миколаїв : НУК, 2015. – 29 с.

Наведено теоретичні відомості та довідкову інформацію для виконання завдань курсових робіт з дисципліни, а саме основні відомості про структуру графічної бібліотеки Swing, використання шаблону проектування MVC, середовищ розробки мовою Java та відповіді на основні питання, пов'язані з організацією s методикою виконання лабораторних робіт.

Призначено для студентів денної і заочної форм навчання спеціальності 8.05010101 "Інформаційні управляючі системи та технології". Можуть бути використані студентами інших спеціальностей.

УДК 004.43(076)
ББК 32.973-01я73

Навчальне видання

БЕРКУНСЬКИЙ Євген Юрійович

МЕТОДИЧНІ ВКАЗІВКИ
до виконання курсових робіт з дисципліни
"Мова програмування Java"

Комп'ютерне верстання *А. Й. Лихіна*
Коректор *М. О. Паненко*

© Беркунський Є. Ю., 2015
© Національний університет кораблебудування
імені адмірала Макарова, 2015

Формат 60×84/16. Ум. друк. арк. 1,7. Обсяг даних 2488 кб. Тираж 15 прим.
Вид. № 45. Зам. № 68.

Видавець і виготівник Національний університет кораблебудування імені адмірала Макарова
просп. Героїв Сталінграда, 9, м. Миколаїв, 54025, e-mail : publishing@nuos.edu.ua

Свідоцтво суб'єкта видавничої справи ДК № 2506 від 25.05.2006 р.

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. 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. Хоча Java-аплети рідко використовуються в настільних комп'ютерах, проте вони в них іноді використовуються для поліпшення функціональності і підвищення безпеки при перегляді всесвітньої павутини.

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

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

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

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. Використовується для розробки програм для мобільних пристроїв, КПК, TV-ресиверів і принтерів.

Java SE – для використання на ПК, ноутбуках, та невеликих серверах.

Java EE – Java SE плюс API, корисне для багатопотокових клієнт-серверних бізнес-застосувань.

У курсовій роботі розроблюється програма для платформи Java SE.

1 Порядок виконання курсової роботи

1.1. Загальні положення

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

1. Опис та аналіз предметної області
2. Постановка задачі на розробку програмного забезпечення
3. Розробка програмного забезпечення
4. Аналіз отриманих результатів
5. Список використаних джерел інформацій
6. Додатки (технічний опис, інструкція з експлуатації, код програми, знімки екранних форм і т. п.).

1.2. Опис розділів

1.2.1. Опис та аналіз предметної області

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

1. Визначення мети та призначення розробки, виявлення предметної області проекту.

2. Збирання інформації по темі роботи, аналіз існуючих у даній галузі рішень. Головною метою даного етапу є виявлення основних характерних рис, технологій, архітектури та дизайну проекту.

3. Вибір та обґрунтування критеріїв якості програмного забезпечення, формування вимог і обмежень.

4. Визначення рішень, технологій (або навіть програмних бібліотек), які присутні у готових продуктах та можуть бути застосовані в даній розробці.

5. Визначення напрямків досліджень для реалізації механізмів, що не є очевидними та не можуть бути повторно використаними чи запозиченими.

Після аналізу предметної області визначаються можливості використання певних технологій проектування. На цьому етапі треба також враховувати архітектурні та функціональні вимоги до проекту, кількість

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

1.2.2. Постановка задачі на розробку програмного забезпечення

На основі результатів, отриманих внаслідок аналізу предметної області, виконується розробка технічного завдання (постановка задачі). 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 тестування.

Проект програми

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 програми.

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

Після вибору технології проектування треба визначити інструментальні засоби створення проекту. 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 2014–2015 рік це – JDK8).

За останні роки для будь-якої мови програмування створено велику кількість бібліотек. Навіть стандартна бібліотека мови Java складається з декількох тисяч класів, що дозволяють вирішувати найрізноманітніші завдання. Щодо середовища програмування, воно повинне бути інтуїтивно зрозумілим, зручним та швидким у роботі. Фактично, стандартни-

ми середовищами для мови програмування Java є NetBeans IDE – вільне безкоштовне середовище, та IntelliJ IDEA – багатофункціональне комерційне середовище, проте безкоштовне для використання в освітніх закладах.

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

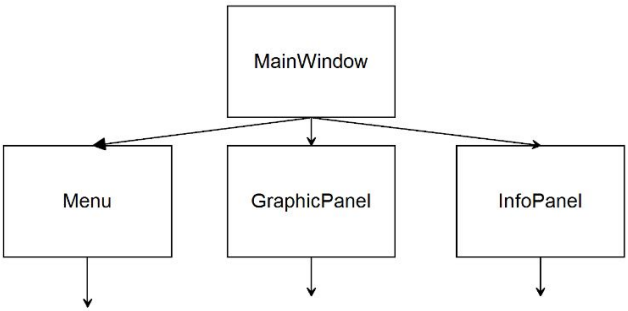

Рис. 1. Ієрархія об'єктів програми

На основі ієрархії об'єктів, описаної в попередньому розділі, розробляється ієрархія успадкування класів проекту. Опис ієрархії успадкування повинен включати специфікації класів та інтерфейсів. 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) класів. У прямокутниках необхідно вказати назви відповідних класів. Приклад діаграми – на рис. 2.

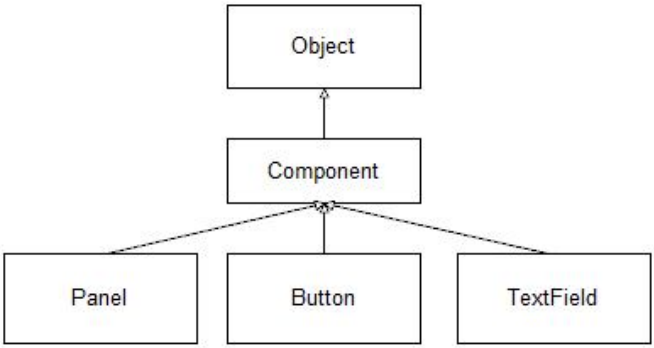

Рис. 2. Ієрархія успадкування класів

Після опису ієрархії успадкування класів необхідно провести опис класів, їхніх полів та методів. 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 рішень. Важливі для проекту алгоритми потрібно ілюструвати за допомогою блок-схем або діаграм взаємодії (UML). Якщо в розробці було використано готові програмні рішення інших розробників (бібліотеки, фреймворки та ін.), необхідно навести короткі описи цих рішень з обґрунтуванням доцільності їхнього вибору.

1.2.3. Розробка програмного забезпечення

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

1.2.4. Аналіз отриманих результатів

Основне призначення такого аналізу – визначення, в якій мірі результати виконаної роботи відповідають початковим вимогам до програмно-

го продукту і поточного стану справ у предметній області. Також цей аналіз може бути використаний для формування на його основі планів і перспективних напрямків розвитку проекту або його частин. 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 функцій. Також не виключається ситуація, що під час роботи над проектом у предметній області відбулись важливі для проекту зміни.

1.2.5. Список літератури

Впорядкований список літератури повинен бути пронумерований арабськими цифрами з крапкою. Основні вимоги до списку літератури:

- відповідність темі курсової роботи;
- різноманітність видів видань (нормативні, навчальні, наукові та ін.);
- перелік літератури подається в алфавітному порядку. Посилання на неї в тексті беруться у квадратні дужки. На неопубліковані роботи посилання не допускаються;
- бібліографічний опис літератури наводять згідно з діючим стандартом ГОСТ 7.1-2006 для бібліографічної та видавничої справи.

1.2.6. Додатки

Тексти програм повинні починатися з найменування програмних модулів (класів, файлів). Кожен з них повинен починатися з нової сторінки. 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. При написанні програм бажано враховувати необхідність друку кодів програмних модулів та, за можливістю, не припускати розривання рядків коду засобами друку текстів.

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

2 Графічна бібліотека Swing

2.1. Обґрунтування використання графічної бібліотеки

Оскільки завданням курсової роботи з дисципліни "Мова програмування Java" 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 бібліотек.

2.2. Огляд існуючих графічних бібліотек для Java

Сучасні програми потребують графічного інтерфейсу користувача (GUI). 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 т. д.

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

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

Кожна бібліотека надає набір класів для роботи з кнопками, списками, вікнами, меню і т. д., 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 просто. 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 хитрощі.

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

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).

2.3.1. Історія виникнення

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

В оригінальній концепції була описана сама ідея і роль кожного з елементів: моделі, представлення та контролера. Але зв'язки між ними були описані без конкретизації. 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. Всі зміни моделі відслідковуються контролером і він же відповідає за відображення представлення, якщо це необхідно. 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. Це дозволяє зберегти незалежність моделі як від контролера, так і від представлення.

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

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.

2.3.2. Призначення

Основна мета застосування цієї концепції полягає в розділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого поділу підвищується можливість повторного використання класів програми. 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 моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми та кругової діаграми.

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 контролер.

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

2.3.3. Концепція

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

модель (англ. ***Model***). Модель надає інформацію: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить відомостей, як цю інформацію можна візуалізувати;

представлення, іноді його називають "вид" (англ. ***View***). Відповідає за відображення інформації (візуалізацію). Часто як представлення виступає вікно або панель з графічними елементами;

контролер (англ. ***Controller***). Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель і представлення для реалізації необхідної реакції.

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

Для реалізації схеми Model-View-Controller використовуються традиційні шаблони проектування, основні з яких "Observer", "Strategy", "Composite".

Найбільш типова реалізація відокремлює представлення від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи механізм "підписка – сповіщення". При кожній зміні внутрішніх даних

в моделі, вона оповіщає всі залежні від неї представлення, і представлення оновлюються. Для цього використовується шаблон "Observer". При обробці реакції користувача представлення вибирає, в залежності від потрібної реакції, потрібний контролер, який забезпечить той чи інший зв'язок з моделлю. Для цього використовується шаблон "Strategy", або замість цього може бути модифікація з використанням шаблону "Command". Якщо представлення має складну ієрархічну структуру, то для можливості однотипної поведінки підоб'єктами такого представлення може використовуватися шаблон "Composite". Крім того, можуть використовуватися й інші шаблони проектування, наприклад, "Factory method", який дозволить задати за замовчуванням тип контролера для відповідного представлення.

2.3.4. Вікно JFrame

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

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

Конструктор `JFrame()` без параметрів створює порожнє вікно. Конструктор `JFrame(String title)` створює порожнє вікно з заголовком title.

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

`setSize (int width, int height)` – встановлює розміри вікна. 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.

`setDefaultCloseOperation (int operation)` – дозволяє вказати дію, яку необхідно виконати, коли користувач закриває вікно натисканням на хрестик. Зазвичай в програмі є одне або кілька вікон при закритті яких програма припиняє роботу. Для того, щоб запрограмувати це поведінка, слід як параметр operation передати константу `EXIT_ON_CLOSE`, описану в класі JFrame.

`setVisible(boolean visible)` – коли вікно створюється, воно за замовчуванням невидиме. Щоб відобразити вікно на екрані, викликається даний метод з параметром `true`. Якщо викликати його з параметром `false`, вікно знову стане невидимим.

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

```
import javax.swing.*;
public class MyClass {
   public static void main (String [] args) {
   JFrame myWindow = new JFrame("Пробне вікно");
   myWindow.setDefaultCloseOperation(JFrame.EXIT_ON_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 програмці. Необхідно створити безліч елементів управління, налаштувати їх зовнішній вигляд, розмістити в потрібних місцях вікна. 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 обробляють. Логічніше було б, щоб кожне вікно займалося своїми розмірами і вмістом самостійно. Тому класична структура програми з вікнами виглядає наступним чином:

Файл SimpleWindow.java:

```
public class SimpleWindow extends JFrame {
   SimpleWindow(){
      super("Пробне 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). Метод main() міститься в іншому класі, відповідальному за управління ходом програми. Кожен з цих класів дуже простий, кожен займається своєю справою, тому в них легко розбиратися і легко супроводжувати (тобто удосконалювати при необхідності).

Примітка. *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 програми.*

2.3.5. Панель вмісту вікна

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

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

Додамо кнопку в панель вмісту вікна командами:

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

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

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

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 панелі.

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 рядком. Наприклад, текстові мітки, іконки, текстові поля і кнопки будуть додаватися в перший умовний рядок, поки в ньому є місце. Коли перший рядок заповниться, компоненти, що залишилися будуть додаватися до наступного рядка і так далі. Якщо користувач змінить розмір вікна, картина може змінитися.

Рис. 3. Початкове розміщення компонентів

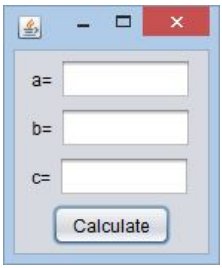

Рис. 4. Автоматичне переміщення компонентів

Якщо потягнути за кут вікна, його розмір зміниться, при цьому `FlowLayout` перемістить елементи вікна відповідно до зміни розмірів.

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

```
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
```

```
        jTextField3 = new javax.swing.JTextField();
            jButton1 = 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

            jLabel1.setText("a=");
            jPanel1.add(jLabel1);

            jTextField1.setColumns(7);
            jPanel1.add(jTextField1);

            jLabel2.setText("b=");
            jPanel1.add(jLabel2);

            jTextField2.setColumns(7);
            jPanel1.add(jTextField2);

            jLabel3.setText("c=");
            jPanel1.add(jLabel3);

            jTextField3.setColumns(7);
            jPanel1.add(jTextField3);

            jButton1.setText("Calculate");
            jPanel1.add(jButton1);

            getContentPane().add(
            jPanel1, java.awt.BorderLayout.CENTER);

            pack();
        }
```

GridLayout – табличне розташування

Клас java.awt.GridLayout дозволяє організувати компоненти, як рядки і стовпці в таблиці. Компоненти будуть додаватися в комірки умовної таблиці. Якщо розмір вікна буде збільшений, комірки стануть більше, але положення компонентів відносно один одного залишиться тим самим. 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):

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

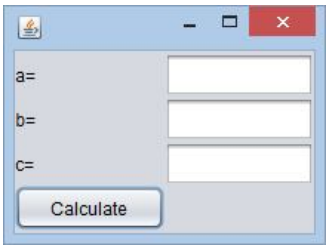

Рис. 5. Сіткове розміщення

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);
```

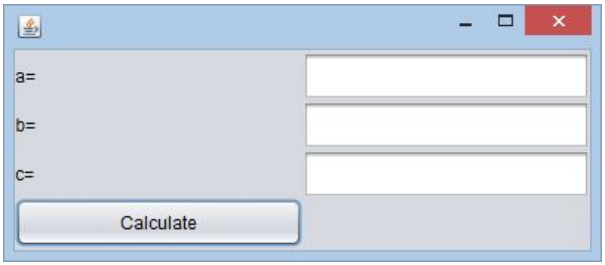

Рис. 6. Зміна розмірів при сітковому розміщенні

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

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). На відміну від менеджера FlowLayout, коли вікно з BoxLayout змінює свій розмір, його елементи керування не зміщуються зі своїх позицій. BoxLayout дозволяє елементам вікна мати різні розміри (чого не дозволяє GridLayout). Наступні два рядки коду задають BoxLayout з вертикальним вирівнюванням на JPanel.

```
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().

GridBagLayout – більш гнучке табличне розташування

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

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 Windows. Як випливає із документації, GroupLayout створювався для використання у спеціалізованих середовищах розробки. Наприклад, середовище NetBeans дозволяє будувати інтерфейси користувача із використанням цього менеджера компоновки за допомогою спеціалізованого дизайнера користувальницького інтерфейсу. Якщо подивитись на код програми, що згенерований за допомогою дизайнера інтерфейсу в NetBeans, можна побачити, що він досить складний і заплутаний. Його майже

неможливо редагувати вручну. 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 сенс.

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".

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

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

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

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

Група компонентів JRadioButton використовується для представлення користувачам набору взаємовиключних варіантів.

Об'єкти класу JComboBox використовуються, щоб надати користувачеві вибір одного з набору варіантів, де користувач, можливо, може ввести нове значення. 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 вибір.

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

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

Об'єкт класу JScrollBar використовується, щоб забезпечити можливість прокрутки для перегляду компоненту, який більше за розміром, ніж доступний вільний простір. JScrollBar може бути встановлений вертикально або горизонтально. Скролінг здійснюється шляхом перетягування

"ручки" вздовж доріжки JScrollBar. Для його правильної роботи потрібно написати логіку для забезпечення можливості прокрутки за допомогою компонента JScrollBar.

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

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

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

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

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

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

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

JToolBar – панель кнопок представляє собою групу невеликих кнопок, які забезпечують найбільш часто використовувані дії для користувача в JFrame. Як правило, панель кнопок використовується разом з меню.

Об'єкт класу JTable використовується для відображення і редагування даних в табличній формі. Він представляє дані у вигляді рядків і стовпців. Кожна колонка має заголовок стовпця. Рядки і стовпці використовують індекси, починаючи з 0.

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

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

Об'єкти класу JSplitPane – розгалужувачі, які можуть бути використані, щоб розділити простір між двома компонентами. Стрічка розділювача може відображатися горизонтально або вертикально. 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 інший. Якщо є достатньо місця, обидва компоненти можуть бути показані повністю.

Об'єкти класу JDialog у бібліотеці Swing – контейнери верхнього рівня. 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 користувача. Об'єкт класу JOptionPane забезпечує багато статичних методів, щоб показати різні типи діалогів для користувача за допомогою екземпляра класу JDialog.

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

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

Бібліотека Swing дозволяє встановити кольори фону і переднього плану компонента. Для цього використовується об'єкт класу java.awt.Color. Він представляє собою певний колір. Ви можете вказати колір за допомогою червоного, зеленого, синього та альфа-компонентів або використовуючи компоненти відтінку, насиченості і яскравості. Об'єкт класу Color – незмінний. Він забезпечує декілька констант, які представляють часто використовувані кольори, наприклад, Color.RED, Color.BLUE представляють червоний та синій кольори.

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

Swing дозволяє встановити шрифт для тексту, відображуваного в компонентах. Для цього використовується клас java.awt.Font – він представляє шрифт у програмі Java.

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

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

Перемалювання деталей може бути виконане на екрані або поза екраном. Якщо виконувати малювання безпосередньо на екрані, це може призвести до мерехтіння. 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 мерехтіння. Такий механізм малювання зображення називається подвійною буферизацією і забезпечує кращий результат для користувача, надаючи ефект гладкого малювання.

Висновки

Для виконання курсової роботи з дисципліни "Мова програмування Java" 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. Приблизний перелік завдань для курсової роботи

Таблиця А1. **Варіанти завдань для курсової роботи**

№	Тип діаграми	Параметри
1	Гістограма	Зміна курсу валют (USD, EUR, RUR) за три декади, $d_{заповнення}$=0.7. Дані нанести на піктограми; піктограми примкнути до відмітки на осях справа
2	Об'ємна гістограма	Зміна курсу валют (USD, EUR, RUR) за три декади, $d_{заповнення}$=0.9. Дані нанести над піктограмами; піктограми примкнути зліва до відміток на осях
3	Лінійчата діаграма	Зміна курсу валют (USD, EUR, RUR) за три декади, $d_{заповнення}$=0.6. Дані нанести на піктограми; піктограми примкнути зверху до відмітки на осях.
4	Об'ємна лінійчата діаграма	Зміна курсу валют (USD, EUR, RUR) за три декади, $d_{заповнення}$=0.7. Дані нанести справа від піктограм; піктограми примкнути знизу до відмітки на осях.
5	Об'ємна гістограма	Зміна середньорічної температури по місяцях, $d_{заповнення}$=0.6. Дані нанести на піктограми; піктограми примкнути до відмітки на осях справа.
6	Лінійчата діаграма	Зміна середньорічної температури по місяцях, $d_{заповнення}$=0.8. Дані нанести на піктограми; піктограми примкнути до відмітки на осях зверху.
7	Гістограма	Зміна середньорічної температури по місяцях, $d_{заповнення}$=0.7. Дані нанести на піктограми; піктограми примкнути до відмітки на осях справа.
8	Об'ємна лінійчата діаграма	Зміна середньорічної температури по місяцях, $d_{заповнення}$=0.6. Дані нанести на піктограми; піктограми примкнути до відмітки на осях зверху.
9	Лінійчата діаграма	Зміна урожайності зернових за сім останніх років (ц/га), $d_{заповнення}$=0.9. Дані нанести на піктограми; піктограми примкнути до відмітки на осях знизу.

Продовж. табл. А1.

№	Тип діаграми	Параметри
10	Об'ємна гістограма	Зміна урожайності зернових за сім останніх років (ц/га), $d_{заповнення}$=0.6. Дані нанести над піктограмами; піктограми примкнути до відмітки на осях зліва.
11	Кругова діаграма	Структура земельних ресурсів країни (ліси, пасовища, пашні та плантації, інші землі). Дані нанести поруч із діаграмою.
12	Розрізна кругова діаграма	Структура земельних ресурсів країни (ліси, пасовища, пашні та плантації, інші землі). Дані нанести поруч із діаграмою. Виділити у відокремлений сектор інші землі.
13	Об'ємна кругова діаграма	Структура земельних ресурсів країни (ліси, пасовища, пашні та плантації, інші землі). Дані нанести поруч із діаграмою.
14	Об'ємна розрізна кругова діаграма	Структура земельних ресурсів країни (ліси, пасовища, пашні та плантації, інші землі). Дані нанести поруч із діаграмою. Виділити у відокремлений сектор інші землі.
15	Кругова діаграма	Вартість комплектуючих у складі комп'ютера (процесор, материнська плата, оперативна пам'ять, жорсткий диск, корпус і т.н.)
16	Розрізна кругова діаграма	Вартість комплектуючих у складі комп'ютера (процесор, материнська плата, оперативна пам'ять, жорсткий диск, корпус і т.н.) Виділити у відокремлений сектор вартість процесора.
17	Об'ємна кругова діаграма	Вартість комплектуючих у складі комп'ютера (процесор, материнська плата, оперативна пам'ять, жорсткий диск, корпус і т.н.)
18	Розрізна об'ємна діаграма	Вартість комплектуючих у складі комп'ютера (процесор, материнська плата, оперативна пам'ять, жорсткий диск, корпус і т.н.) Виділити у відокремлений сектор вартість процесора.
19	Об'ємна кругова діаграма	Структура сімейних витрат (продукти, одежа, ліки, послуги, інші витрати). Дані нанести поруч із діаграмою.
20	Об'ємна розрізна кругова діаграма	Структура сімейних витрат (продукти, одежа, ліки, послуги, інші витрати). Дані нанести поруч із діаграмою. Виділити у відокремлений сектор інші витрати.

Додаток Б. Вимоги до виконання роботи та оформлення пояснювальної записки

***Мета роботи*:** закріплення теоретичних знань, придбання умінь і навичок програмування складних застосувань із використанням взаємодії об'єктів, опанування засобами створення програм з графічним інтерфейсом користувача.

***Завдання*:** згідно із своїм варіантом, розробити програму, що за даними, які вводить користувач, будує діаграму заданого типу.

Програма повинна мати такі елементи:

1. Інформаційне вікно, що містить відомості про виконавця роботи: номер групи, прізвище, ім'я, по батькові студента, номер варіанту, назву графіку.

2. Вікно для введення параметрів графіку: кількість вхідних параметрів, список вхідних даних.

3. Головне вікно, яке містить збудовану діаграму.

Послідовність виконання завдання

Написання програми складається з послідовності операцій, додержання якої може полегшити виконання завдання.

1. Створити головне вікно. Розмістити в ньому інформаційні панелі, кнопки, меню тощо.

2. Створити клас, що описує дані, які необхідні для побудови діаграми – масив, список, або асоціативний масив (відображення) відповідних даних, у деяких варіантах – два, або три масиви. Конкретний тип для зберігання початкових даних обирається самостійно. Для усіх даних використати принцип інкапсуляції, тобто всі дані повинні бути приватними та до них повинні існувати методи доступу.

3. Створити панель, що буде призначена для побудови діаграми.

3.1. У створеній панелі зобразити осі координат, використовуючи граничні параметри панелі, отримати їх можна за допомогою методів getWidth(), getHeight() класу JPanel.

3.2. У класі панелі розмістити посилання на об'єкт, що містить дані діаграми/графіку.

3.3. Відповідно до даних, що містяться у цьому об'єкті, перевизначити метод paintComponent() так, щоб він зображував у панелі вказану діаграму.

3.4. Додати створену панель діаграми у головне вікно.

4. Створити діалогове вікно для введення початкових даних, які необхідні для побудови діаграми.

5. Зібрати всі класи у єдину програму, скомпілювати і налагодити програму.

6. Розробити пояснювальну записку.

ЗМІСТ