МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного додатку для побудови згрупованих рядів для задач метеорології та екології

Виконала студентка 5 курсу групи КН-5Т
Спеціальність 122 комп'ютерні науки
Руда Валентина Олегівна

Керівник асистент
Штефан Наталія Зінов'ївна

Консультант к.геогр.н., доцент
Бургаз Олексій Анатолійович

Рецензент д.т.н., професор
Мещеряков Володимир Іванович

Одеса 2020

ЗМІСТ

ВСТУП

Статистика – наука про закономірності масових явищ або подій. Об'єктом дослідження математичної статистики є статистична сукупність. а предметом вивчення – методи статистичної обробки. Статистична сукупність – всяке безліч подібних в деяких відносинах, але індивідуально розрізняються об'єктів. Сукупностями будуть, наприклад, результати спостережень за температурою повітря, рівнем загрезнения на будь-якої станції за певний період.

Основним методом математичної статистики є вибірковий метод. Сутність його полягає в тому, що суцільне обстеження масових однорідних об'єктів замінюється приватним, причому без істотних помилок у висновках.

Мета роботи: розробка нового програмного додатку розрахунку побудови згрупованих рядів для задач метеорології та клімотології а також створення оптимального інтерфейсу для користувача з урахуванням побажань ви-кладачів і студентів [1][1)].

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 50
- кількість рисунків – 25
- кількість таблиць – 0
- кількість посилань – 17

[1)] [1] Тема 7. Ряды распределения. URL: http://www.hi-edu.ru/e-books/xbook096/01/part-007.htm. (дата звернення 01.03.2020).

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ОДЕКУ – Одеський Державний Екологічний Університет

ПЗ – Програмне Забезпечення

API – application programming interface

AWT – Abstract Window Toolkit

IDE – Integrated Development Environment

JDK – Java Development Kit

JVM – Java Virtual Machine

PDF – Portable Document Format

PNG – Portable Network Graphics

UML – Unified Modeling Language

Медіана – значення випадкової величини, яке розділяє область існування цієї величини на дві частини, для яких виконується рівність

Мода – значення випадкової величини, що зустрічається частіше усього

Рефакторинг коду – переробка коду, рівносильне перетворенню алгоритмів

Activity Diagram – діаграма діяльності

Actor – актор

Ant/JUnit – середовище тестування

C++ – мова програмування

Closure – замикання у коді

Java – мова програмування

JetBrains IntelliJ IDEA – програмне забезпечення

Mockingbird − графічний онлайн сервіс

Pascal – мова програмування

Nashhorn JavaScript Engine – двигун JS

TabbedPane – набір розширених елементів управління Swing

Tree – набір розширених елементів управління Swing

Static Structure diagram – діаграма класів

Use Case diagram – діаграма варіантів використання

Swing – інструмент для розробки GUI для Java

SWT – бібліотека з відкритим кодом для розробки графічних інтерфейсів

JavaFX – платформа для створення додатків

JSF – веб фреймворк

1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Найважливішою частиною статистичного аналізу є побудова рядів розподілу (структурної угруповання) з метою виділення характерних властивостей і закономірностей досліджуваної сукупності. Залежно від того, яка буде ознака (кількісний або якісний) узятий за основу угруповання даних, розрізняють відповідно типи рядів розподілу.

Якщо за основу угруповання взято якісної ознаки, то такий ряд розподілу називають атрібутівним (розподіл за видами праці, по підлозі, по професії, за релігійною ознакою, національної приналежності і т.д.).

Якщо ряд розподілу побудований за кількісною ознакою, то такий ряд називають варіаціонним. Побудувати варіаційний ряд – значить упорядкувати кількісний розподіл одиниць сукупності за значеннями ознаки, а потім підрахувати числа одиниць сукупності з цими значеннями (побудувати групову таблицю).

Виділяють три форми варіаційного ряду: ранжируваних ряд, дискретний ряд і інтервальний ряд [1][1)].

1.2 Характеристика об’єкту розробки

Об’єкт розробки – це програмний додаток для студентів ОДЕКУ, який допоможе їм на практичних заняттях у розрахунку побудови згрупованих рядів для задач метеорології та клімотології.

Основні вимоги до проекту, які були відвинути замовником:

- реалізація алгоритму розрахунку даних (відповідно до теоретичного матеріалу замовника);

[1)] [1] Тема 7. Ряды распределения. URL: http://www.hi-edu.ru/e-books/xbook096/01/part-007.htm. (дата звернення 01.03.2020).

- розробка оптимального інтерфейсу користувача для полегшення вводу і отримання інформації;
- реалізація функції «створити звіт розрахунків» у текстовому форматі, а такой у форматі таблиці Exel для подальшого аналізу результатів;
- розробка додаткових функцій для паралельної обробки великих рядків даних з метою прискорення часу обчислення.

1.3 Аналіз існуючих аналогів

В ролі першого аналога – онлайн калькулятор «math.semestr»[2][1)] (рис. 1). За допомогою онлайн-калькулятора користувач має змогу:

- побудувати варіаційний ряд, побудувати гістограму та полігон;
- знайти показники варіації (середню, моду (в т.ч. і графічним способом), медіану, розмах варіації, децили, коефіцієнт варіації і інші показники).

Для угруповання ряду необхідно вибрати вид одержуваного варіаційного ряду (дискретний або інтервальний) і вказати кількість даних (кількість рядків). Отримане рішення зберігається в файлі Word.

Якщо з’являються питання щодо користування ресурсом, є можливість переглянути відео-інструкцію щодо коректного завдання вхідних даних і формування звіту по закінченню розрахунків.

На даному онлайн ресурсі для користувача представлено пошагову інструкцію користування (рис. 2). В цілому, даний калькулятор може розраховувати не великі ряди даних.

Крім цього, після отримання результатів є можливість побудувати графіки і зробити порівняння показників на ньому.

[1)] [2] Группировка данных и построение ряда распределения. URL: https://math.semestr.ru/group/group.php. (дата звернення 01.03.2020).

Рисунок 1 – Головна форма ресурсу «Math.semestr»

Рисунок 2 – Інструкція розрахунку на ресурсі«Math.semestr»

Після завершення розрахунків програма відображає результат, який можна завантажити у doc-файл як звіт.

До недоліків даного ресурсу слід віднести відсутність завантаження додатку до комп'ютера користувача, швидкість розрахунку залежить від інтернет доступу.

Другим аналогом розглянуто онлайн ресурс для математичних розрахунків [3][1)].

На даному онлайн ресурсі є можливість провести розрахунки (рис. 3).

Для початку роботи необхідно ввести в першу колонку дані першого виміру («До»), а в другу колонку дані другого виміру («Після»).

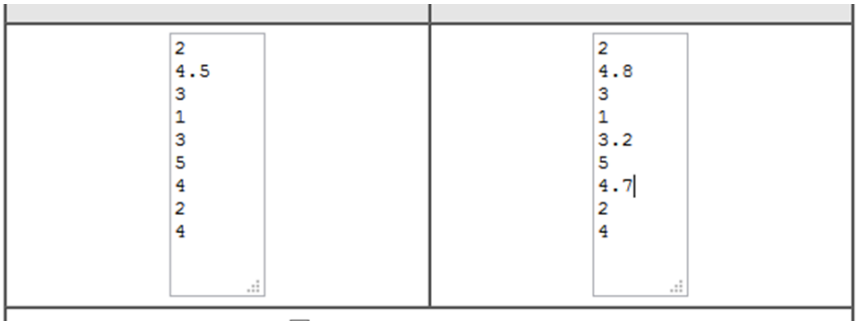


Рисунок 3 – Головна форма для завдання початкових даних

Дані вводяться по одному числу на рядок, без пробілів, пропусків і т.д. Вводяться тільки цифри. Дробові числа вводяться зі знаком «.», причому підказок для цього випадку не передбачено.

Після заповнення колонок слід натиснути на кнопку «Крок 2», щоб розрахувати ряд даних, або кнопку «Сброс» у разі помилкового введення вхідних даних для розрахунків. Дані також можна завантажити з текстового файлу чи файлу Excel-формату, але ця функція на даному ресурсі працює не коректно.

[1)] [3] Расчет сгруппированных рядов. URL: https://www.psychol-ok.ru/statistics/12. (дата звернення 03.03.2020).

Після введення даних для аналізу переходимо до другої форми калькулятора (рис. 4), на якій відображаються результати розрахунків у вигляду таблиці.

Градации температуры	m_i	НЧ	ВГ	p_i,%	$P=\Sigma p_i$,%
16,0-19,9	3	3	19,95	5	5
20,0-23,9	9	12	23,95	15	20
24,0-27,9	12	24	27,95	21	41
28,0-31,9	15	39	31,95	25	66
32,0-35,9	12	51	35,95	21	87
36,0-39,9	8	59	39,95	13	100

Рисунок 4 – Приклад результатів розрахунків

Крім цього, виводиться графік (рис. 5):

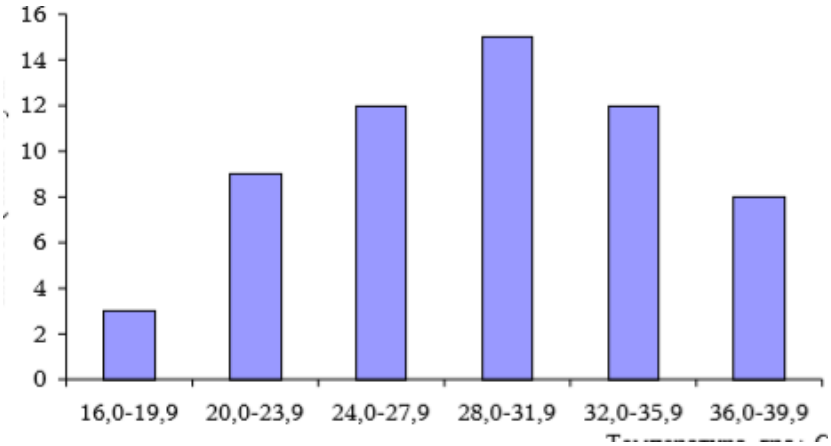


Рисунок 5 – Приклад результатів у вигляді графіків

Взагалі, даний ресурс має багато переваг:

- зручний інтерфейс для користувача;
- є підказки для коректного вводу початкових даних;
- відображення таблиці з результатами розрахунків;

- побудова графіку.

Недоліки:

- це он-лайн ресурс, а це означає пряму залежність від доступу до глобальної мережі;
- відсутня функція формування звіту з результатами;
- дані потрібно вводити тільки з клавіатури (функція воду даних з файлу не працює);
- можливість розрахунку тільки для невеликої послідовності даних;
- не зберігається історія розрахунків для подальшого аналізу.

В ролі останнього аналогу слід розглянути програмний додаток, яким користуються студенти у даний час. Це консольна програма, яка працює з маленькою кількістю даних.

Цей додаток написано мовою програмування Pascal. Програма має ряд недоліків, які будуть усунені в процесі реалізації дипломного проекту, а саме:

- консольний незручний інтерфейс для звичайного користувача. Вибір дії здійснюється натисканням цифри, яка припадає тієї або іншої дії;
- одночасно можна обробляти тільки один критерій (тобто немає можливості одну і ту ж пару обчислювати за двома критеріями паралельно);
- відсутність можливості паралельно обробляти кілька пар;
- ядро написано на Pascal, де всі змінні включаючи і ітератори оголошені глобально, що загальмовує доступ до цих змінним, а, отже, гальмує роботу програми;
- використовує базовий (навчальний) вид сортування, він більш тривалий.

За результатами аналізу аналогів поставлена мета для об’єкту розробки: створити програмний додаток для побудови групованих рядів для задач метеорології і кліматології, який ліквідує усі вищезазначені недоліки аналогів, а саме зручний інтерфейс користувача, зручне введення даних за допомогою

завантаження файлу, а збереження результатів у окремому файлі для подальшої обробки.

Крім цього, слід звернути увагу на оптимізацію програмного алгоритму для прискорення розрахунків, так як на практиці студенти будуть обчислюватись великий об’єм інформації.

Цей програмний додаток допоможе прискорити розрахунки і оптимізувати учбовий процес на практичних роботах студентів ОДЕКУ.

1.4 Вибір засобів розробки

1.4.1 Інтегрована середа розробки IntelliJ IDEA

Програмне забезпечення JetBrains IntelliJ IDEA – це провідна середовище швидкої розробки на мові Java. IntelliJ IDEA являє собою високотехнологічний комплекс тісно інтегрованих інструментів програмування, що включає:

інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації,

- потужні інструменти рефакторинга коду;
- вбудовану підтримку технологій J2EE;
- механізми інтеграції з середовищем тестування Ant/JUnit і системами управління версіями;
- унікальний інструмент оптимізації та перевірки коду Code Inspection;
- інноваційний візуальний конструктор графічних інтерфейсів.

Унікальні можливості JetBrains IntelliJ IDEA (рис. 6) позбавляють програміста від вантажу рутинної роботи, допомагають своєчасно усунути помилки і підвищити якість коду, піднімаючи продуктивність розробника на нову висоту [4][1)].

[1)] [4] JetBrains IntelliJ IDEA. URL: https://itpro.ua/product/jetbrains-intellij-idea/?tab=description. (дата звернення 11.03.2020).

Кожен компонент IntelliJ IDEA створений спеціально для того, щоб максимально підвищити продуктивність розробки. Розумний редактор коду в поєднанні з ергономічним дизайном роблять розробку не тільки ефективною, а й приємною.

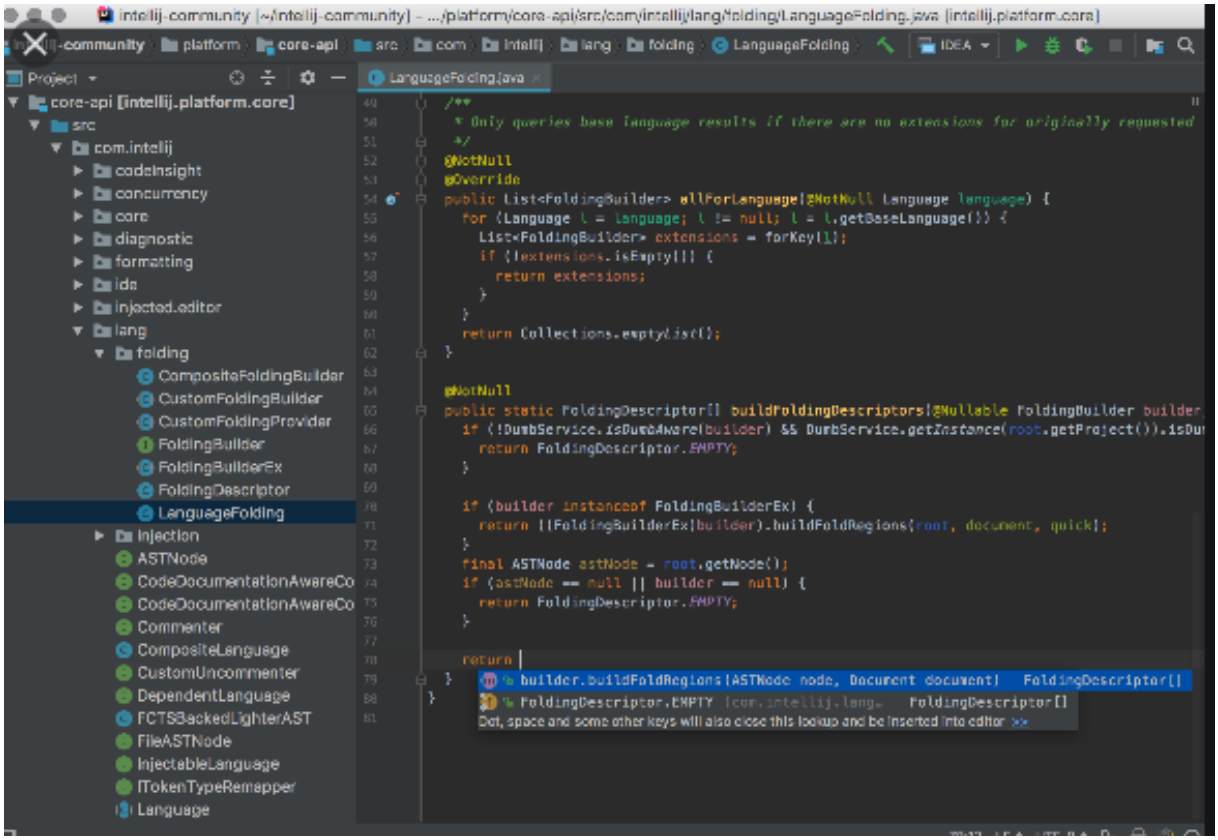


Рисунок 6 – Інтерфейс IntelliJ IDEA

Сьогодні про IntelliJ IDEA знає, без перебільшення, весь світ. Хоча платформа орієнтована, в першу чергу, на Java-кодинг, її універсальність дозволяє працювати з різними мовами.

Звичайно, тут немає такого масованого набору плагінів, як у будь-якого open-source конкурента, але зате в їх якості можна бути відносно переконаним [5][1)].

---

1) [5] IntelliJ IDEA – решение для топ-разработчиков. URL: https://webformyself.com/intellij-idea-reshenie-dlya-top-razrabotchikov/. (дата звернення 11.03.2020).

1.4.2 Мова програмування Java 8

Java – сильно типізована об’єктно-орієнтована мова програмування. Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якій комп'ютерній архітектурі, за допомогою віртуальної Java-машини. Байт-код виконується віртуальною машиною Java (JVM) – програмою, що обробляє байтовий код та передає інструкції обладнанню як інтерпретатор.

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи та устаткування, що дозволяє виконувати Java-програми на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною.

Лямбда-вирази і зміни в класах інтерфейсів роблять Java 8 новою мовою. Вони являють собою блоки коду, які може передавати по посиланню. Вони подібні до замикань (closure) в деяких інших мовах програмування: код, який реалізує функцію, опціонально приймає один або більше вхідних параметрів і опціонально повертає значення результату.

Лямбда-вирази в Java 8 – це фактично спеціалізація анонімних внутрішніх класів, з якими знайомий практично кожен Java-розробник. Анонімні внутрішні класи надають вбудовану реалізацію інтерфейсу або підклас базового класу, який ви хочете використовувати лише в одній точці свого програмного коду.

Вони використовуються таким же чином, але зі скороченим синтаксисом, що робить їх більш компактними, ніж стандартне визначення внутрішнього класу [6][1)].

[1)] [6] Изменения в языке Java 8. URL: https://www.ibm.com/developerworks/ru/library/j-java8lambdas/index.html. (дата звернення 15.03.2020).

Короткий опис удосконалень, реалізованих у версії Java 8:

- методи лямбда-виразів і віртуального розширення;
- найбільш примітна функція Java SE 8 – реалізація лямбда-виразів і допоміжних компонентів для мови програмування і платформи Java;
- API-інтерфейс дати і часу – цей новий API-інтерфейс дозволить розробникам використовувати більш природні і зрозумілі методи обробки дати і часу;
- механізм Nashhorn JavaScript Engine – нова полегшена високопродуктивна реалізація механізму JavaScript Engine інтегрована в JDK і доступна для додатків Java через існуючі API-інтерфейси;
- покращена безпека – підтримуваний вручну список методів, чутливих до духів програмами, замінений новим механізмом, який з високою точністю визначає ці методи і дозволяє стабільно виявляти їх викликають програми [7][1)].

1.4.3 Графічний інтерфейс SWING/AWT

Є кілька інструментів для розробки GUI для Java: найпопулярніші з них - Swing, SWT, JavaFX, JSF. Бібліотека Swing – це старий, але надійний крос-платформний інструмент, інтегрований в різні Java-IDE, в тому числі Eclipse.

Swing API – це набір розширюваних компонентів графічного інтерфейсу, який полегшує життя розробника для створення додатків на основі JAVA Front End/GUI.

Він побудований на основі API-інтерфейсу AWT і виступає в якості заміни API-інтерфейсу AWT, оскільки має майже всі елементи управління, відповідні елементам управління AWT.

[1)] [7] Информация о Java 8. URL: https://www.java.com/ru/download/faq/java8.xml. (дата звернення 16.03.2020).

Компонент Swing слід архітектурі Model-View-Controller для виконання наступних критеріїв:

- одного API має бути досить для підтримки множинного зовнішнього вигляду;
- API повинен бути орієнтований на модель, щоб не було потрібно, щоб у API найвищого рівня були дані;
- API полягає в використанні моделі java bean, щоб інструменти builder і IDE могли надавати розробникам більш якісні послуги для використання;
- одного API має бути досить для підтримки множинного зовнішнього вигляду;
- API повинен бути орієнтований на модель, щоб не було потрібно, щоб у API найвищого рівня були дані.

До особливостей Swing можна віднести наступне:

1. Легка вага – компоненти Swing не залежить від API власної операційної системи, так як елементи управління Swing API відображаються в основному з використанням чистого коду JAVA, а не викликів базової операційної системи.
2. Багаті елементи управління – Swing надає багатий набір розширених елементів управління, таких як Tree, TabbedPane, слайдер, палітра кольорів і елементи управління таблицями.
3. Широкі можливості налаштування – елементи управління Swing можна легко налаштувати, оскільки зовнішній вигляд не залежить від внутрішнього уявлення.
4. Змінний зовнішній вигляд – графічний інтерфейс на основі SWING Зовнішній вигляд програми може змінюватися під час виконання в залежності від доступних значень [8][1)].

[1)] [8] SWING – Краткое руководство. URL: https://coderlessons.com/tutorials/java-tekhnologii/nauchitsia-kachatsia/swing-kratkoe-rukovodstvo. (дата звернення 16.03.2020).

1.4.4 Онлайн конструктор Mockingbird

У кожного програмного додатку свій інтерфейс, який є мостом взаємодії між користувачем і машинним кодом. Чим краще інтерфейс, тим ефективніше взаємодія. Однак далеко не всі розробники і навіть дизайнери, замислюються про створення зручного і зрозумілого графічного інтерфейсу користувача.

Mockingbird − це онлайн сервіс, який дозволяє користувачу будувати прототипи додатків, щоб показати найважливіше: ідею, інформацію, взаємодії (рис. 7). До можливостей Mockingbird можна віднести наступне:

- створення прототипів для користувача інтерфейсів за короткий час з функцією Drag and drop – як результат користувач отримує макет майбутнього інтерфейс ;
- можна додавати декілька сторінок до макету;
- перехід між сторінками використовуючи посилання;
- редагування макета співробітниками або клієнтами в режимі реального часу;
- експорт в форматах PNG і PDF.

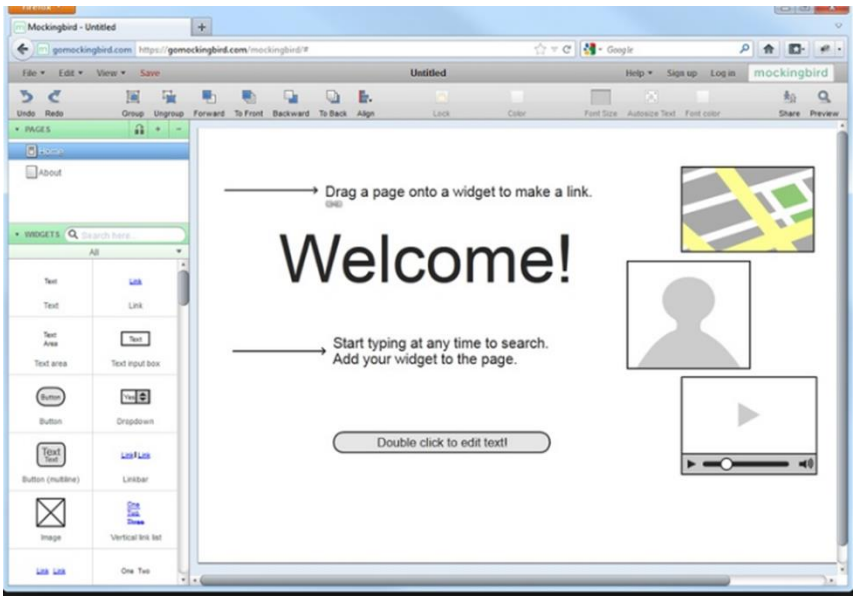


Рисунок 7 – Інтерфейс онлайн конструктора Mockingbird

При проектуванні інтерфейсу слід пам’ятати деякі вимоги:

- розробка інтерфейсу зазвичай починається з визначення завдання або набору задач, для яких продукт призначений;
- просте має залишатися простим, тобто не слід ускладнювати інтерфейси;
- його користувачі не замислюються над тим, як влаштована програма. Все, що вони бачать – це інтерфейс. Тому, з точки зору споживача саме інтерфейс є кінцевим продуктом;
- інтерфейс повинен бути орієнтованим на людину, тобто відповідати потребам людини і враховувати його слабкості. Потрібно постійно думати про те, з якими труднощами може зіткнутися користувач;
- слід пам’ятати про поведінку і звички користувачів [9][1)].

[1)] [9] Проектирование графического интерфейса пользователя. URL: https://habr.com/ru/post/208966/. (дата звернення 22.03.2020).

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1 Статистична оцінка параметрів розподілу випадкових величин по згрупованому ряду

2.1.1 Побудова та графічне представлення згрупованого ряду

Статистичний ряд (вибірка) – обмежена кількість значень випадкової величини, добута випадковим чином із генеральної сукупності. Тому статистичні ряди називають вибірками із генеральної сукупності. Вибірки випадкові та число їх безмежне. Важливою ознакою ряду є його об'єм, тобто кількість членів, що складають сукупність.

Вибірки характеризуються статистичними оцінками параметрів. Значення параметра генеральної сукупності, яке розраховують на основі вибірки, називають статистичною оцінкою ($\hat{\theta}$) цього параметра θ і позначають символом "^". Первинною формою зображення екологічної інформації є простий статистичний ряд, значення якого розташовуються в хронологічній послідовності [10][1)].

Вихідні дані подаються у вигляді простого статистичного ряду головним чином у тих випадках, коли задача дослідження полягає у вивченні особливостей їх часової мінливості. Якщо така задача не ставиться, то ряди випадкових величин можуть зображатися у більш компактній формі – у вигляді згрупованого ряду.

Побудова згрупованого ряду на основі простого статистичного ряду проводиться наступним чином:

1) визначають область значень величини X $[x_{min}; x_{max}]$, де x_{min} – мінімальне, x_{max} – максимальне значення із вихідного ряду даних;

[1)] [10] Ряды распределения и их построение/. URL: http://www.hi-edu.ru/e-books/xbook096/01/part-008.htm. (дата звернення 24.03.2020).

2) всі члени вихідного ряду ранжуються – розташовуються в напрямку їх збільшення (або зменшення);
3) знаходять k – кількість часткових інтервалів (градацій), на які треба поділити область значень. Для цього використовується формула:

$$k = 5 \lg n \ , \qquad (1)$$

де n – об'єм ряду.

У випадку коли k не є цілим числом, його округлюють до цілого.

4) знаходиться довжина часткового інтервалу **c** за формулою:

$$c = \frac{x_{max} - x_{min}}{k} \ ; \qquad (2)$$

5) визначається значення випадкової величини X на межах часткових інтервалів. Для i-того часткового інтервалу, значення величини X на лівій межі є $\left[x_{min} + (i-1)c\right]$, а на правій – $\left[x_{min} + ic\right]$. Можна позначати ліву межу часткового інтервалу як x_{i-1}, а праву – $x_{i+1}\left(i = \overline{1,k}\right)$.

Кінець попередньої і початок наступної градацій будуть повторюватися. Тому треба визначити закриті й відкриті межі градацій, тобто встановити, яку з величин враховувати в даній градації, щоб виключити повторення одних і тих же значень випадкової величини X, які дорівнюють значенню на межі градації;

6) підраховують кількість членів ряду, що потрапляють до кожного i – того часткового інтервалу – $m_i \left(i = 1, 2, \ldots, k \right)$. Величини m_i називають інтервальними емпіричними частотами. Сума частот по всіх часткових інтервалах дорівнює об'єму вибірки X;

7) розраховують інтервальні частості p_i (відносні інтервальні частоти) за формулою:

$$p_i = \frac{m_i}{n}. \tag{3}$$

8) знаходять $\tilde{x}_i$ – значення випадкової величини X на середині кожного часткового інтервалу за формулою:

$$\tilde{x}_i = \frac{x_{i-1} + x_{i+1}}{2} \quad \left(i = \overline{1,k}\right). \tag{4}$$

Згрупованим називають такий ранжований статистичний ряд, який представляють сукупністю значень випадкової величини X на серединах часткових інтервалів $\tilde{x}_i$ і відповідних інтервальних частот m_i, або частостей p_i:

$$X: \begin{cases} \tilde{x}_1, \tilde{x}_2, \tilde{x}_3, ..., \tilde{x}_i, ..., \tilde{x}_k \\ m_1, m_2, m_3, ..., m_i, ..., m_k \end{cases} \left(i = \overline{1,k}\right). \tag{5}$$

Згрупований ряд, може зображатися за допомогою діаграм: гістограми чи полігону.

Гістограма – це система прямокутників, основи яких дорівнюють довжині часткового інтервалу, а висоти – відповідним інтервальним частотам (або частостям).

Якщо точки з координатами $\left(\tilde{x}_i; m_i\right)$ або $\left(\tilde{x}_i; \hat{p}_i\right)$ з'єднати відрізками прямої, то отриману таким чином діаграму називають полігоном [10][1)].

[1)] [10] Ряды распределения и их построение/. URL: http://www.hi-edu.ru/e-books/xbook096/01/part-008.htm. (дата звернення 24.03.2020).

2.1.2 Розрахунок статистичних оцінок параметрів розподілу

Основні властивості випадкових величин характеризуються початковими (ν), центральними (μ) та основними (r) моментами розподілу різних порядків.

Оцінка першого початкового моменту розподілу ($\hat{\nu}_1$) є оцінкою математичного сподівання ($\hat{m}_x$) і дорівнює середньому значенню ($\overline{x}$) випадкової величини X:

$$\hat{\nu}_1 = \hat{m}_x = \overline{x} = \frac{1}{n}\sum_{i=1}^{k} \tilde{x}_i m_i \qquad (6)$$

Центральні моменти розподілу оцінюються, починаючи з другого моменту (l=2), тому, що перший центральний момент завжди дорівнює нулю, як і його оцінка. Центральний момент другого порядку має сенс дисперсії випадкової величини: $\mu_2 = \sigma_x^2$.

Статистичної оцінка центрального моменту розподілу другого порядку на основі згрупованого ряду розраховується за формулою:

$$\hat{\mu}_2 = \hat{\sigma}_x^2 = \frac{1}{n}\sum_{i=1}^{k} \left(\tilde{x}_i - \overline{x}\right)^2 m_i \,, \qquad (7)$$

Статистична оцінка другого центрального моменту розподілу, що розраховується за формулою (7), є зсуненою оцінкою дисперсії [11][1)].

[1)] [11] Статистичні критерії перевірки основної гіпотези. URL: https://studopedia.com.ua/1_47878_statistichniy-kriteriy-perevirki-osnovnoi-gipotezi.html. (дата звернення 25.03.2019).

Незсунену, ефективну та умотивовану оцінку дисперсії випадкової величини X позначають S_x^2 і розраховують за формулою (8):

$$S_x^2 = \frac{1}{n-1}\sum_{i=1}^{k}(\tilde{x}_i - \bar{x})^2 m_i \,. \qquad (2.8)$$

Статистична оцінка середнього квадратичного відхилу цієї величини розраховується за формулою:

$$S_x = \sqrt{S_x^2} \,. \qquad (9)$$

Оцінка третього основного моменту характеризує асиметрію кривої розподілу інтервальних частостей (або частот) і називається коефіцієнтом асиметрії: $\hat{r}_3 = As$.

$$As = \hat{r}_3 = \frac{1}{n}\sum_{i=1}^{k}\frac{(\tilde{x}_i - \bar{x})^3 m_i}{S_x^3} \qquad (10)$$

Крива розподілу має правосторонню асиметрію за умови $As > 0$, і лівосторонню – за умови $As < 0$. Вона є симетричною відносно центру розподілу, якщо $As = 0$ (рис. 8).

Крім асиметрії, крива розподілу, порівняно з кривою нормального розподілу, може бути витягнутою або сплюснутою (рис. 9). Мірою цього є коефіцієнт ексцесу E:

$$E = r_4 - 3 = \left(\frac{1}{n}\sum_{i=1}^{k}\frac{(\tilde{x}_i - \bar{x})^4 m_i}{S_x^4}\right) - 3. \qquad (11)$$

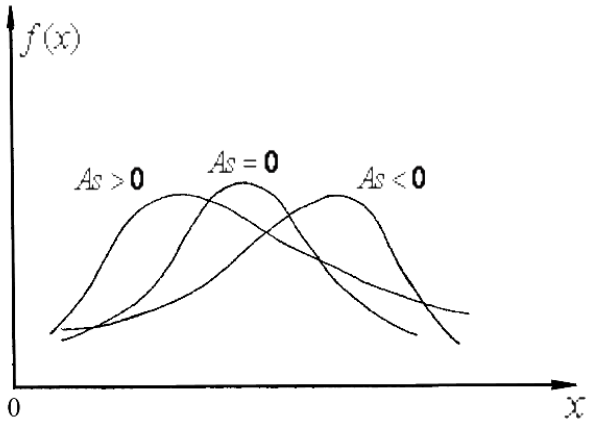


Рисунок 8 – Форми кривих розподілу при різних значеннях коефіцієнту асиметрії

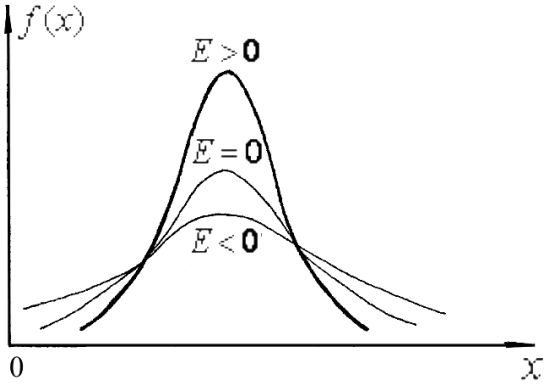


Рисунок 9 – Форми кривих розподілу при різних значеннях коефіцієнту ексцесу

Інколи при статистичних дослідженнях рядів екологічної інформації необхідно визначити модальне значення Mo та медіану Me.

Мода – значення випадкової величини, що зустрічається частіше усього, тобто має максимальну ймовірність для дискретної величини або максимум функції щільності ймовірності в даній точці для безперервної випадкової величини [12][1)].

[1)] [12] Лоєва І. Д., Бургаз О.А. Збірник методичних вказівок до практичних занять з дисципліни «Обробка і аналіз інформації» для студентів денної форми навчання, напрям підготовки 6.040106 «Екологія, охорона навколишнього середовища та збалансоване природокористування». Одеса: ОДЕКУ, 2010. 65 с.

Якщо ряд є згрупованим та ранжування проводилося в бік зростання значень випадкової величини, то моду визначають за формулою:

$$M_o = x_0 + c\frac{\left(m_i - m_{i-1}\right)}{\left(2m_i - m_{i-1} - m_{i+1}\right)}, \tag{12}$$

де x_0, c, m_i – відповідно початок, довжина та емпірична частота модального інтервалу;

m_{i-1}, m_{i+1} – частоти попереднього і наступного за модальним часткових інтервалів.

Медіана – значення випадкової величини, яке розділяє область існування цієї величини на дві частини, для яких виконується рівність. Тобто ймовірність того, що значення випадкової величини X більше медіани, дорівнює ймовірності того, що значення випадкової величини X менше медіани [12][1)].

Медіану розраховують за формулою:

$$Me = x_e + \frac{c\left(\frac{n}{2} - m^*\right)}{m_e}, \tag{13}$$

де n – об’єм вибірки;

m^* – накопичена частота до медіанного інтервалу;

x_e, c, m_e – відповідно початок, довжина та частота медіанного інтервалу.

[1)] [12] Лоєва І. Д., Бургаз О.А. Збірник методичних вказівок до практичних занять з дисципліни «Обробка і аналіз інформації» для студентів денної форми навчання, напрям підготовки 6.040106 «Екологія, охорона навколишнього середовища та збалансоване природокористування». Одеса: ОДЕКУ, 2010. 65 с.

2.2 Моделювання UML-засобами

В ході проектування програмного додатку створюється проектна документація, що включає текстові описи, діаграми, моделі майбутньої програми. У таких випадках зручно використовувати мову UML.

UML – є графічною мовою для візуалізації, опису параметрів, конструювання та документування різних систем (програм зокрема). Діаграми створюються за допомогою спеціальних CASE-засобів, наприклад Rational Rose і Enterprise Architect.

На основі технології UML будується єдина інформаційна модель. Наведені вище CASE засоби здатні генерувати код на різних об'єктно-орієнтованих мовах, а так само мають дуже корисною функцією реверсивного інжинірингу [13][1)].

2.2.1 Діаграма варіантів використання

Процес моделювання в UML – це поетапний спуск від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи.

Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування [14][2)].

Проектована система представляється у вигляді безлічі сутностей, що взаємодіють з системою за допомогою варіантів використання. При цьому

[1)] [13] Проектирование графического интерфейса пользователя. URL: https://habr.com/ru/post/208966/. (дата звернення 25.03.2020).

[2)] [14] Диаграмма вариантов использования. URL: https://www.uml-diagrams.org/use-case-diagrams.html. (дата звернення 25.05.2020).

актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні.

Варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором.

У найзагальнішому випадку, діаграма варіантів використання − граф спеціального виду, який є графічною нотацією для представлення конкретних варіантів використання, акторів, можливо деяких інтерфейсів, і відносин між цими елементами [15][1)].

Слід зазначити, що відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до моделюємої системи.

Приклад діаграми варіантів використання представлено на рисунку 10:

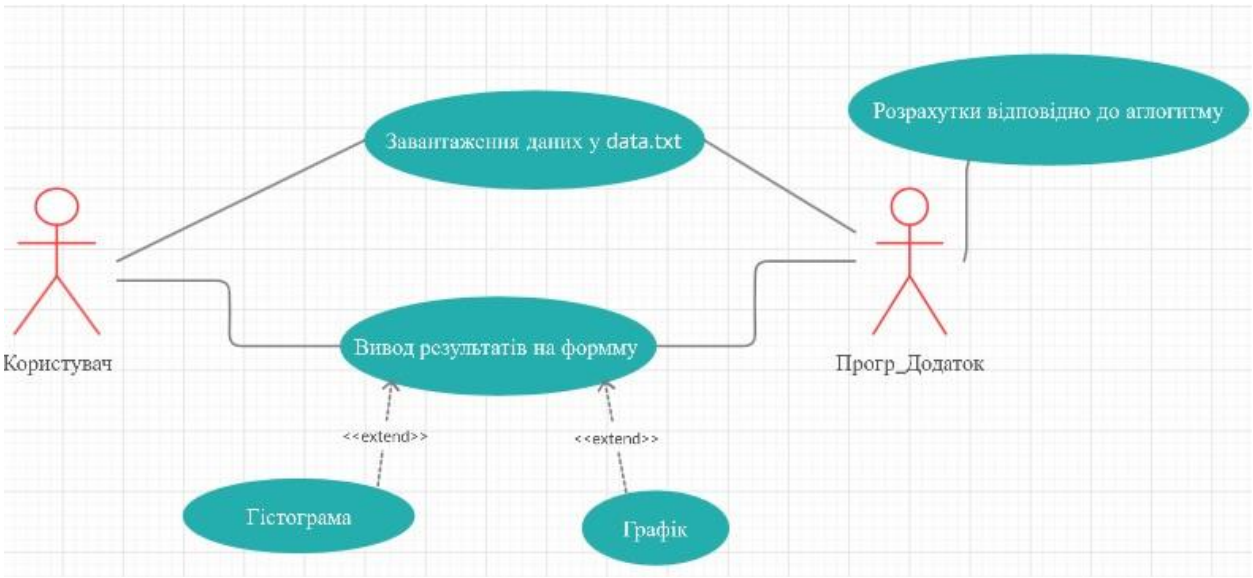


Рисунок 10 − Діаграма варіантів використання для об’єкту розробки

[1)] [15] UML 2 Use Case Diagrams. URL: www.agilemodeling.com/artifacts/useCaseDiagram.htm. (дата звернення 28.03.2020).

2.2.2 Діаграма діяльності

Діаграма діяльності або діаграма активності (англ. Activity Diagram) – UML-діаграма, на якій показані дії, стану яких описано на діаграмі станів. Під діяльністю (англ. Activity) розуміється специфікація виконуваної поведінки у вигляді координованого послідовного та паралельного виконання підлеглих елементів – вкладених видів діяльності та окремих дій (англ. Action), з'єднаних між собою потоками, що йдуть від виходів одного вузла до входів іншого.

Діаграма виглядає найбільш простою, оскільки нагадує звичну всім блок-схему. Насправді ж діаграма активності – це щось більше, ніж блок-схема, хоча цілі у них схожі – обидві вони відображають певний алгоритм.

Нотація UML пропонує п'ять видів системи:

– вид системи з точки зору прецедентів;
– вид з точки зору проектування;
– вид з точки зору процесів;
– вид з точки зору розгортання;
– вид з точки зору реалізації.

При цьому кожен з перерахованих способів подання системи може містити послідовності дій, які можуть бути описані за допомогою алгоритмів. Ось тут, так би мовити, і виходять на сцену діаграми діяльностей.

Взагалі кажучи, будь-який елемент моделі, що має динамічну поведінку, може бути доповнений діаграмою діяльності – саме для уточнення цієї самої динаміки. Ось вже і є безпосередньо динаміка [16][1)].

Саме на діаграмі діяльності представлені переходи потоку управління від однієї діяльності до іншої. Діаграма діяльності може бути приєднана до будь-якого елементу моделі, що має динамічну поведінку. До речі, логічніше говорити не «діаграма діяльності», а «діаграма діяльностей» – у множині.

[1)] [16] Диаграмма активностей (Activity diagram). URL: https://flexberry.github.io/ru/fd_activity-diagram.html. (дата звернення 02.04.2020).

Діаграми діяльності використовуються при моделюванні бізнес-процесів, технологічних процесів, послідовних та паралельних обчислень. Діаграми діяльності складаються з обмеженої кількості фігур, з'єднаних стрілками.

Для дипломного проекту було розроблено діаграму (рис. 11), яка відображає процес побудови групованого ряду:

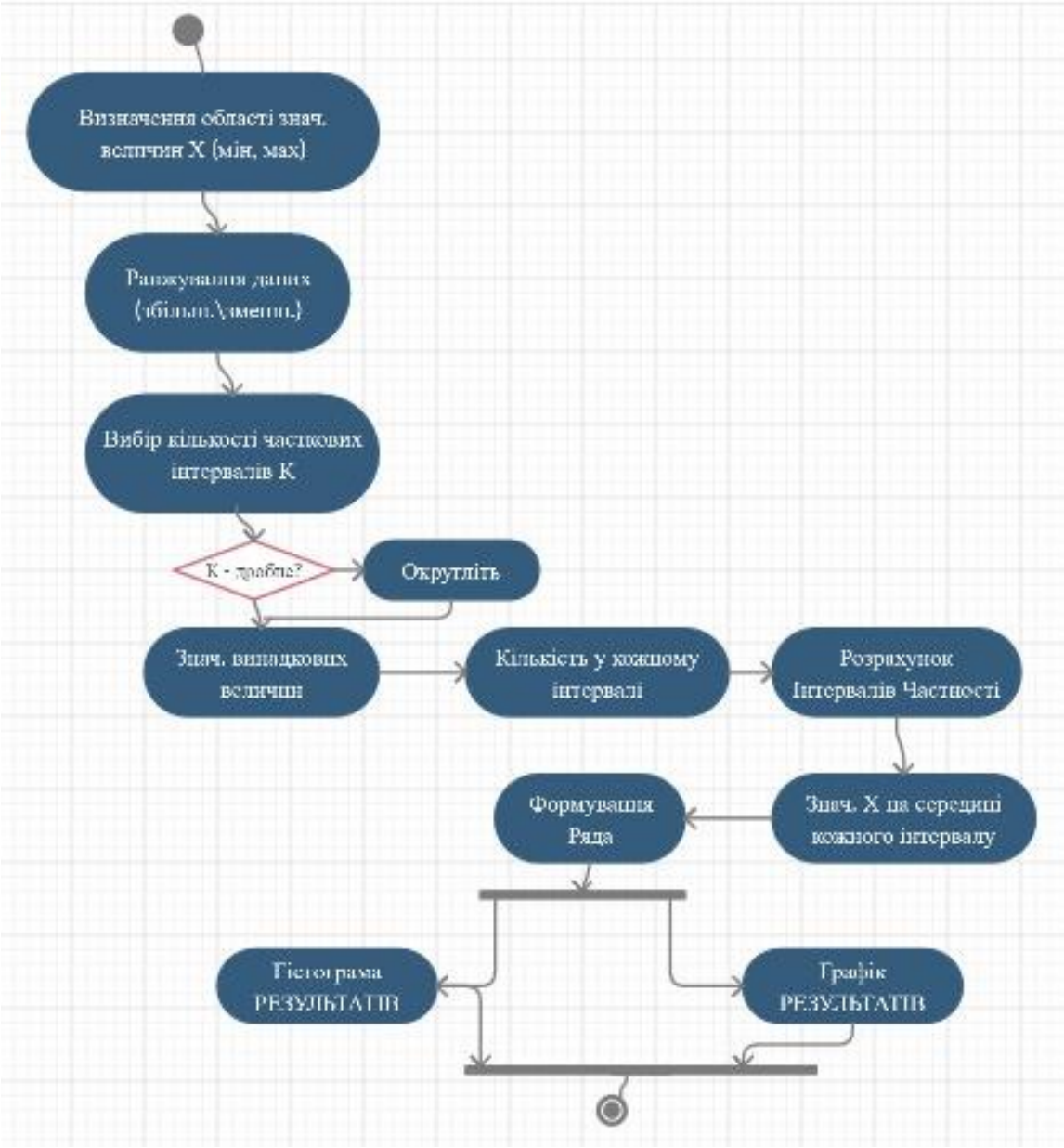


Рисунок 11 – Діаграма діяльності для алгоритму побудови групованого ряду

2.2.3 Діаграма класів

Діаграма класів (англ. Static Structure diagram) – діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв'язку між ними. Являється однією з основних UML-діаграм. Це набір статичних, декларативних елементів моделі.

Клас – це основний будівельний блок. Між класами UML і програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів або для виконання реінжинірингу. Кожен клас має назву, атрибути і операції.

Клас на діаграмі показується у вигляді прямокутника, розділеного на 3 області. У верхній міститься назва класу, в середній – опис атрибутів (властивостей), в нижній – назви операцій, що надаються об'єктами цього класу.

Діаграми класів можуть застосовуватися й при прямому проектуванні, тобто в процесі розробки нової системи, та при зворотному проектуванні – описі існуючих та використовуваних систем.

Інформація з діаграми класів безпосередньо відображається в вихідному коді програми – в більшості існуючих інструментів UML-моделювання можлива кодогенерація для певної мови програмування (зазвичай Java або C++).

Таким чином, діаграма класів – кінцевий результат проектування та відправна точка процесу розробки [17][1)]. Діаграма класів займає центральне місце в проектуванні об'єктно-орієнтованої системи.

Нотація класів використовується на різних етапах проектування та будується з різним ступенем деталізації. Мова UML застосовується не тільки для проектування, але і з метою документування, а також створення ескізів проекту.

[1)] [17] Отношения классов – от UML к коду. URL: https://habr.com/ru/post/150041/. (дата звернення 06.04.2020).

У будь-якому об'єктно-орієнтованому процесі проектування діаграма класів є результатом, тому що вона є моделлю, найбільш близькою до реалізації, тобто коду. Існують інструменти, що здатні перетворити діаграму класів в код – такий процес називається кодогенерацією та підтримується безліччю IDE та засобів проектування.

Діаграму класів представлено на рисунку 12:

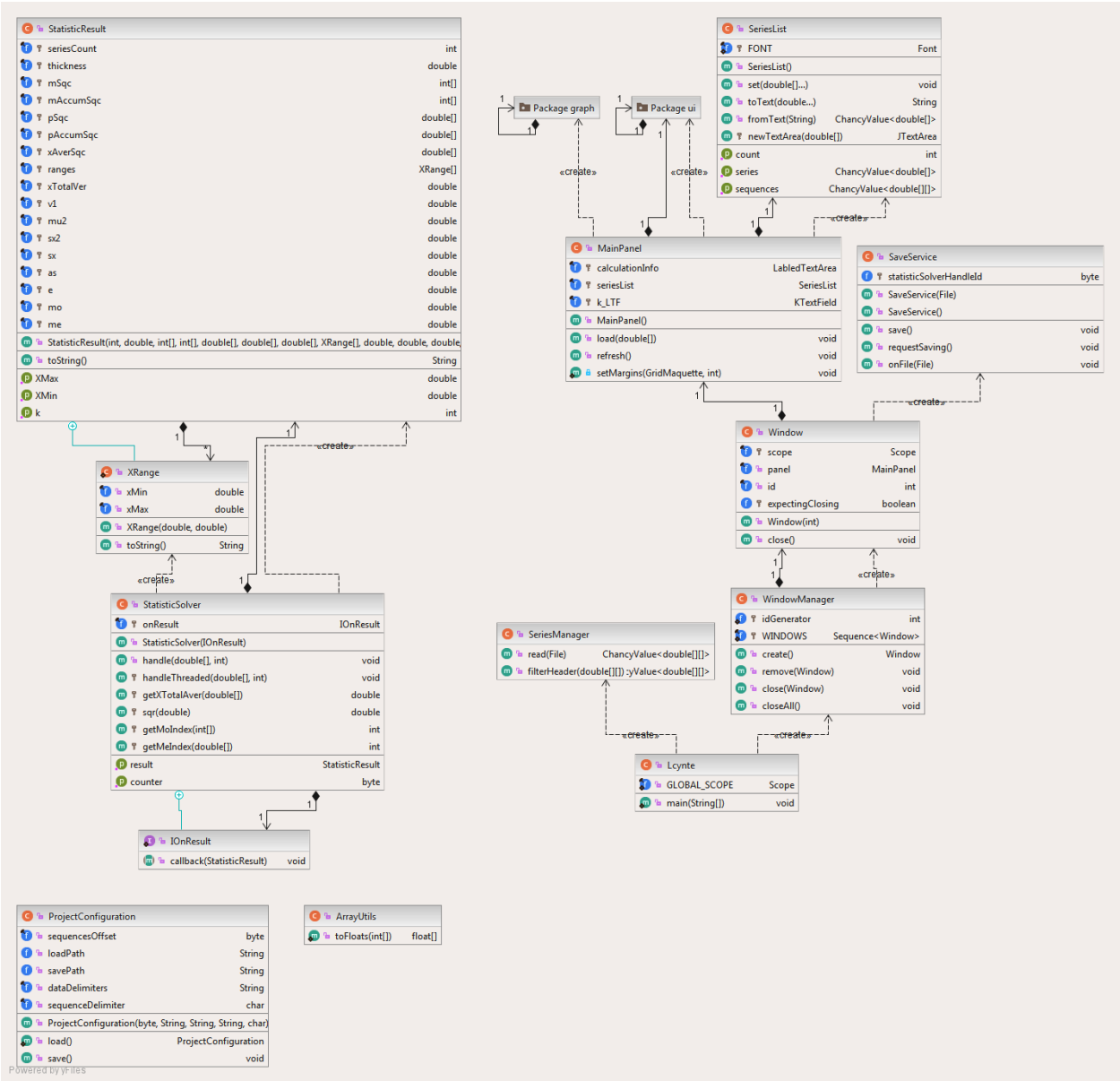


Рисунок 12 – Діаграма класів об’єкту розробки

2.3 Проектування макету інтерфейсу користувача

Під час проектування інтерфейсу користувача були враховані вимоги замовника:

- не перевантажувати зір користувача яркими кольоровими рішеннями;
- інтерфейс повинен бути інтуїтивно зрозумілим. Таким, щоб користувач розумів як ним користуватися.

Розробка інтерфейсу зазвичай починається з визначення завдання або набору задач, для яких продукт призначений. Користувач не повинен замислюватись щодо логіки роботи програмного долатку, тобто, з точки зору споживача саме інтерфейс є кінцевим продуктом.

Для створення макету інтерфейса програми використовувався он-лайн конструктор Mockingbird (рис.13).

Рисунок 13 – Макет інтерфейсу програмного додатку

3 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

3.1 Основні елементи інтерфейсу

Огляд програми почнемо з його зовнішнього вигляду і взаємодії на рівні користувача, а потім розглянемо внутрішні аспекти роботи програми. Отже, спочатку запускаючи додаток, воно має такий вигляд (рис. 14):

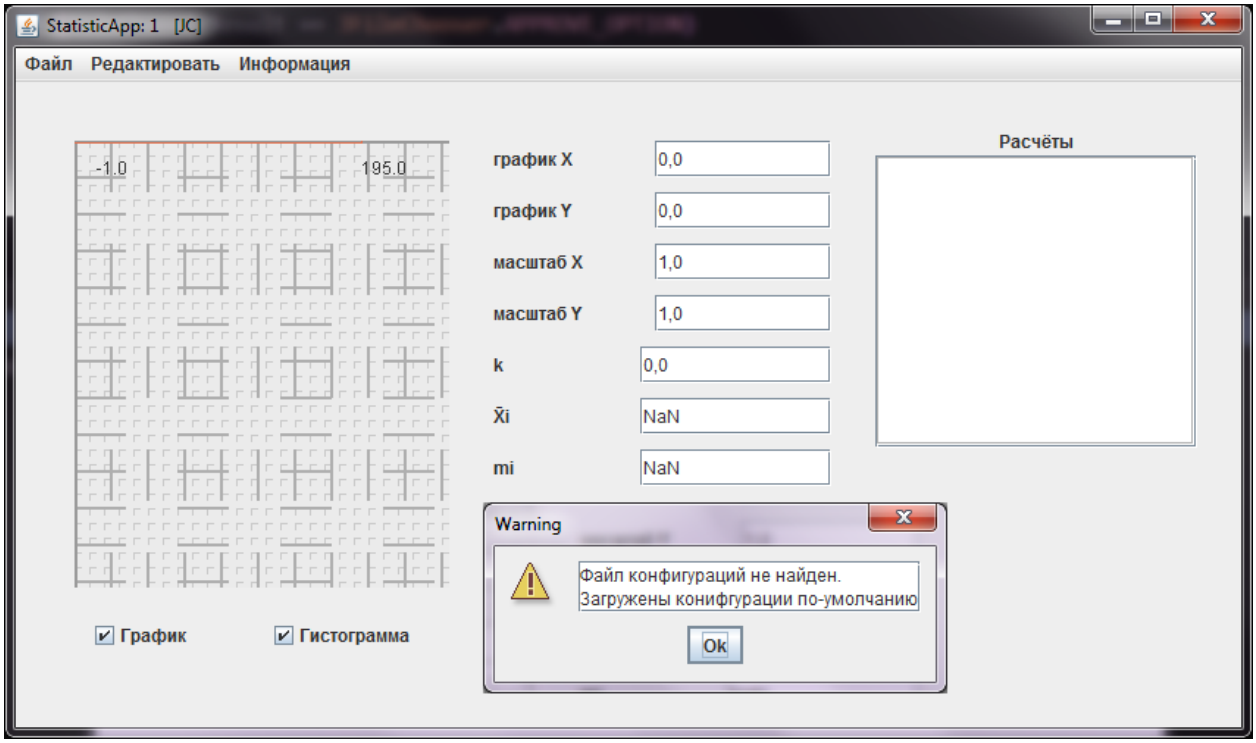


Рисунок 14 – Головне вікно додатку

Як можна помітити, у нас з'явився Alert, який повідомляє про те, що поруч з додатком відсутні конфігурації. В такому випадку будуть створені конфігурації за замовчуванням. У цей набір налаштувань входять шлях завантаження даних, шлях збережень результатів, роздільники при читанні даних з файлів та інше.

У самому головному вікні ми бачимо Декартову площину. Тут буде відображена гістограма і графік по ній. Під площею розміщені два чек-боксу, для включення/відключення графіка і гістограми відповідно.

Праворуч розміщені координати площині, її масштаб, параметр k, який буде враховуватися в обрахунку статистики. За замовчуванням його значення дорівнює 5*lg(n), але можна поставити і своє. Це додає гнучкості.

Наступні два значення будуть відображати по активної секції гістограми її параметри. У самій правій частині будуть відображені розрахунки нашого застосування.

Наступним кроком ми завантажимо файл з рядом (рис. 15):

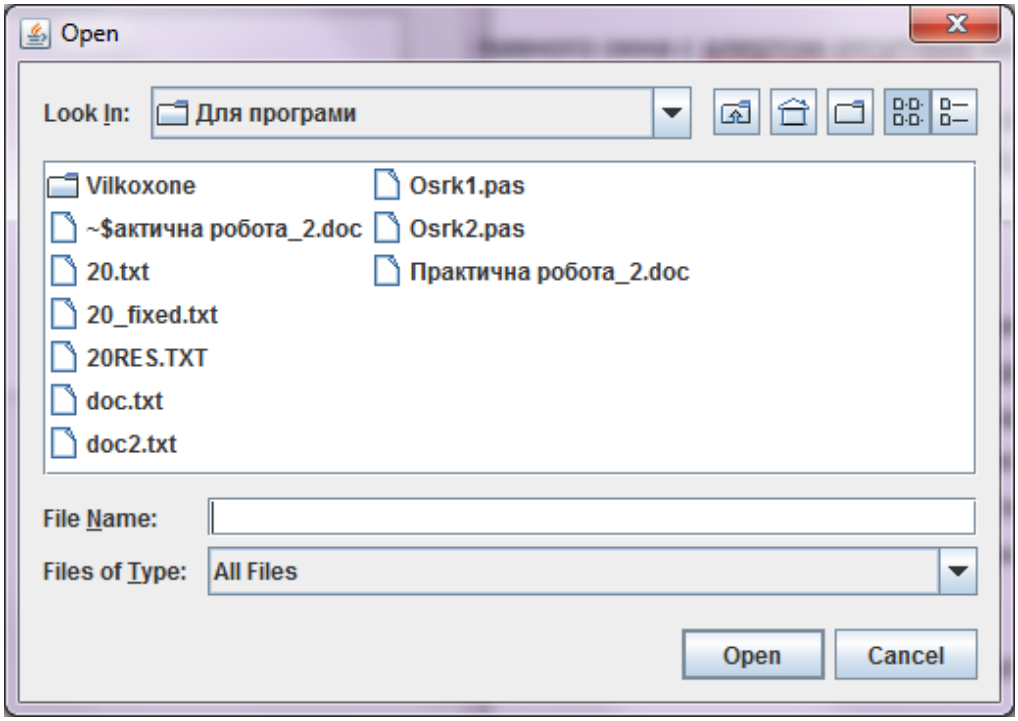


Рисунок 15 – Завантаження файлу з даними

Під час завантаження вискакує віконце, в якому запитується читати з хедером або без нього (рис. 16). По суті це підтримка старого формату зі старою версією проекту, де спочатку файлу вказувався ряд, значення якого ідентифікували розмірності рядів для обрахунку.

У новому форматі такої хедер не потрібен. Після обробки на вікні відображаються результати роботи.

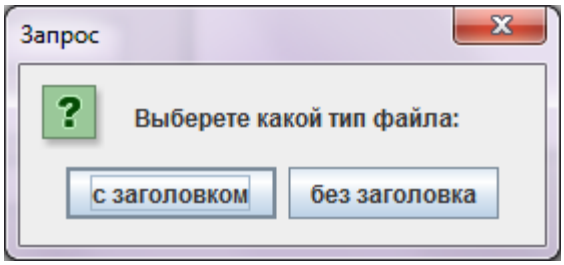


Рисунок 16 – Запит на режим зчитування файлу

Перше на що звертаємо увагу, це що графік дуже маленький. Для переміщення по графіку можна використовувати стрілочки, чи поєднання стрілок і ctrl (це збільшить крок переміщення).

«Ctrl +» – відцентруйте графік, «Ctrl + B» відцентруйте і скине масштаб. Щоб збільшити або зменшити можна користуватися комбінаціями «ctrl + '+'» або «ctrl + '-'», а так само можна вручну змінювати масштаб у відповідних місцях. Так і вчинимо – поставимо розмірність по x рівну 700, а по y 15 (рис. 17).

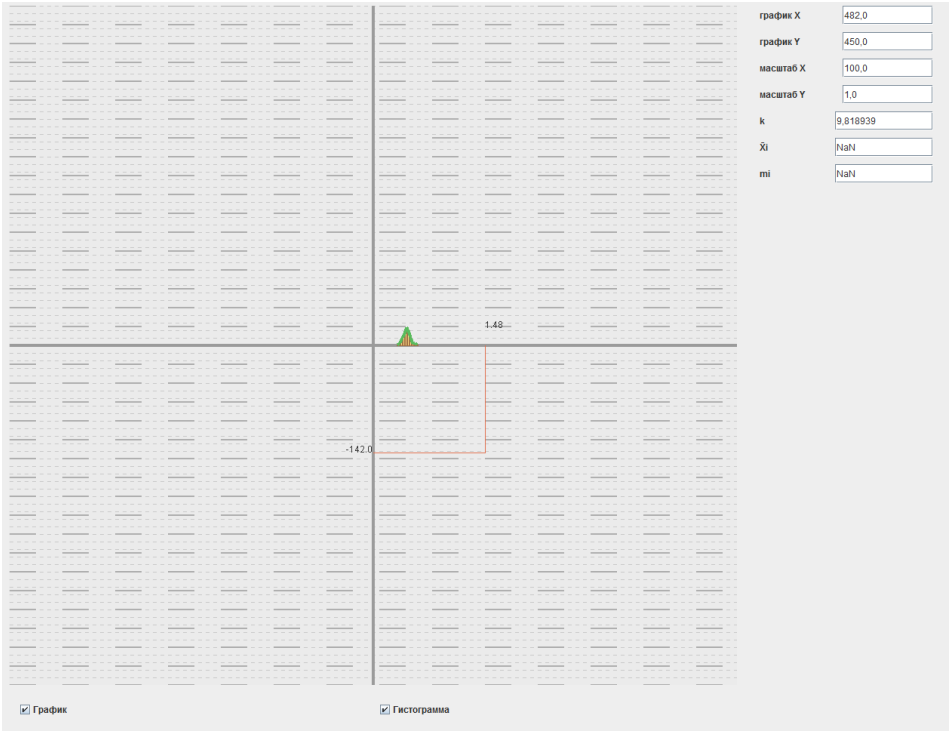


Рисунок 17 – Масштабування графіку

В результаті вивод гістограми буде мати наступний вигляд (рис. 18):

Рисунок 18 – Приклад відображення графіку

Рухаючи курсор миші можна помітити червоні виноски – вони відображають поточний стан курсору в координатної осі, це дозволяє простіше орієнтуватися.

Наводячи мишей на гістограму можна помітити, що стовпець підсвічується, а два останніх текстових поля праворуч відображають дані стовпця.

Клацнувши по чек-боксів внизу графіка видно, як відбувається візуалізація графіка і гістограми (рис. 19):

Рисунок 19 – Приклад візуалізації графіку та гістограми

У верхній правій частині головного вікна є панель, на якій вісвічуються координати та інші показчики (рис. 20):

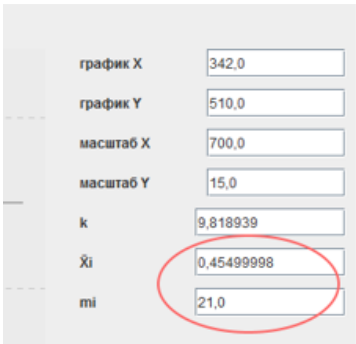


Рисунок 20 – Показники графіку

Для зручності користувача у програмному додатку є функція вибору типу відображення даних: графік (рис. 21) чи гістограма (рис. 22):

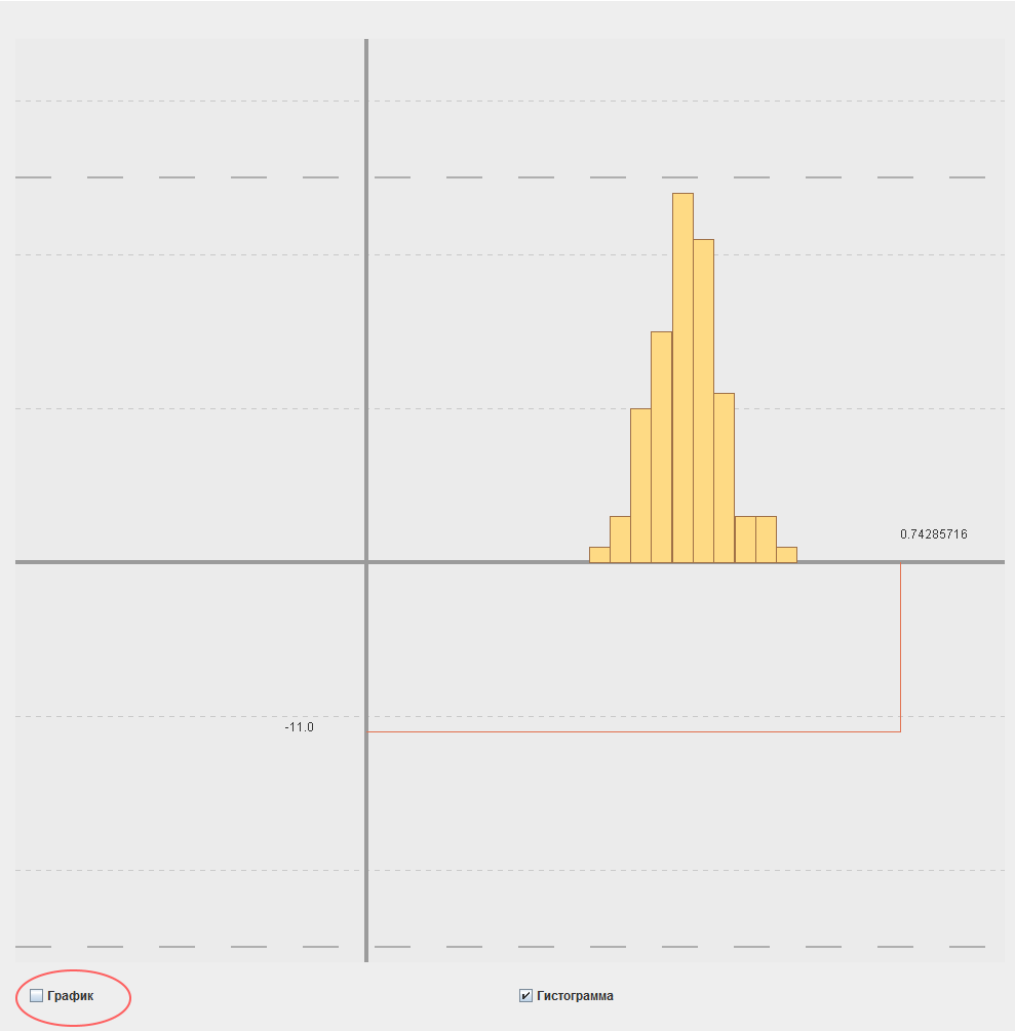


Рисунок 21 – Приклад гісторгами

У самій правій частині програми відображені результати роботи програми. Тут виводиться інформація про загальну кількість елементів ряду, про розмір стовпчика гістограми, середні значення стовпців, його межі, мода, медіана і інші параметри.

Внизу відображено поле з самим поруч. Додаток дозволяє всередині нього модифікувати цей ряд, а потім на його підставі зробити перерахунок.

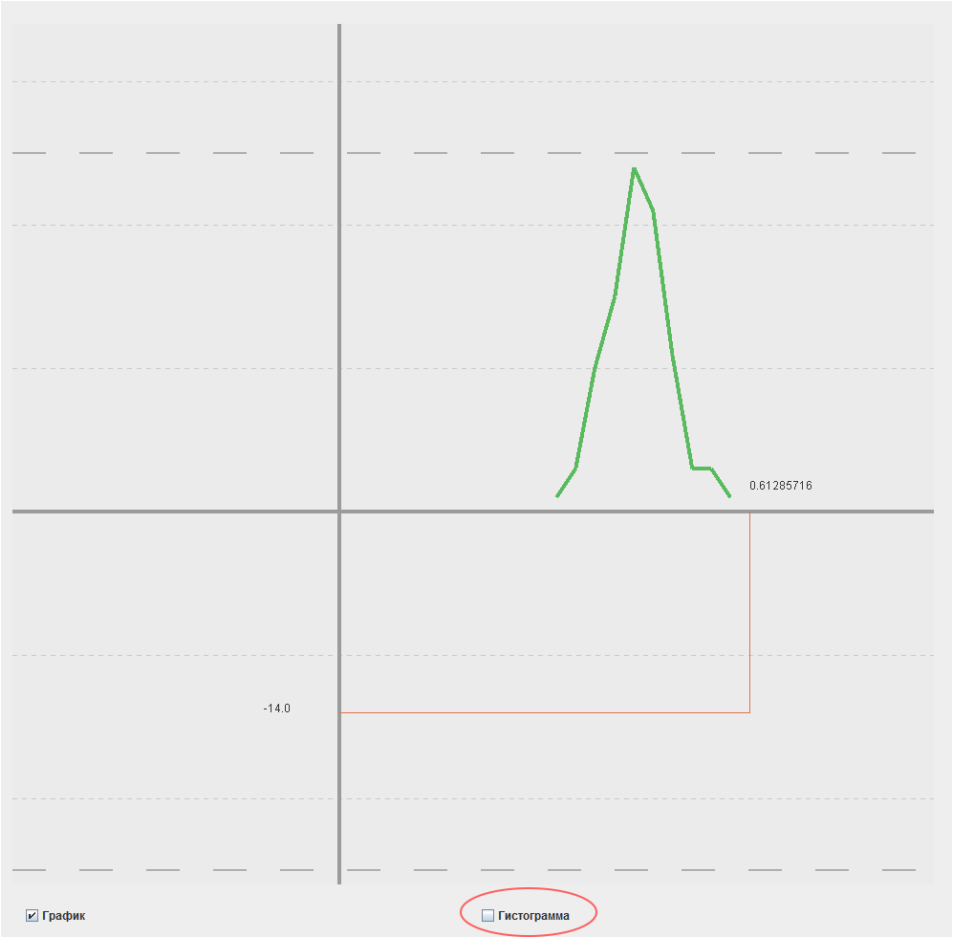


Рисунок 22 – Приклад графіку даних

В кінці можна спробувати закрити програму, і в разі, якщо розрахунки були збережені, то додаток запропонує зберегти їх (рис. 23). На малюнку збереження можна помітити, що можна зберегти в тестовому файлі, а можна в excel на увазі таблиці (рис. 24).

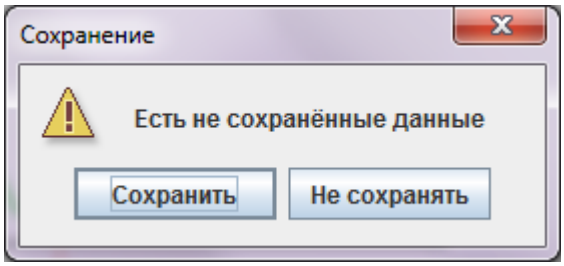


Рисунок 23 – Приклад повідомлення щодо зберігання даних

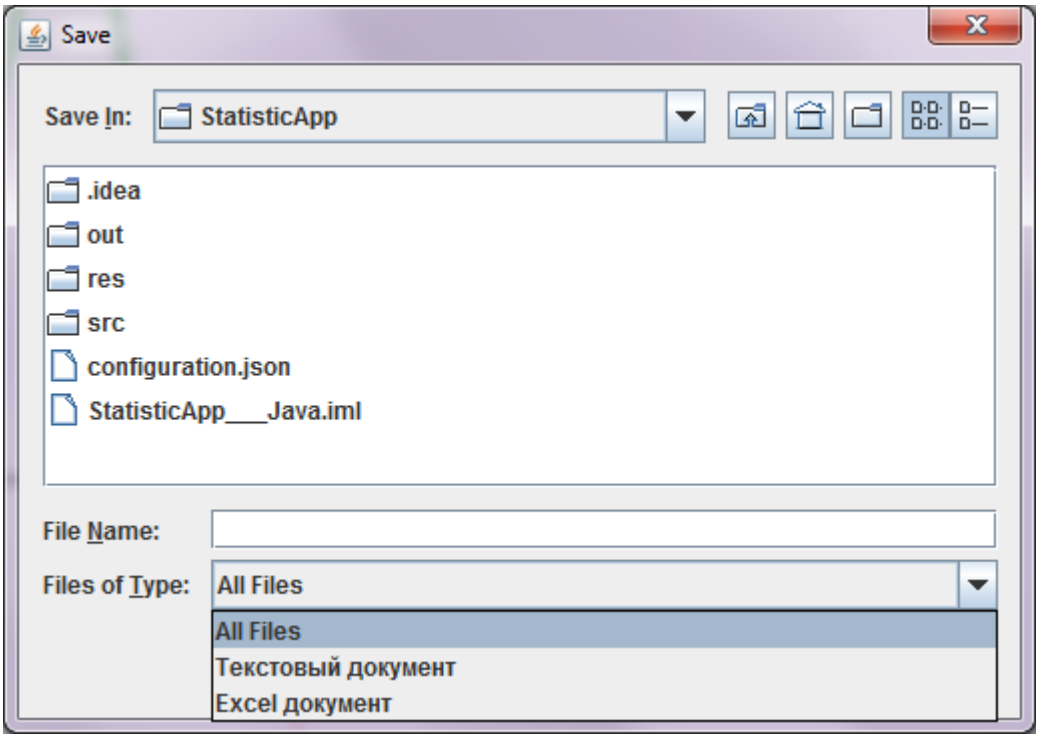


Рисунок 24 – Вибір формату збереження даних

Якщо файл вже існує, то додаток запропонує три варіанти на вибір – перезаписати, вибрати інший або не зберігати зовсім (рис. 25):

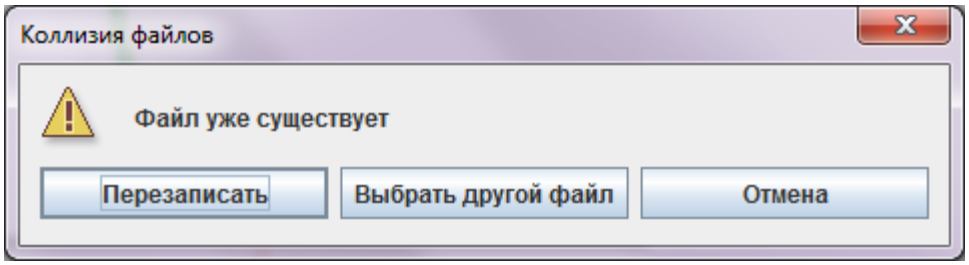


Рисунок 25 – Вибір дії у разі колізії файлів

Додаток запам'ятовує, де була здійснена остання завантаження даних і де користувач зберігав результати. Це дуже зручно, не потрібно кожного разу шукати директорію.

3.2 Відображення результатів розрахунків

Вікно з розгорнутими розрахунками має наступний вигляд (рис. 26):

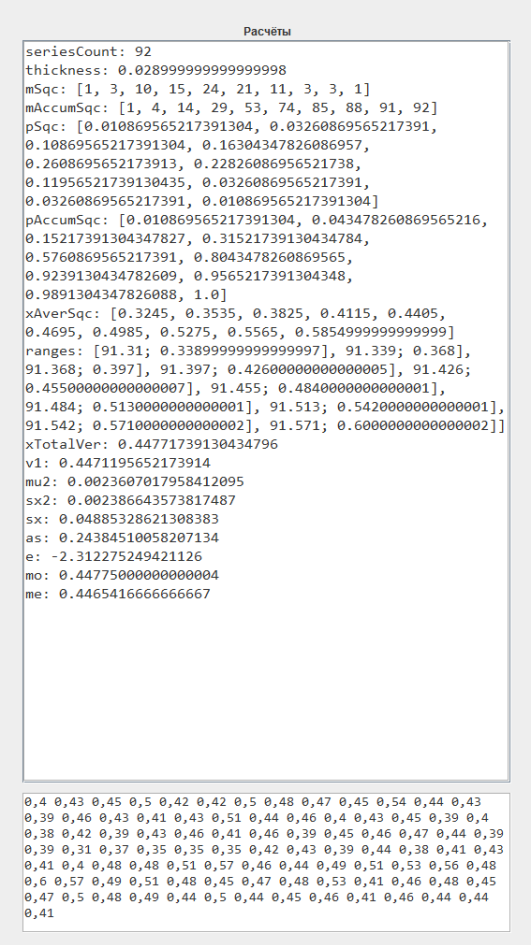
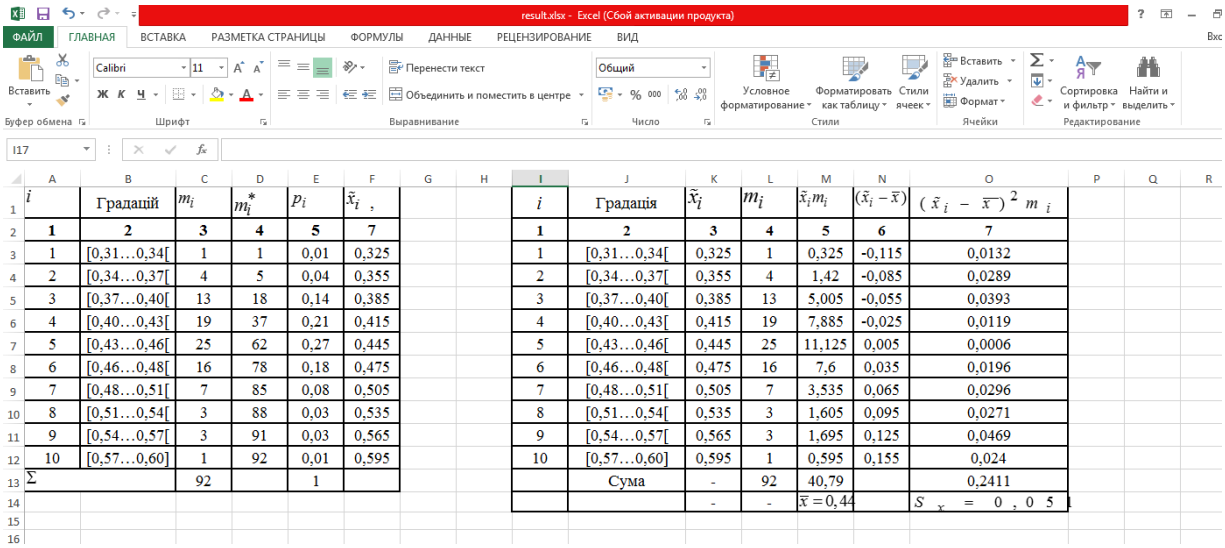


Рисунок 26 – Приклад виводу розрахунків

В результаті роботи програмного додатку користувач отримує повну інформацію щодо кожного параметру, який був задіяно у алгоритмах. При виконанні практичних робіт студенти аналізують чи порівнюють отримані показчики.

Якщо є необхідність у збережені даних у вигляді окремого файлу, текстового чи файлу формату excel, у програмному додатку є функція «зберегти у файлі» (рис. 27):



result.xlsx - Excel (Сбой активации продукта)

ФАЙЛ | ГЛАВНАЯ | ВСТАВКА | РАЗМЕТКА СТРАНИЦЫ | ФОРМУЛЫ | ДАННЫЕ | РЕЦЕНЗИРОВАНИЕ | ВИД

i	Градацій	m_i	m_i^*	p_i	$\tilde{x}_i$,
1	2	3	4	5	7
1	[0,31…0,34[	1	1	0,01	0,325
2	[0,34…0,37[	4	5	0,04	0,355
3	[0,37…0,40[	13	18	0,14	0,385
4	[0,40…0,43[	19	37	0,21	0,415
5	[0,43…0,46[	25	62	0,27	0,445
6	[0,46…0,48[	16	78	0,18	0,475
7	[0,48…0,51[	7	85	0,08	0,505
8	[0,51…0,54[	3	88	0,03	0,535
9	[0,54…0,57[	3	91	0,03	0,565
10	[0,57…0,60]	1	92	0,01	0,595
Σ		92		1	

i	Градація	$\tilde{x}_i$	m_i	$\tilde{x}_i m_i$	$(\tilde{x}_i - \bar{x})$	$(\tilde{x}_i - \bar{x})^2 m_i$
1	2	3	4	5	6	7
1	[0,31…0,34[	0,325	1	0,325	-0,115	0,0132
2	[0,34…0,37[	0,355	4	1,42	-0,085	0,0289
3	[0,37…0,40[	0,385	13	5,005	-0,055	0,0393
4	[0,40…0,43[	0,415	19	7,885	-0,025	0,0119
5	[0,43…0,46[	0,445	25	11,125	0,005	0,0006
6	[0,46…0,48[	0,475	16	7,6	0,035	0,0196
7	[0,48…0,51[	0,505	7	3,535	0,065	0,0296
8	[0,51…0,54[	0,535	3	1,605	0,095	0,0271
9	[0,54…0,57[	0,565	3	1,695	0,125	0,0469
10	[0,57…0,60]	0,595	1	0,595	0,155	0,024
	Сума	-	92	40,79		0,2411
		-	-	$\bar{x} = 0,44$		$S_x = 0,051$

Рисунок 27 – Результат збереження даних в excel

Повний вид результату роботи програми представлено на рисунку 28:

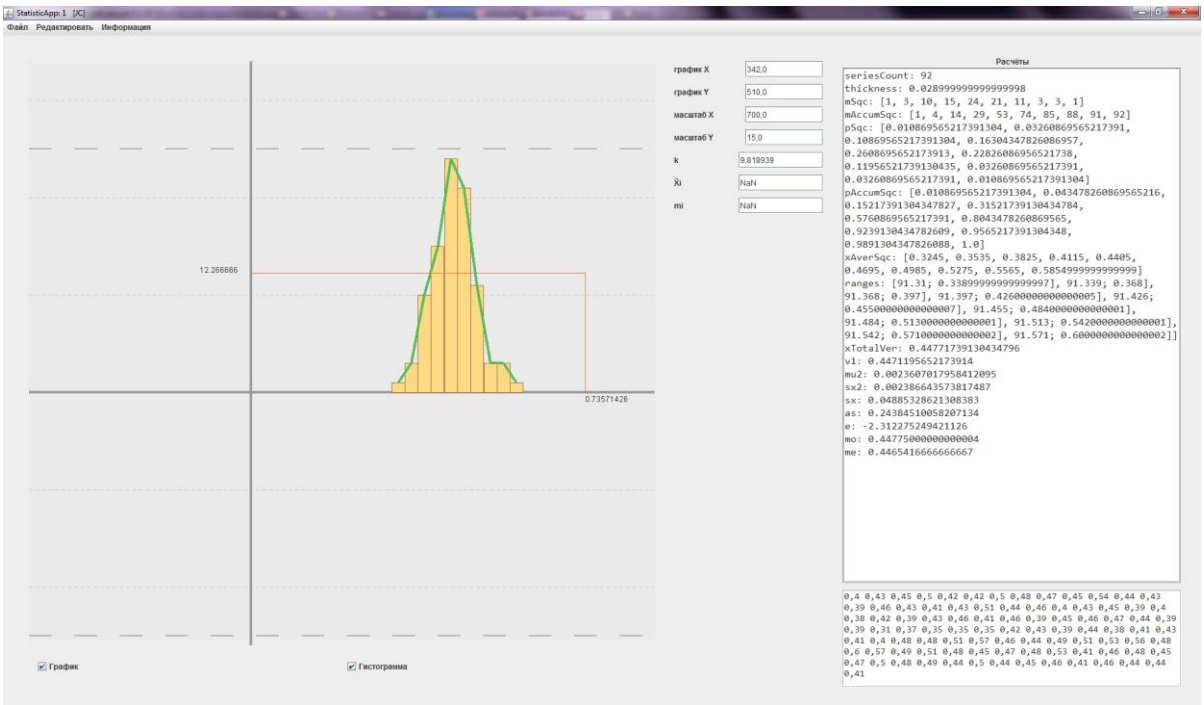


Рисунок 28 – Результат розрахунків на головній формі додатку

4 ОПИС ПРОГРАМНОЇ ЧАСТИНИ ДОДАТКУ

4.1 Використані бібліотеки

Тепер час розібрати внутрішню реалізацію програми та як вона влаштована. Розглянемо бібліоте, які використовувалися додатком.

1. ChancyValue.

Основна суть – це валідація роботи гілки викликів. Розберемо на прикладі нашого застосування – нам треба вважати дані з файлу, розпарсити їх в масив і обробити. Лише після успішного виконання відбувається відображення результатів.

При читанні з файлу ми перевіряємо наявність файлу і доступу читання. Якщо ці умови невірні, то метод поверне «ChancyValue» з кодом проблеми, інакше поверне лічені дані на увазі рядки. При пасігне даних може піти щось не так. Банально, це може бути невірний файл, що не містить рядів, або структура файлу може бути пошкоджена. І в такому випадку парсер так само поверне «ChancyValue» з кодом помилки. І нарешті обробка так само може повернути або успішний результат, або код помилки.

Це краще блоку try-catch і відлов помилок. По-перше метод сам валідірует помилку, сам відповідає за те, що відбувається всередині нього. Якщо в його обробці щось відбувається не так, він сигналізує про це метод рівнем вище, який не знає, що за обробка відбувається всередині.

Зовнішній метод по «ChancyValue» може посторить вже свою логіку роботи. Так само не відбувається нагромажденій блоків «try-catch» і такий код більш читаємо. І на останок – помилки гальмують роботу програми, оперуючи по-особливому зі стеком. У тих випадках, коли можна передбачити помилку, можна залишити тільки код, ніж гальмувати програму викидом помилки.

2. Logger.

Основне завдання Логгер в тому, щоб записувати максимально все те, що відбувається з програмою. Так розробнику простіше здійснювати

підтримку програми. У разі, якщо щось пішло не так і додаток нормально не відпрацьована, йому не треба допитувати клієнта про те, які дії привели до такого результату, він може це прочитати в логах.

До того ж, на етапі безпосередньої розробки так простіше налагоджувати додаток – видно, як програма перетікає з одного стану в інший.

3. ScopeFactory.

Наприклад, в додатку є безліч всяких сервісів і компонентів, , є сервіс по збереженню результатів в файл, є сервіс по обрахунку статистики, є конфігурації, сервіс по роботі з файлами, сервіс по роботі з вікнами, вікно і його компоненти. Всі ці елементи мають можливість роботи один з одним. Абсолютно недоцільно в кожному такому елементі створювати полем посилання на кожен елемент, з яким він може взаємодіяти.

Причин кілька:

- зменшується читаність коду;
- збільшується його об'єм;
- збільшується об'єм пам'яті під екземпляр класу.

ScopeFactory створює окрему область, в яку поміщаються об'єкти. Далі кожен елемент може отримувати посилання на ці об'єкти в будь-який момент.

До того ж для зручності Scope має можливість створювати об'єкт, якщо такого не було в області раніше. Так можна створювати цілі ланцюжки методів, без перевірки на null.

4. Sequence.

Це інша реалізація стандартного «ArrayList» мови java. Основними особливостями бібліотеки – це гнучка розширюваність і можливість доступатся до "сирих" даних, тобто масиву.

Це дозволяє всіляко надстраиваться над цією структурою даних, а так само швидко ітерованих по ній. На проекті Sequence використовувалася для зберігання динамічних масивів. Це масиви, які здатні до розширення, але статистично мало до цього схильні, тому списку не доцільно було використовувати для таких даних, по скільки список більш важкоатлет і

повільніше в плані ітерірования. Так само реалізація ScopeFactory побудована на Sequence.

5. LList.

Це бібліотека, що заміщає стандартний LinkedList мови java. Особливості подібні для Sequence. Присутній гнучкість для розширення, чого в стандартному немає. За рахунок чого, або неправильно вибудовується архітектура проекту (модуля) або зовсім нераціонально.

Так само найважливішим недоліком стандартного списку – це закриті контейнери. Це тягне за собою створення ітераторів для обходу структури даних. Крім цього, список двусвязний, але реалізація Джави не реалізує весь потенціал двусвязного списку - видалення по елементу або вставка в середину списку завжди тягне за собою спочатку пошук потрібного контейнера, і лише потім проводиться операція.

До того ж, якщо треба якось доповнити контейнер, то best practices пропонують створення врапперов, які не дотримуються принципів архітектури, не кажучи вже про те, що відбувається створення нової сутності і не можна толком узагальнити цей врапперов. І потім доводиться вводити надстандарти, щоб орієнтуватися в цьому і з ним працювати. Тут так само передбачаються QueryAPI, який дозволяє, викликаючи методи по ланцюжку фільтрувати і конвертувати дані всередині. Це подібно до StreamAPI в java.

6. Gson.

Бібліотека від Google, яка призначена для сериализации і десеріалізациі даних в json формат. На проекті я використовував це для зберігання конфігурацій. Json легко редагувати вручну і він по обьему набагато менше xml, тому вибір впав в його сторону.

7. ActCompilation.

Це набір інструментів, який включає в себе таких інструментів, як:

- файловий інструментарій;
- інструменти по роботі з рядками;
- інструменти по роботі з помилками;

– набір математичних функцій.

Так само всередині є сервіс по збереженню. Всередині він має базову логіку по відкриттю файлу на підставі останнього відкритого. FileManager здійснює зручний інтерфейс з читання / запису в файл, а так же набір методів по отриманню розширення файлу, конкатенації шляху, заміни батьківської директорії (особливо корисно при обробці цілого дерева директорій) і іншого. StringUtils має методи по парсингу рядки в масиви примітивів, а ThrowableUtils по формуванню stackTrace.

8. ControlsAct.

Призначена для обробки подій після натискання на кілька клавіш. Справа в тому, що в Джаві немає реалізації, яка б могла по набору клавіш здійснювати дію. Колбекі спрацьовують при натисканні будь-якої клавіші, і всередині колбека треба запам'ятовувати і фільтрувати послідовності клавіш. Для зручності це було винесено в окрему бібліотеку. Це активно використовується в графіку, де по комбінаціям можна рухати і масштабувати графік.

9. GraphBuilder.

У джава спочатку немає коштів для малювання графіків, є лише примітиви, наприклад, отрисовать лінію, прямокутник або коло. GraphBuilder вирішує цю проблему - на базі примітивів він формує Декартових двовимірну площа, а так само надає можливість гнучко задати її контент – тут можна задавати осі, різні позначки, що обробляється графік по формулі або по точкам, є гістограми і водоспади. Дає цілий спектр в плані налаштування кожного елемента, такі як колір, товщина і форма (суцільна лінія, штрих, інше). Ця бібліотека була використана для побудови графіка і гістограми.

10. UIAct.

Це зібрання візуальних компонентів і інших інструментів для їх роботи. Серед компонентів можна виділити VerticalLabel, який отрісовиваєт рядок вертикально. Є HeaderedTable, який формує таблицю з набором вертикальних і горизонтальних хедеров, ImageView для відображення картинки, підписані

текстові поля та інше. Крім цього, є кілька типів макетів, в java аналоги називаються як layout. Є сітковий макет, який контент розміщує по сітці. Є макет який розміщує контент пропорційно розміру вікна.

4.2 Опис основних функцій коду

Життєвий цикл програми: при старті формуємо глобальний Скоуп, туди поміщаємо інстанси FileManager, конфігурації, менеджер по роботі з рядами і віконний менеджер. Віконний менеджер відразу ж створює одне вікно. Саме з цим вікном ми і працюємо.

Далі вибираємо файл з даними для завантаження. Якщо файл був прочитаний успішно і структура файлу відповідає вимогам, тобто файл розпарсити, то можна приступати до обрахунку. Якщо у файлі кілька рядів, то створяться кілька вікон – кожне окремо для ряду.

Після обрахунку програма дозволяє управляти графіком, є можливість змінювати вихідний ряд, а так же параметр k, і потім перерахувати дані.

По закінченню роботи можна зберегти результати в текстовий файл, або файл excel.

Розберемо код програми. В даному випадку будемо розбирати обрахунку. Спочатку нам надходить ряд на увазі масиву. Ми його сортуємо:

```
QuickSort.sortPrimitive (series;
```

Далі визначаємо як константи розмірність ряду, так як відносно часто будемо використовувати цей параметр при обрахунку, а так само мінімальне і максимальне значення ряду. У сортованому масиві це буде перше і останні значення відповідно.

```
final int SERIES_COUNT = series.length;
final double MIN = series [0];
final double MAX = series [SERIES_COUNT - 1];
```

Наступним кроком визначаємо товщину стовпчика. Суть статистики в тому, щоб відсортоване ряд розбити на рівні проміжки. Кількість проміжків розраховується за формулою k = 5 * log (n), де n це кількість елементів ряду. Але його так само може ввести користувач.

Відповідно ширина стовпця мудет обраховуватися на підставі k, а так само мінімального і максимального значень ряду:

```
final double THICKNESS = (MAX - MIN)/(double) k;
final int VALUES_COUNT = k;
```

Знаючи кількість стовпців можна заздалегідь виділити всі необхідні масиви результатів, а саме кількість елементів ряду, що потрапляють в частковий інтервал, накопичення цієї величини, відносні інтервальні частоти, накопичення частот, середні значення інтервалів, межі інтервалів

```
int[] mSqc = new int[VALUES_COUNT];
int[] mAccumSqc = new int[VALUES_COUNT];
double[] pSqc = new double[VALUES_COUNT];
double[] pAccumSqc = new double[VALUES_COUNT];
double[] xAverSqc = new double[VALUES_COUNT];
StatisticResult.XRange[] ranges = new
                      StatisticResult.XRange[VALUES_COUNT];
```

Далі визначаємо першу границю і починаємо обхід циклом весь сортовані ряд, при цьому вважаємо кількість значень ряду. Як тільки значення ряду перевалить через кордон, ми поміщаємо вирахувана кількість ряду в масив mSqc, продовжуємо кордон, а так само визначаємо і інші параметри. Деякі з них залежать від mSqc.

```
int valuesIndex = 0;
     int mi = 0;
     double pAccum_sum = 0;
     double miEdge = MIN + THICKNESS;
     for (int i = 0, m_sum = 0; i < SERIES_COUNT;++i){
          double value = series[i];
          if (value < miEdge)
               ++mi;
```

```
                else{
                     mSqc[valuesIndex] = mi;
                     ranges[valuesIndex] = new
StatisticResult.XRange(miEdge - THICKNESS, miEdge);
                     miEdge += THICKNESS;
```

Після запису у масив визначаємо наступні параметри:

```
                     m_sum += mi;
                     mAccumSqc[valuesIndex] = m_sum;

                     pSqc[valuesIndex] = (mi /(double)
SERIES_COUNT);
                     pAccum_sum += pSqc[valuesIndex];
                     pAccumSqc[valuesIndex] = pAccum_sum;
      xAverSqc[valuesIndex] = MIN + ((valuesIndex << 1) + 1) *
THICKNESS / 2;
                     mi = 1;
                     ++valuesIndex;}
```

Наступним кроком визначаємо середнє значення по ряду:

```
final int COUNT = series.length;
double sum = 0;
for (int i = 0; i < COUNT; ++i)
              sum += series[i];
return sum / COUNT;
```

Далі необхідно визначити такі значення як перший початковий момент, центральний момент розподілу другого порядку, ефективна оцінка дисперсії випадкової величини, а також середньоквадратичне відхилення.

```
double v1;
double mu2;
double sx2;
double sx;
double v1_sum = 0;
double mu2_sum = 0; // equals to sx2_sum
     for (int i = 0; i < VALUES_COUNT; ++i){
          int mi = mSqc[i];
          double xAveri = xAverSqc[i];
```

Далі йде розрахунок сум показчиків:

```
            v1_sum += xAveri * mi;
            mu2_sum += sqr(xAveri - xTotalAver) * mi; }

        v1 = v1_sum / SERIES_COUNT;
        mu2 = mu2_sum / SERIES_COUNT;
        sx2 = mu2_sum / (SERIES_COUNT - 1);
        sx = Math.sqrt(sx2);
```

Следующими данными являются коэффициент асимметрии и коэффициент эксцеса.

```
double as;  double e;  double as_sum = 0; double e_sum = 0;
double sx_3 = Math.pow(sx, 3);
     double sx_4 = sqr(sx2);
     for (int i = 0; i < VALUES_COUNT; ++i){
          int mi = mSqc[i];
          double xAveri = xAverSqc[i];

          as_sum = Math.pow(xAveri - xTotalAver, 3) * mi / sx_3;
          e_sum = Math.pow(xAveri - xTotalAver, 4) * mi / sx_4; }
     as = as_sum / SERIES_COUNT;
     e = e_sum / SERIES_COUNT - 3;
```

Останніми параметрами, які нам необхідні для результату це мода і медіана. Для початку треба знайти потрібний інтервал. Для моди це той інтервал, в який потрапило найбільше елементів ряду. Для медіани ж це той інтервал, у якого накопичена частота перевалила за 50%.

Розраховуємо моду:

```
int index = 0; int max = 0;
          final int COUNT = mSqc.length;
          for (int i = 0; i < COUNT; ++i){
               int m = mSqc[i];
               if (m > max){ max = m;  index = i; } }
          return index;
```

Розраховуємо медіану:

```
          final int COUNT = pAccumSqc.length;
          for (int i = 0; i < COUNT; ++i)
               if (pAccumSqc[i] >= 0.5f)
```

```
                return i;
        return -1;
```

І лише після того, як будуть обчислені індекси інтервалів, лише тоді можна обрахувати дані параметри:

```
double mo; double me;
     int moIndex = getMoIndex(mSqc);
     double x0 = ranges[moIndex].xMin;
     mo = x0 + THICKNESS * (mSqc[moIndex] - mSqc[moIndex - 1]) /
((mSqc[moIndex] << 1) - mSqc[moIndex - 1] - mSqc[moIndex + 1]);

     int meIndex = getMeIndex(pAccumSqc);
     me = ranges[meIndex].xMin + THICKNESS * ((SERIES_COUNT >> 1)
- mAccumSqc[meIndex - 1]) / mSqc[meIndex];
```

ВИСНОВКИ

Об’єкт розробки являє собою програмний додаток для студентів ОДЕКУ. Він стане допомогою у практичних заняттях з дисципліни «Обробка і аналіз інформації» для студентів напряму підготовки 6.040106 «Екологія, охорона навколишнього середовища та збалансоване природокористування».

Дипломний проект допоможе у побудові групованих рядів для задач метеорології та кліматології.

На першій стадії розробки диплома було виконано:

- аналітичний огляд предметної області, а саме методичні вказівки вищезазначених дисциплін і аналоги програмного додатку;
- за результатами аналізу рахом із замовником сформовані вимоги і задачі до об’єкту розробки;
- обрані програмні засоби для реалізації програмного додатку.

На етапі проектування виконано:

- побудовані UML-діаграми Use-Case, діяльності та діаграми класів для об’єкту розробки;
- розроблено алгоритм для паралельної обробки вхідних даних для прискорення розрахунків.

Під час розробки були запрограмовані такі вимоги до проекту, як:

- реалізація алгоритму розрахунку даних;
- розробка оптимального інтерфейсу користувача для полегшення вводу інформації;
- реалізація функції для даних «зберегти у файл»;
- розробка додаткових функцій для паралельної обробки великих рядків даних з метою прискорення часу обчислення.

Подальшим розвитком проекту буде доповнення алгоритму функціями для обчислювання іншими методами, які використовують студенти ОДЕКУ під час свого навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тема 7. Ряды распределения. URL: http://www.hi-edu.ru/e-books/xbook096/01/part-007.htm. (дата звернення 01.03.2020).
2. Группировка данных и построение ряда распределения. URL: https://math.semestr.ru/group/group.php. (дата звернення 01.03.2020).
3. Расчет сгруппированных рядов. URL: https://www.psychol-ok.ru/statistics/12. (дата звернення 03.03.2020).
4. JetBrains IntelliJ IDEA. URL: https://itpro.ua/product/jetbrains-intellij-idea/?tab=description. (дата звернення 11.03.2020).
5. IntelliJ IDEA – решение для топ-разработчиков. URL: https://webformyself.com/intellij-idea-reshenie-dlya-top-razrabotchikov/. (дата звернення 11.03.2020).
6. Изменения в языке Java 8. URL: https://www.ibm.com/developerworks/ru/library/j-java8lambdas/index.html. (дата звернення 15.03.2020).
7. Информация о Java 8. URL: https://www.java.com/ru/download/faq/java8.xml. (дата звернення 16.03.2020).
8. SWING – Краткое руководство. URL: https://coderlessons.com/tutorials/java-tekhnologii/nauchitsia-kachatsia/swing-kratkoe-rukovodstvo. (дата звернення 16.03.2020).
9. Проектирование графического интерфейса пользователя. URL: https://habr.com/ru/post/208966/. (дата звернення 22.03.2020).
10. Ряды распределения и их построение/. URL: http://www.hi-edu.ru/e-books/xbook096/01/part-008.htm. (дата звернення 24.03.2020).
11. Статистичні критерії перевірки основної гіпотези. URL: https://studopedia.com.ua/1_47878_statistichniy-kriteriy-perevirki-osnovnoi-gipotezi.html. (дата звернення 25.03.2019).

12. Лоєва І. Д., Бургаз О.А. Збірник методичних вказівок до практичних занять з дисципліни «Обробка і аналіз інформації» для студентів денної форми навчання, напрям підготовки 6.040106 «Екологія, охорона навколишнього середовища та збалансоване природокористування». Одеса: ОДЕКУ, 2010. 65 с.
13. Проектирование графического интерфейса пользователя. URL: https://habr.com/ru/post/208966/. (дата звернення 25.03.2020).
14. Диаграмма вариантов использования. URL: https://www.uml-diagrams.org/use-case-diagrams.html. (дата звернення 25.03.2020).
15. UML 2 Use Case Diagrams. URL: www.agilemodeling.com/artifacts/useCaseDiagram.htm. (дата звернення 28.03.2020).
16. Диаграмма активностей (Activity diagram). URL: https://flexberry.github.io/ru/fd_activity-diagram.html. (дата звернення 02.04.2020).
17. Отношения классов – от UML к коду. URL: https://habr.com/ru/post/150041/. (дата звернення 06.04.2020).