Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# КУРСОВА РОБОТА

з дисципліни: «Інженерія програмного забезпечення»

на тему: програмний додаток «Редактор кільцевих діаграм»

Студента 2-го курсу групи ІО-12

Напряму підготовки: 6.050102 Комп’ютерна інженерія

Спеціальності: 7.8.05010201 Комп’ютерні системи та мережі

Мроця Юрія Богдановича

Керівник к.т.н., доцент Абу Усбах О. Н.

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь,прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь,прізвище та ініціали)

Київ - 2013 рік

ЗМІСТ

ВСТУП……………………………………………………………………..……….3

1. ТЕХНІЧНЕ ЗАВДАННЯ………………………………………………….…..…4

1.1 Загальне завдання..……………………………..………..…....….….…...4

1.2 Функціональність..………………………………..………...……....……4

1.3 Вимоги до реалізації…………………………….…….….….…..….…...4

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ……………………..…….…..5

2.1 Прецеденти..……………………………….………………………….......5

2.2 Граничні класи …………………………………………………….…….9

2.3 Проектування графічного інтерфейсу користувача…..…...…..…...…10

2.4 Проектування структури ……………………….……………….........…12

3.РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ……….……………………………....13

3.1 Розробка структури ………………………………………………..........13

3.2 Інструкції для користувача ………………….….…….....……………...14

3.3 Специфікації класів ……………………………………………..……....14

4. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.………………………..…17

ВИСНОВКИ…………………………….…………………………………………….21

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………….…...22

ДОДАТКИ…………………………………………………………………………....23

А. Лістинг коду програмного додатку……………………………….……..23

ВСТУП

Метою курсової роботи є розробка програмного додатку для редагування кругових діаграм. Програма повинна мати простий і зручний інтерфейс, а також набір операцій для маніпулювання над вищезгаданими діаграмами. На етапі проектування потрібно ознайомитись з двома графічними пакетами мови програмування Java, а саме: Swing та AWT; вибрати один з них для реалізації інтерфейсу користувача та обґрунтувати свій вибір.

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Загальне завдання

Потрібно розробити програмний додаток, що буде візуалізовувати кругові діаграми та матиме можливість оперувати над ними.

1.2 Функціональність

Програма повинна мати наступну функціональність:

* + - можливість завантаження/збереження даних у форматі CSV-файлу;
    - візуалізація даних у вигляді кругової діаграми;
    - можливість швидко і зручно оперувати діаграмою;
    - можливість відкрити декілька діаграм в одному вікні і перемикати їх за допомогою вкладок;

1.3 Вимоги до реалізації

* + - Використанням мови програмування Java і бібліотеки SWING;
    - заборонено використовувати спеціалізовані компоненти для побудови діаграм;
    - інтерфейс користувача має забезпечувати доступ до всієї функціональності програми;
    - проект має бути повністю задокументований за допомогою JavaDoc;
    - проект має повністю відповідати правилам CheckStyle;
    - можливість роботи програми з більше ніж з однією серією даних

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1 Прецеденти

2.1.1 Початковий опис потоку подій роботи з редактором кільцевих діаграм

1. Прецедент починається, коли користувач запускає програму.
2. Користувач може виконати одну з наступних дій:
3. Створити файл.
4. Зберегти файл.
5. Відкрити збережений файл.
6. Додати сектор в діаграму.
7. Видалити сектор з діаграми.
8. Змінити назву сектора.
9. Змінити колір сектора.
10. Змінити розмір сектора.
11. Закрити файл.
12. Очистити діаграму.
13. Прецедент закінчується, коли користувач завершує роботу з програмою.

2.1.2 Розробка орієнтирів

1. Прецедент починається, коли користувач запускає програму.
2. Користувач може виконати одну з наступних дій:
   1. Створити файл. [Створюється кільцева діаграма з назвою diagram1 і сектором, що займає 100% місця і називається sector1.]
3. Зберегти файл. [Файл повинен мати розширення CSV.]
4. Відкрити збережений файл. [Файл відкривається в новій вкладці, яка після відкриття буде активною.]
5. Додати сектор в діаграму. [Користувач повинен мати можливість ввести назву, розмір або відсоток, що займатиме новий сектор.]
6. Видалити сектор з діаграми.
7. Змінити колір сектора.
8. Змінити назву сектора.
9. Змінити розмір сектора.
10. Закрити файл. [Якщо файл не був збережений, користувач має бути повідомлений про це.]
11. Очистити діаграму. [Користувач повинен підтвердити очищення всієї діаграми.]
12. Прецедент закінчується, коли користувач завершує роботу з програмою.[Якщо є незбережені файли, користувач отримує повідомлення про це.]

2.1.3 Розробка атрибутів

1. Прецедент починається, коли користувач запускає програму.
2. Користувач може виконати одну з наступних дій:
   1. Створити файл.
3. Зберегти файл. {Назва файлу в середньому 8 символів.}
4. Відкрити збережений файл. {Назва файлу в середньому 8 символів.}
5. Додати сектор в діаграму. {Середній розмір сектора - 200}
6. Видалити сектор з діаграми.
7. Змінити колір сектора.
8. Змінити назву сектора. { Назва сектору в середньому 8 символів.}
9. Змінити розмір сектора. {Середній розмір сектора - 200}
10. Закрити файл.
11. Очистити діаграму.
12. Прецедент закінчується, коли користувач завершує роботу з програмою.

2.1.4 Розробка інтенсивності використання

1. Прецедент починається, коли користувач запускає програму.
2. Користувач може виконати одну з наступних дій:
   1. Створити файл. (Використовується в 60% випадків.)
3. Зберегти файл. (Використовується в 95% випадків.)
4. Відкрити збережений файл. (Використовується в 40% випадків.)

d) Додати сектор в діаграму. (Використовується в 85% випадків.)

e) Видалити сектор з діаграми. (Використовується в 70% випадків.)

f) Змінити колір сектора. (Використовується в 50% випадків.)

g) Змінити назву сектора. (Використовується в 55% випадків.)

h) Змінити розмір сектора. (Використовується в 40% випадків.)

i) Закрити файл. (Використовується в 100% випадків.)

j) Очистити діаграму. (Використовується в 35% випадків.)

1. Прецедент закінчується, коли користувач завершує роботу з програмою.

2.1.5 Результуючий опис потоку подій програми

1) Прецедент починається, коли користувач запускає програму.

2) Користувач може виконати одну з наступних дій:

a) Створити файл. (Використовується в 60% випадків.); [Створюється кільцева діаграма з назвою diagram1 і сектором, що займає 100% місця і називається sector1.]

b) Зберегти файл. (Використовується в 95% випадків.); {Назва файлу в середньому 8 символів.}; [Файл повинен мати розширення CSV.]

c) Відкрити збережений файл. (Використовується в 40% випадків.); {Назва файлу в середньому 8 символів.}; [Файл відкривається в новій вкладці, яка після відкриття буде активною.]

d) Додати сектор в діаграму. (Використовується в 85% випадків.); {Середній розмір сектора - 200}; [Користувач повинен мати можливість ввести назву, розмір або відсоток, що займатиме новий сектор.]

e) Видалити сектор з діаграми. (Використовується в 70% випадків.)

f) Змінити колір сектора. (Використовується в 50% випадків.)

g) Змінити назву сектора. (Використовується в 55% випадків.); { Назва сектору в середньому 8 символів.}

h) Змінити розмір сектора. (Використовується в 40% випадків.); {Середній розмір сектора - 200}

i) Закрити файл. (Використовується в 100% випадків.)

j) Очистити діаграму. (Використовується в 35% випадків.); [Користувач повинен підтвердити очищення всієї діаграми.]

3) Прецедент закінчується, коли користувач завершує роботу з програмою. [Якщо є незбережені файли, користувач отримує повідомлення про це.

2.1.6 Доповнення опису ілюстрованого сценарію об’єктами діаграми

1) Прецедент починається, коли користувач запускає програму. «Редактор кільцевих діаграм»

2) Користувач може виконати одну з наступних дій:

a) Створити файл. (Використовується в 60% випадків.); [Створюється кільцева діаграма з назвою diagram1 і сектором, що займає 100% місця і називається sector1.]; «Кнопка Новий файл»

b) Зберегти файл. (Використовується в 95% випадків.); {Назва файлу в середньому 8 символів.}; [Файл повинен мати розширення CSV.]; «Кнопка Зберегти файл»

c) Відкрити збережений файл. (Використовується в 40% випадків.); {Назва файлу в середньому 8 символів.}; [Файл відкривається в новій вкладці, яка після відкриття буде активною.]; «Кнопка Відкрити файл»

d) Додати сектор в діаграму. (Використовується в 85% випадків.); {Середній розмір сектора - 200}; [Користувач повинен мати можливість ввести назву, розмір або відсоток, що займатиме новий сектор.]; «Кнопка Додати сектор»

e) Видалити сектор з діаграми. (Використовується в 70% випадків.); «Кнопка Видалити сектор»

f) Змінити колір сектора. (Використовується в 50% випадків.); «Кнопка Змінити параметри сектора»

g) Змінити назву сектора. (Використовується в 55% випадків.); { Назва сектору в середньому 8 символів.} ; «Кнопка Змінити параметри сектора»

h) Змінити розмір сектора. (Використовується в 40% випадків.); {Середній розмір сектора - 200}; «Кнопка Змінити параметри сектора»

i) Закрити файл. (Використовується в 100% випадків.) ; «Кнопка Закрити файл»

j) Очистити діаграму. (Використовується в 35% випадків.); [Користувач повинен підтвердити очищення всієї діаграми.] ; «Кнопка Очистити діаграму»

3) Прецедент закінчується, коли користувач завершує роботу з програмою. [Якщо є незбережені файли, користувач отримує повідомлення про це.]

2.2 Діаграма граничних класів

На рис. 2.1 зображена діаграма класів програмного додатку “Редактор кільцевих діаграм”.

******

Рис. 2.1 Діаграма граничних класів програмного додатку

2.3 Проектування графічного інтерфейсу користувача

На рисунку 2.3.1 зображено ескіз моделі графічного інтерфейсу програмного додатку “Редактор кільцевих діаграм”. На ньому виділено області для розташування всіх елементів. Графічний інтерфейс розроблявся таким чином, щоб користувач мав можливість швидко і легко знайти потрібну йому інформацію чи вибрати операцію. Також при розробці інтерфейсу брались до уваги праці [8, 9], де автори описують тонкі аспекти побудови моделі вікна програмного додатку, які суттєво впливають на його привабливість і легкість сприйняття.

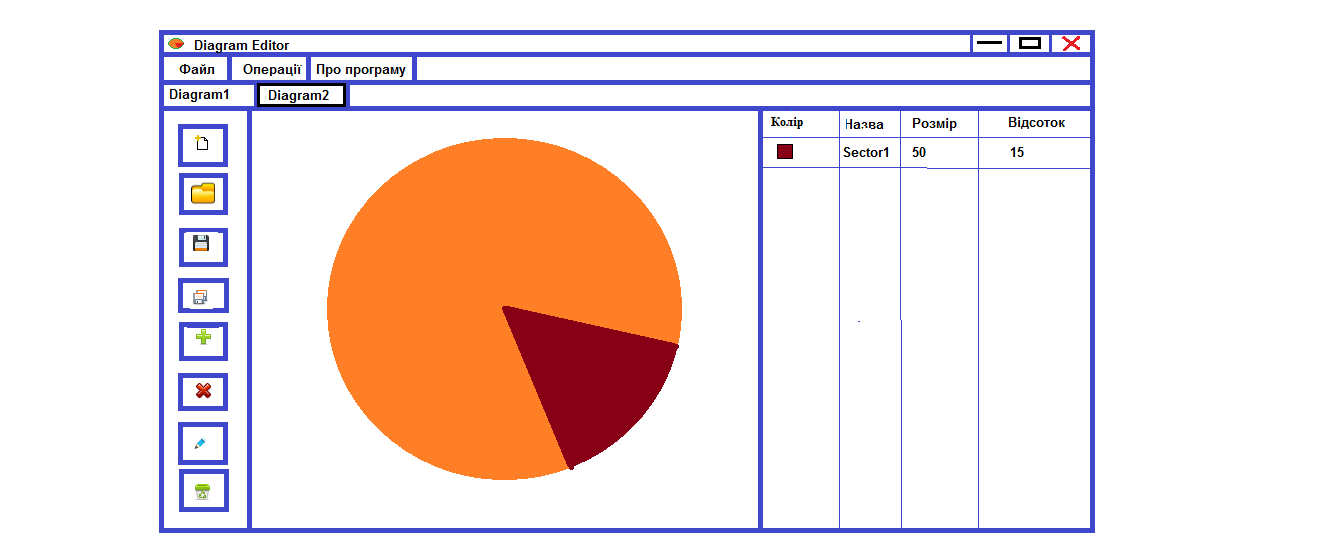
**

Рис. 2.1 Ескіз моделі графічного інтерфейсу

Графічний інтерфейс розроблений за допомогою стандартних засобів Java. Всі бібліотеки, які були використані наведені у табл. 2.3.1(див. Додаток Б) .

2.4 Проектування структури

На етапі проектування будо створено наступні класи.

“Client” – головний клас, який використовується для запуску програми. “DE\_ModelDelegate” – клас, який відіграє роль вікна і вміщує в собі усі об’єкти граничних класів. “CSVProcessor” – клас, в якому реалізовані методи завантаження/вивантаження CSV-файлу в/з агрегованого об’єкту класу ArrayList, в якому знаходяться рядки CSV-файлу, що містять інформацію про сектори діаграми, а також переміщення даних з об’єкту типу ArrayList в об’єкт типу String[][]. “CSVParseException ” – клас виключень, який викликається при утворені виключень пов’язаних з роботою над CSV-файлами, у ньому описані дії, які потрібно прийняти, для кожної виключної ситуації.

Спроектована структура програмного додатку зображена на рисунку 2.2.

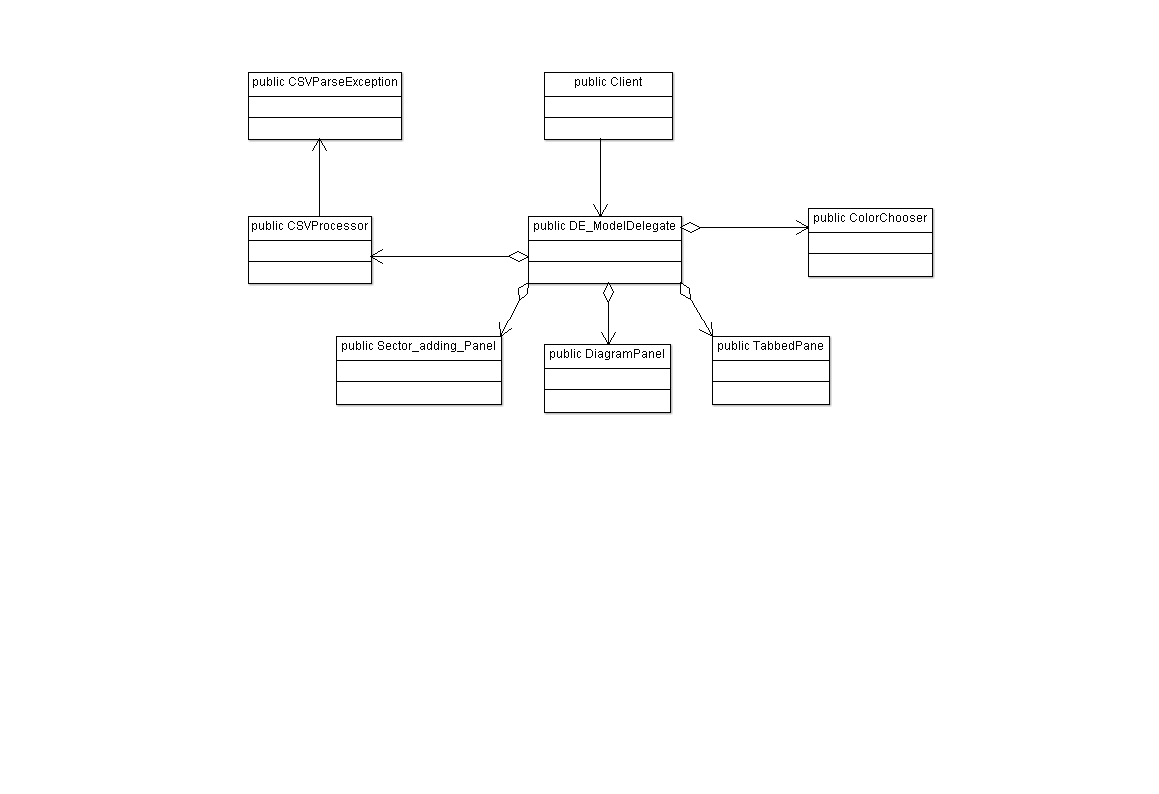


Рис. 2.2 Структура програмного додатку “Редактор кільцевих діаграм”

3. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Структура програмного додатку

Діаграма класів програмного додатку “Редактор кільцевих діаграм” створена для розуміння структури програми в цілому. На ній зображена відповідність всіх класів(див. Додаток В).

3.2 Інструкції для користувача

У програмному додатку “Редактор кільцевих діаграм” була розроблена логіка взаємодії з CSV-файлами, а саме: завантаження і вивантаження даних діаграми. Розробка проводилась на основі праць [3, c. 483-555; 2, c. 595-641; 6, c. 729-833]. В результаті це дозволило оперувати з CSV-файлами.

Також в програмному додатку присутній такий функціонал як: додавання, видалення, редагування секторів та очищення діаграми і, власне, створення нової діаграми. Написання логіки і візуального представлення для цих можливостей проводилось на основі джерел [1; 5, розділи 7-10; 6, розділи 6-7; 2, розділи 22-26,30,31].

Окрім цього, вищезгадані матеріали дали мені змогу побудувати весь графічний інтерфейс користувача. Для реалізації інтерфейсу користувача я віддав перевагу графічному пакету Swing для візуалізації програмного додатку та обробки подій, обумовлюючись тим, що елементи пакету java.swing є “облегшені” на відміну від java.awt, цей аспект доступно описаний в праці [2, c. 954].

Для того, щоб обробляти виключні ситуації у програмі, був реалізований клас виключень CSVParseException на основі книг [2, розділ 10; 3, розділ 13, 5, розділ 11] .

3.3 Специфікації класів

## C

[**ColorChooser**](file:///D:\--==--\Workspace\Kursova\doc\components\ColorChooser.html) - Клас в пакеті [components](file:///D:\--==--\Workspace\Kursova\doc\components\package-summary.html)

Клас, що реалізує вікно для вибору кольору сектора

[**ColorChooser()**](file:///D:\--==--\Workspace\Kursova\doc\components\ColorChooser.html#ColorChooser()) – Конструктор класу [ColorChooser](file:///D:\--==--\Workspace\Kursova\doc\components\ColorChooser.html)

Конструктор класу, в якому ініціалізується вікно

[**createAndShowGUI(JFrame)**](file:///D:\--==--\Workspace\Kursova\doc\logic\Sector_adding_Panel.html#createAndShowGUI(javax.swing.JFrame)) – Метод класу logic.[Sector\_adding\_Panel](file:///D:\--==--\Workspace\Kursova\doc\logic\Sector_adding_Panel.html)

Створення і відображення вікна

[**CSVParseException**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVParseException.html) – Клас виключень пакету [logic](file:///D:\--==--\Workspace\Kursova\doc\logic\package-summary.html)

Клас виключень, що викидається при помилці

[**CSVParseException()**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVParseException.html#CSVParseException()) – Конструктор для класу виключень logic.[CSVParseException](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVParseException.html)

Конструктор, реалізовує виведення помилки на екран

[**CSVProcessor**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html) – Класу пакету [logic](file:///D:\--==--\Workspace\Kursova\doc\logic\package-summary.html)

У класі CSVProcessor реалізувані методи завантаження/вивантаження CSV-файлу в/з агрегований об’єкт класу ArrayList, в якому знаходяться рядки CSV-файлу.

[**CSVProcessor(int)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#CSVProcessor(int)) – Конструктор для класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Конструктор, в якому задається значення буфера

## D

[**DE\_ModelDelegate**](file:///D:\--==--\Workspace\Kursova\doc\logic\DE_ModelDelegate.html) – Клас пакету [logic](file:///D:\--==--\Workspace\Kursova\doc\logic\package-summary.html)

Клас в якому реалізоване головне вікно

[**DE\_ModelDelegate()**](file:///D:\--==--\Workspace\Kursova\doc\logic\DE_ModelDelegate.html#DE_ModelDelegate()) – Конструктор класу logic.[DE\_ModelDelegate](file:///D:\--==--\Workspace\Kursova\doc\logic\DE_ModelDelegate.html)

Конструктор, де будується вікно програмного додатку

[**deserialization(File)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#deserialization(java.io.File)) - Метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод десеріалізує дані з файлу і повертає у вигляді об'єкту типу ArrayList

[**DiagramPanel**](file:///D:\--==--\Workspace\Kursova\doc\logic\DiagramPanel.html) – Клас пакету  [logic](file:///D:\--==--\Workspace\Kursova\doc\logic\package-summary.html)

Клас, в якому реалізована панель для рисування діаграми

[**DiagramPanel(int[], Color[])**](file:///D:\--==--\Workspace\Kursova\doc\logic\DiagramPanel.html#DiagramPanel(int[], java.awt.Color[])) Конструктор класу logic.[DiagramPanel](file:///D:\--==--\Workspace\Kursova\doc\logic\DiagramPanel.html)

Конструктор, який прорисовує діаграму на основі вхідних параметрів

## G

[**getBuf\_size()**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#getBuf_size()) – Статичний метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод повертає значення буфера

## L

[logic](file:///D:\--==--\Workspace\Kursova\doc\logic\package-summary.html) - пакет logic

пакет у якому описано логіку програми

## M

[**main(String[])**](file:///D:\--==--\Workspace\Kursova\doc\logic\TestIO.html#main(java.lang.String[]))  Статичний метод класу.[TestIO](file:///D:\--==--\Workspace\Kursova\doc\logic\TestIO.html)

## P

[**paint(Graphics)**](file:///D:\--==--\Workspace\Kursova\doc\logic\DiagramPanel.html#paint(java.awt.Graphics)) - метод класу logic.[DiagramPanel](file:///D:\--==--\Workspace\Kursova\doc\logic\DiagramPanel.html)

 Метод прописування діаграми

[**parse(ArrayList<String>)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#parse(java.util.ArrayList)) – Метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод зчитує з об'єкту типу ArrayList дані і записує їх в об'єкт типу String[][].

## R

[**read\_CSV(File)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#read_CSV(java.io.File)) – Метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод зчитує інформацію з CSV-файлу за допомогою потоків вводу пакету java.nio

## S

[**Sector\_adding\_Panel**](file:///D:\--==--\Workspace\Kursova\doc\logic\Sector_adding_Panel.html) - Class in [logic](file:///D:\--==--\Workspace\Kursova\doc\logic\package-summary.html)

Клас, що реалізує вікно додавання секторів

[**Sector\_adding\_Panel(int, JFrame)**](file:///D:\--==--\Workspace\Kursova\doc\logic\Sector_adding_Panel.html#Sector_adding_Panel(int, javax.swing.JFrame)) - Конструктор класу logic.[Sector\_adding\_Panel](file:///D:\--==--\Workspace\Kursova\doc\logic\Sector_adding_Panel.html)

Конструктор вікна для додавання секторів

[**serialization(ArrayList<String>, File)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#serialization(java.util.ArrayList, java.io.File)) - метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод серіалізує дані об'єкту array у файл

[**setBuf\_size(int)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#setBuf_size(int)) – Статичний метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

 Метод для задання розміру буфера

[**stateChanged(ChangeEvent)**](file:///D:\--==--\Workspace\Kursova\doc\components\ColorChooser.html#stateChanged(javax.swing.event.ChangeEvent)) - метод класу components.[ColorChooser](file:///D:\--==--\Workspace\Kursova\doc\components\ColorChooser.html)

Метод, який реагує на зміну вибору кольору у палітрі

## T

[**TabbedPane**](file:///D:\--==--\Workspace\Kursova\doc\components\TabbedPane.html) - Class in [components](file:///D:\--==--\Workspace\Kursova\doc\components\package-summary.html)

Клас, у якому реалізовано панель вкладок

[**TabbedPane()**](file:///D:\--==--\Workspace\Kursova\doc\components\TabbedPane.html#TabbedPane()) – Конструктор класу components.[TabbedPane](file:///D:\--==--\Workspace\Kursova\doc\components\TabbedPane.html)

Конструктор класу, в якому ініціалізується панель вкладок

## W

[**write\_CSV(ArrayList<String>, File)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#write_CSV(java.util.ArrayList, java.io.File)) - метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод записує дані з заданого масиву типу ArrayList в файл за допомогою потоків виводу пакету java.nio

[**write\_CSV\_modif(ArrayList<String[]>, File)**](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html#write_CSV_modif(java.util.ArrayList, java.io.File)) - метод класу logic.[CSVProcessor](file:///D:\--==--\Workspace\Kursova\doc\logic\CSVProcessor.html)

Метод записує дані з заданого масиву типу String[][] в файл за допомогою потоків виводу пакету java.nio

4. ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

Програмний додаток “Редактор кільцевих діаграм” реалізований згідно всіх вимог зазначених в технічному завдання і підтримує весь функціонал операцій, що перелічені у вищезгаданому розділі. Під час тестування програми не виявлено помилок чи зависань. На всіх етапах роботи вона працює коректно.

Під час запуску наше головне вікно має вигляд проілюстрований на рисунку 4.1.

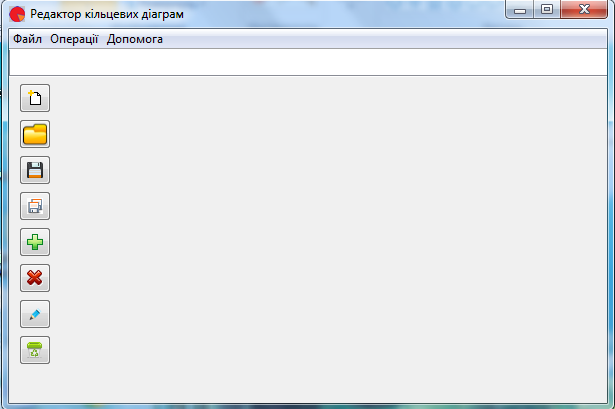


Рис. 4.1 Головне вікно після заспуску

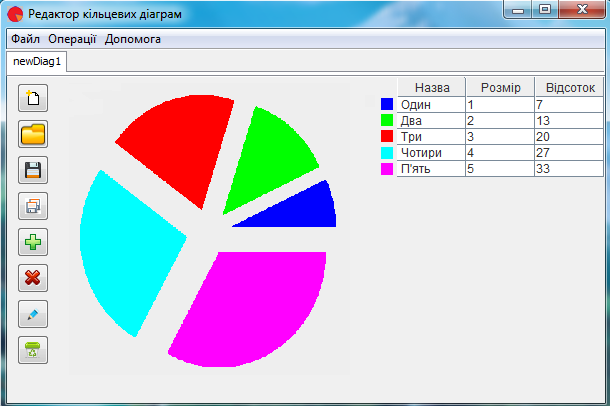
На рисунку 4.2 продемонстровано візуалізацію діаграми з п’ятьма секторами.

Рис. 4.2 Візуалізація діаграми з п’ятьма секторами

Операції відкриття, створення, збереження та маніпуляції з секторами можна проводити за допомогою кнопок лівої бокової панелі або через меню “Файл” та “Операції”

На рисунках 4.3, 4.4, 4.5 показаний вміст розділів меню.

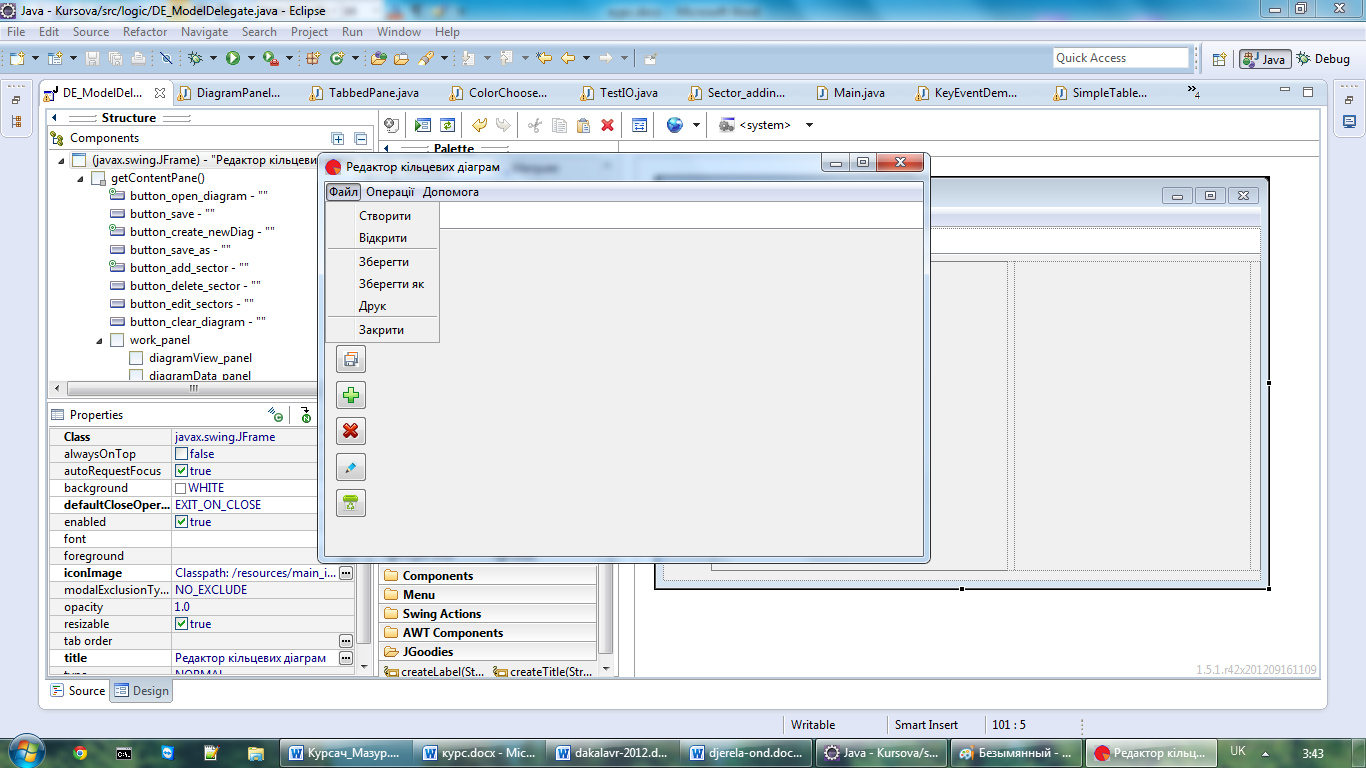


Рис. 4.3 Меню “Файл”

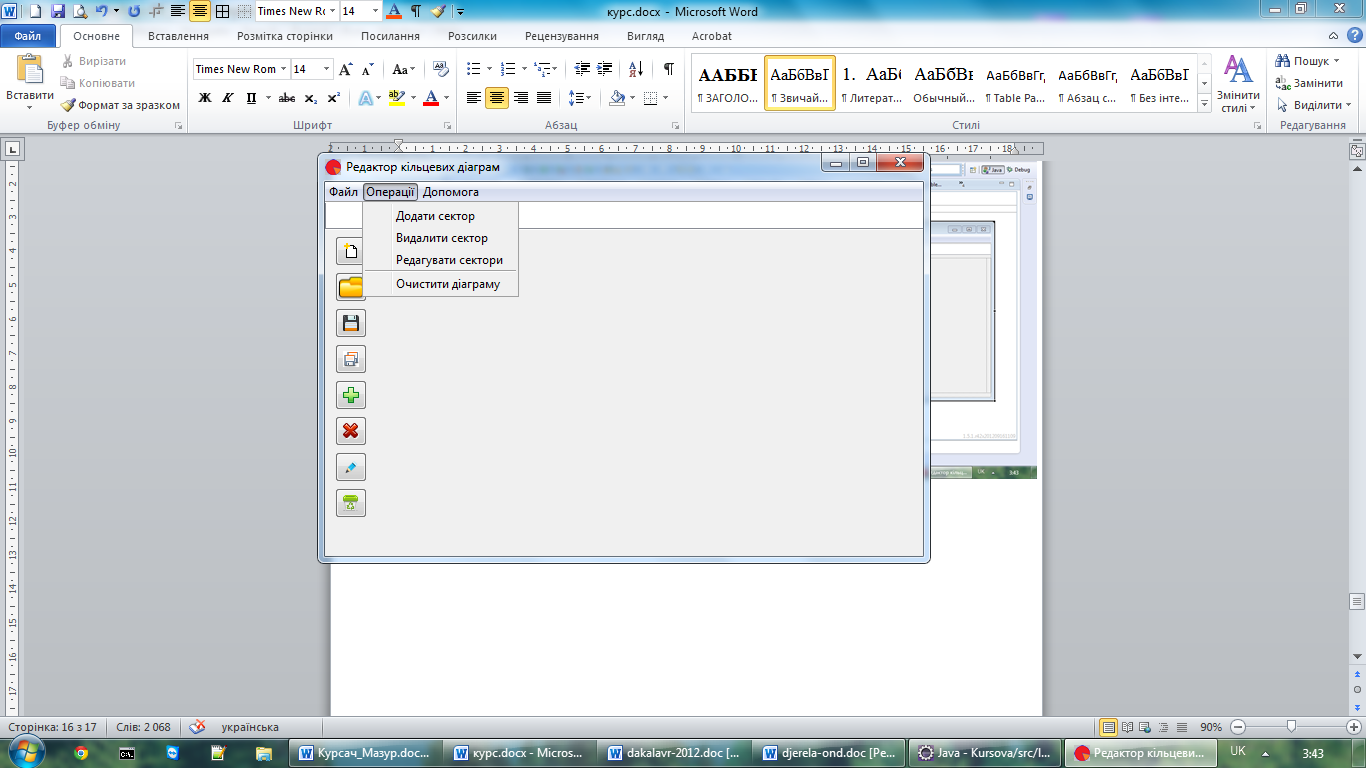


Рис. 4.4 Меню “Операції”

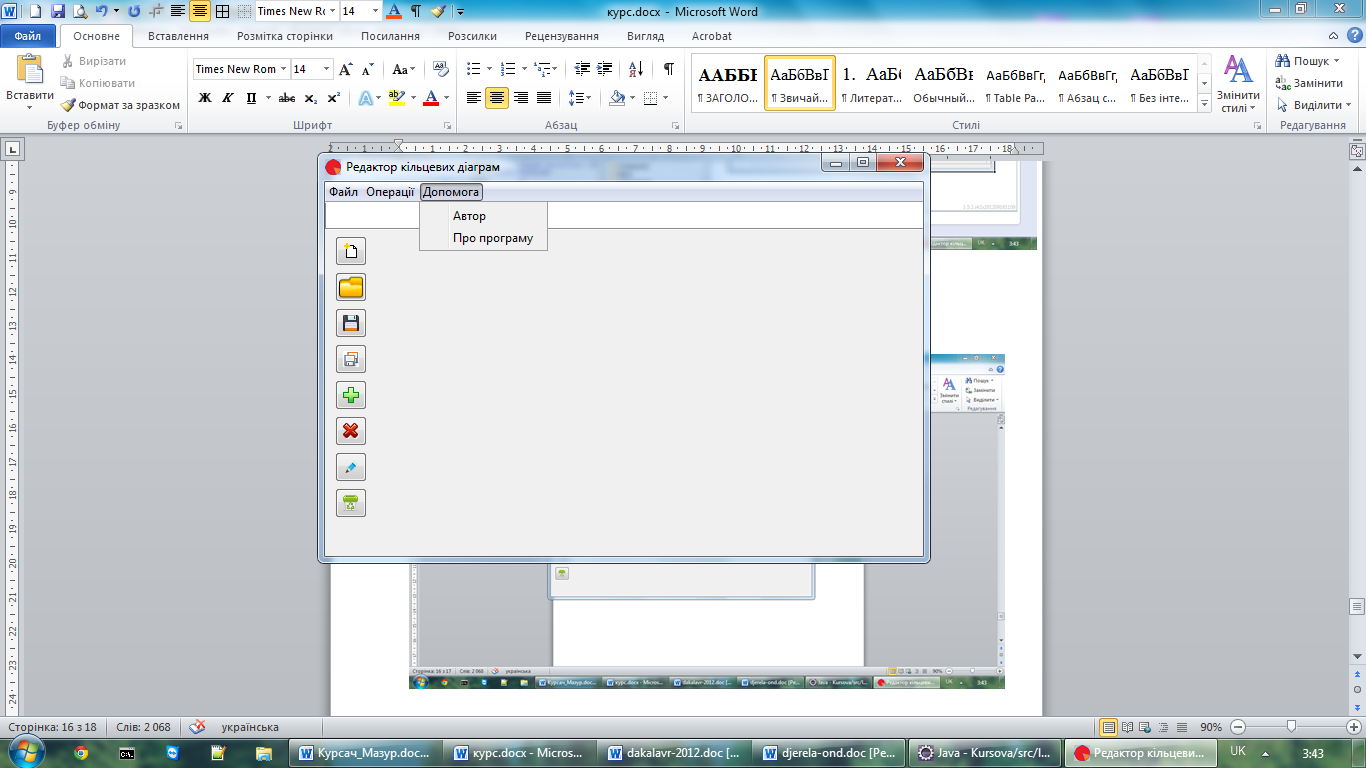


Рис. 4.5 Меню “Допомога”

На рисунку 4.6 зображено віконечко, що появляється, при нажатті на кнопку “Про програму”

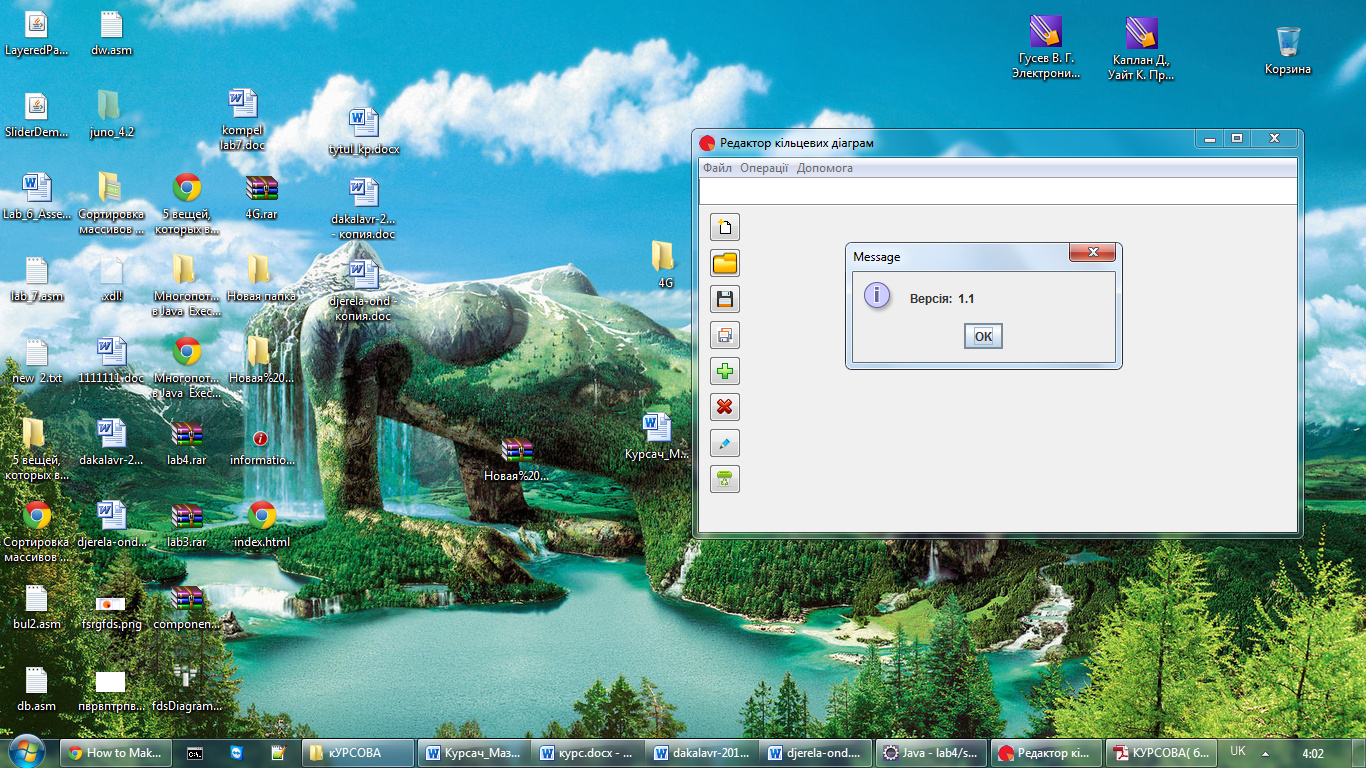


Рис. 4.6 Інформаційне вікно, що показує версію програми

На рисунку 4.7 зображено віконечко, що появляється, при нажатті на кнопку “Автор”

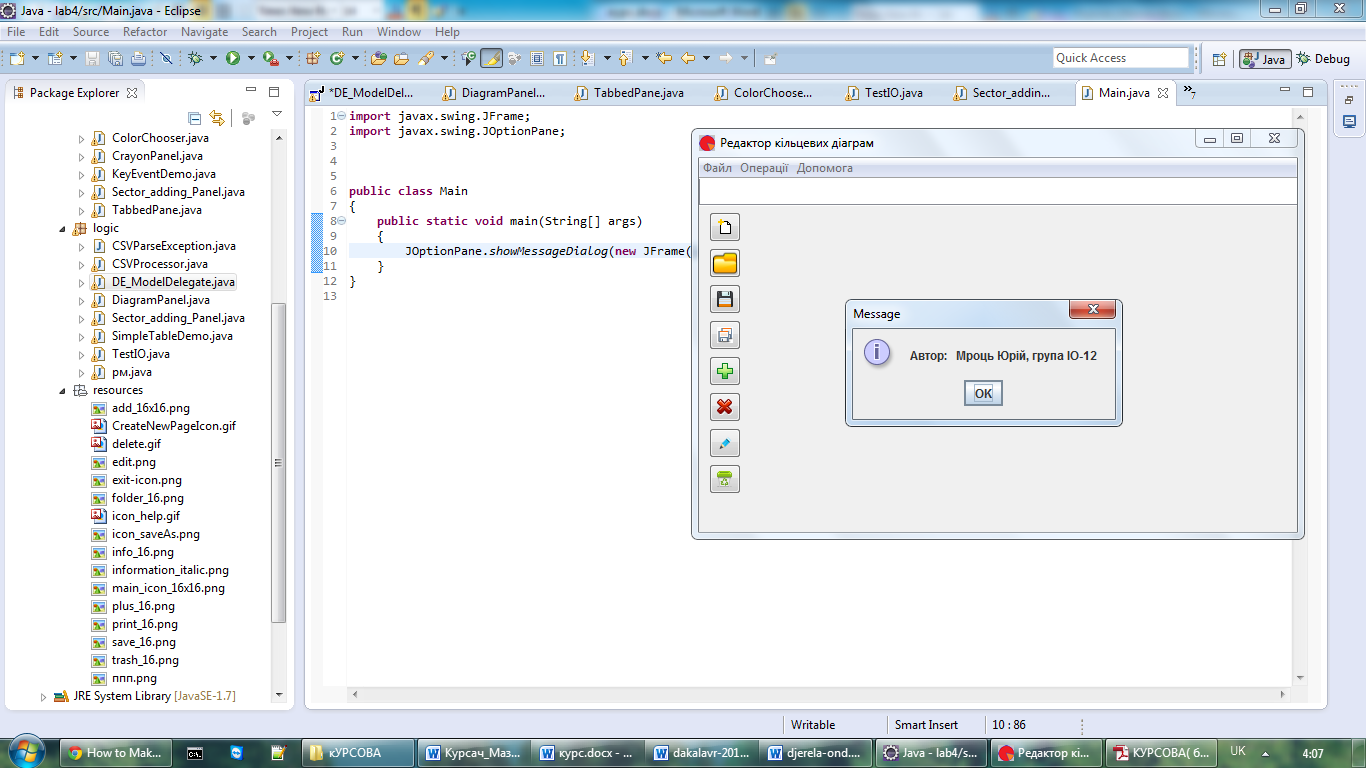


Рис. 4.7 Інформаційне вікно, що показує автора програми

На рисунку 4.8 продемонстроване діалогове вікно програми для відкриття збережених діаграм.

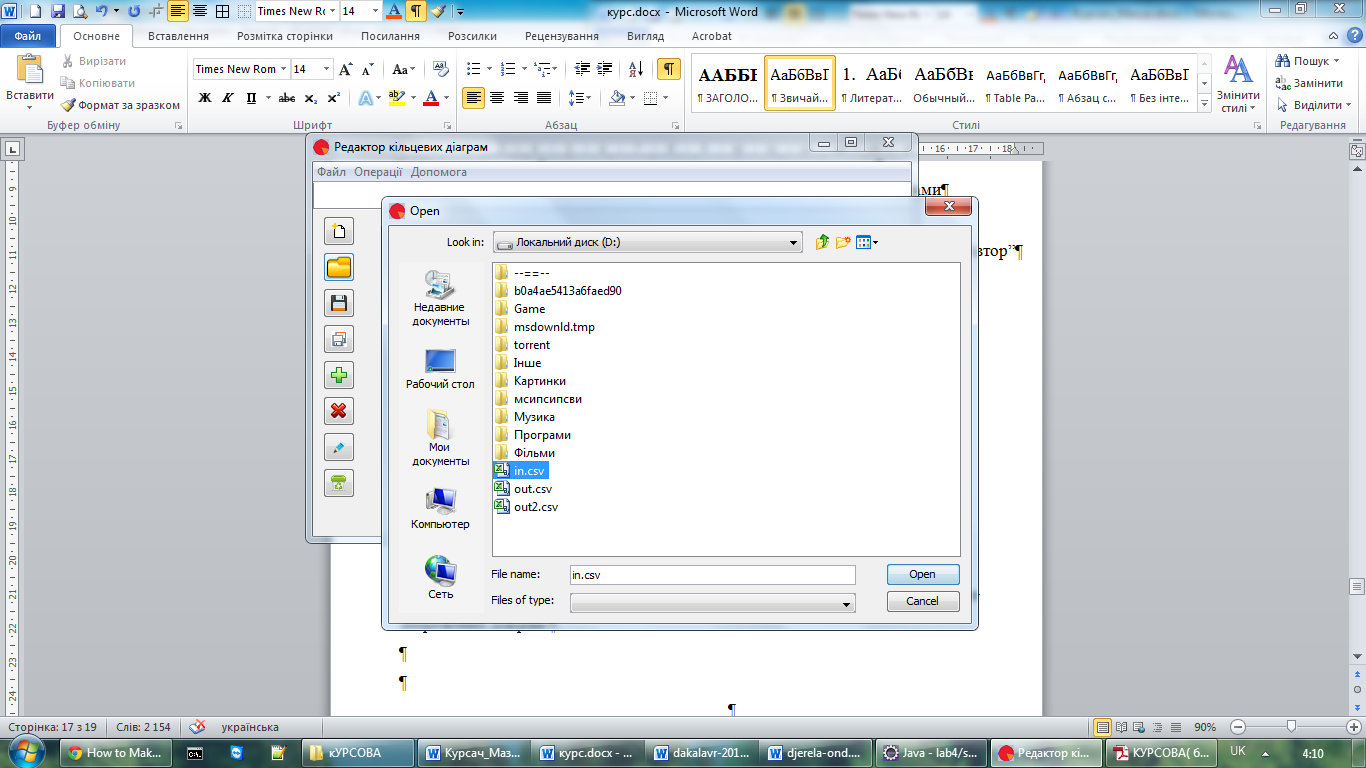


Рис. 4.8 Діалогове вікно відкриття файлу

На рисунку 4.9 зображено вікно вибору кольору для сектора, де можна вибрати колір в одній з п’яти палітр. Після вибору можна побачити в вікні Preview, як він буде дивитись в різних випадках.

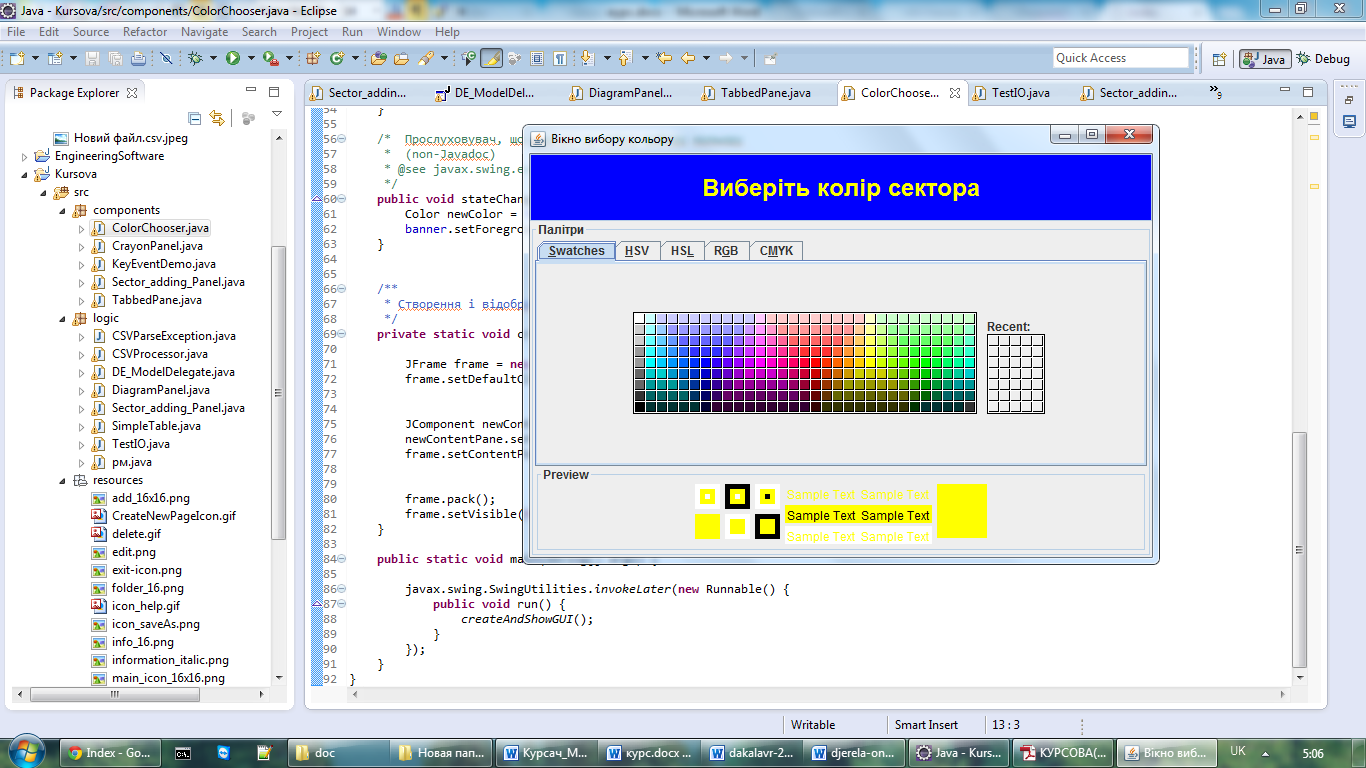


Рис. 4.8 Вікно вибору кольору

ВИСНОВКИ

Всі вимоги, які були зазначені щодо функціональності, реалізовані в програмному додатку в повній мірі. Інтерфейс користувача розроблений так, щоб можна було швидко і зручно оперувати над даними діаграми. В якості графічного пакету для реалізації інтерфейсу був обраний Swing, тому що його елементи менш ресурсозатратні в порівнянні з AWT. В процесі розробки програмного додатку, я ознайомився з основними класами пакетів Swing та AWT, за допомогою яких будуються графічні за стосунки, а також закріпив ці знання на практиці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Герберт Шилдт SWING: руководство для начинающих– М.:“Вильямс”, 2007. – С. 720. – ISBN 0-07-226314-8.
2. Герберт Шилдт Java. Полное руководство, 8-е изд. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 20012. – 1104 с. – ISBN 978-5-8459-1759-1 (рус.)
3. Эккель Б. Философия Java / Эккель Брюс; Пер.с англ. Е.Матвеев.– 4-е изд.–СПб.: Питер, 2010. – 640с.: ил. – (Библиотека программиста). – Алф.указ.:с.631. – ISBN 978-5-388-00003-3.
4. Приемы объектно-ориентированого проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. – СПб.: Питер, 2011 – 368 с.: ил. –ISBN 978-5-469-01136-1.
5. Хорстманн Кей С. Java 2. Том 1. Основы / Кей Хорстманн, Гари Корнелл; Пер с англ. – Изд. 8-е. – М.: ООО “И.ДВильямс”, 2011. – 816 c.: ил. – Парал. тит. англ. – (Библиотека профессионала). –ISBN 978-5-8459-1378-4 (рус.).
6. Хорстманн Кей С. Java 2. Том 2. Тонкости программирования / Кей Хорстманн, Гари Корнелл; Пер с англ. – Изд. 8-е. – М.: ООО “И.ДВильямс”, 2011. – 992 c.: ил. – Парал. тит. англ. – (Библиотека профессионала). –ISBN 978-5-8459-1482-8 (рус.).
7. Стелтинг Стивен Применение шаблонов Java /Стелтинг Стивен, Маасен Олав; Пер. с англ. –М.: Издательский дом “Вильямс”, 2002. – 576 c.: ил. – Парал. тит. англ. – (Библиотека профессионала). – ISBN 5-8459-0339-4 (рус.).
8. Дженифер Тидвелл Разработка пользовательских интрефейсов; Пер. с англ. –Е. Шикарева: Издательский дом “Питер”, 2008. – 416 c. – ISBN 978-5-91180-073-4 (рус.).
9. [А. К. Гультяев](http://www.ozon.ru/context/detail/id/5192567/#tab_person)  Проектирование и дизайн пользовательского интерфейса / В. А. Машин; Пер. с англ. : Издательский дом “ [Корона-Принт](http://www.ozon.ru/context/detail/id/856652/) ”, 2010. – 350 c. – ISBN 978-5-7931-0814-0 (рус.).

ДОДАТОК А. ЛІСТИНГ КОДУ ПРОГРАМНОГО ДОДАТКУ

**package logic;**

**import java.io.File;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**import java.io.ObjectInputStream;**

**import java.io.ObjectOutputStream;**

**import java.nio.ByteBuffer;**

**import java.nio.channels.FileChannel;**

**import java.nio.charset.Charset;**

**import java.util.ArrayList;**

**/\*\***

**\* У класі CSVProcessor реалізувані методи завантаження/вивантаження CSV-файлу в/з агрегований об’єкт класу ArrayList,**

**\* в якому знаходяться рядки CSV-файлу. Реалізувати методи серіалізації/десеріалізації агрегованого об’єкту класу ArrayList**

**\* в файлі serialized.dat.**

**\* @author Aruy**

**\*/**

**public class CSVProcessor {**

**/\*\***

**\* Розмір буфера.**

**\*/**

**private static int buf\_size;**

**/\*\***

**\* Конструктор**

**\* @param buf\_size розмір буфера.**

**\*/**

**public CSVProcessor(int buf\_size) {**

**setBuf\_size(buf\_size);**

**}**

**/\*\***

**\* Метод зчитує з об'єкту типу ArrayList<String> дані і записує їх в об'єкт типу String[][].**

**\* @param array об'єкт звідки треба зчитати дані**

**\* @return об'єкт, типу String[][]**

**\* @throws CSVParseException**

**\*/**

**public ArrayList<String[]> parse(ArrayList<String> array) throws CSVParseException{**

**ArrayList<String[]> mass = null;**

**int j = 0;**

**String k;**

**String[] tmp = new String[8];**

**for (int i = 0; i < array.size(); i++) {**

**k = array.get(i);**

**if (k != null) {**

**if (!k.equals("\n")) {**

**tmp[j] = k;**

**j++;**

**} else {**

**mass.add(tmp);**

**tmp = null;**

**j = 0;**

**}**

**}**

**}**

**return mass;**

**}**

**/\*\***

**\* Метод зчитує інформацію з CSV-файлу за допомогою потоків вводу пакету java.nio**

**\* @param f CSV-файл, звідки зчитуються дані**

**\* @return масив елементів типу String, що містяться в класі ArrayList**

**\* @throws IOException**

**\*/**

**public ArrayList<String> read\_CSV (File f) throws CSVParseException, IOException{**

**ArrayList<String> array = new ArrayList<String>();**

**FileChannel in\_channel = null;**

**ByteBuffer buf;**

**try {**

**in\_channel = new FileInputStream(f).getChannel();**

**buf = ByteBuffer.allocate(getBuf\_size());**

**while ((in\_channel.read(buf)) != -1 ) {**

**buf.flip();**

**for (int i = 0; i < buf.limit(); i++)**

**array.add((char)buf.get() + "");**

**buf.clear();**

**}**

**} catch(Exception e) {**

**throw new CSVParseException();**

**} finally {**

**if (in\_channel != null)**

**in\_channel.close();**

**}**

**return array;**

**}**

**/\*\***

**\* Метод записує дані з заданого масиву типу ArrayList<String> в файл за допомогою потоків виводу пакету java.nio**

**\* @param array масив даних, що записуватимуться в файл**

**\* @param f об'єкт типу File, що містить шлях до файлу, куди записуватимуться дані**

**\*/**

**public void write\_CSV (ArrayList<String> array, File f) {**

**FileChannel out\_channel = null;**

**ByteBuffer buf;**

**try {**

**out\_channel = new FileOutputStream(f).getChannel();**

**buf = ByteBuffer.allocate(getBuf\_size());**

**for (int i = 0; i < array.size(); i++) {**

**buf.put(array.get(i).getBytes(Charset.forName("utf-8")));**

**if (buf.capacity() == buf.position()) {**

**buf.flip();**

**out\_channel.write(buf);**

**buf.clear();**

**}**

**}**

**if (buf.position() > 0) {**

**buf.flip();**

**out\_channel.write(buf);**

**buf.clear();**

**}**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**} catch (IOException e) {**

**e.printStackTrace();**

**} finally {**

**try {**

**if (out\_channel != null)**

**out\_channel.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**/\*\***

**\* Метод записує дані з заданого масиву типу String[][] в файл за допомогою потоків виводу пакету java.nio**

**\* @param array масив даних, що записуватимуться в файл**

**\* @param f об'єкт типу File, що містить шлях до файлу, куди записуватимуться дані**

**\*/**

**public void write\_CSV\_modif (ArrayList<String[]> array, File f) {**

**FileChannel out\_channel = null;**

**ByteBuffer buf;**

**try {**

**out\_channel = new FileOutputStream(f).getChannel();**

**buf = ByteBuffer.allocate(getBuf\_size());**

**for (int i = 0; i < array.size(); i++) {**

**for (int j = 0; j < array.get(i).length; j++) {**

**byte[] k;**

**if ((k = array.get(i)[j].getBytes(Charset.forName("utf-8"))) == null) {**

**continue;**

**} else {**

**buf.put(k);**

**}**

**if (buf.capacity() == buf.position()) {**

**buf.flip();**

**out\_channel.write(buf);**

**buf.clear();**

**}**

**}**

**}**

**if (buf.position() > 0) {**

**buf.flip();**

**out\_channel.write(buf);**

**buf.clear();**

**}**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**} catch (IOException e) {**

**e.printStackTrace();**

**} finally {**

**try {**

**if (out\_channel != null)**

**out\_channel.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**/\*\***

**\* @return the buf\_size**

**\*/**

**public static int getBuf\_size() {**

**return buf\_size;**

**}**

**/\*\***

**\* @param buf\_size the buf\_size to set**

**\*/**

**public static void setBuf\_size(int buf\_size) {**

**CSVProcessor.buf\_size = buf\_size;**

**}**

**/\*\***

**\* Метод серіалізує дані об'єкту array у файл**

**\* @param array об'єкт, що серіалізуватиметься**

**\* @param f об'єкт типу File, що містить шлях до файлу, в який серіалізується об'єкт**

**\*/**

**public void serialization (ArrayList<String> array, File f) {**

**ObjectOutputStream oos = null;**

**try {**

**oos = new ObjectOutputStream(new FileOutputStream(f));**

**oos.writeObject(array);**

**} catch (IOException e) {**

**e.printStackTrace();**

**} finally {**

**try {**

**if (oos != null)**

**oos.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**/\*\***

**\* Метод десеріалізує дані з файлу і повертає у вигляді об'єкту типу ArrayList<String>**

**\* @param f об'єкт типу File, що містить шлях до файлу, в якому міститься серіалізований об'єкт**

**\* @return десеріалізований об'єкт типу ArrayList<String>**

**\*/**

**@SuppressWarnings("unchecked")**

**public ArrayList<String> deserialization (File f) {**

**ArrayList<String> array = null;**

**ObjectInputStream ois = null;**

**try {**

**ois = new ObjectInputStream(new FileInputStream(f));**

**array = (ArrayList<String>) ois.readObject();**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**} catch (IOException e) {**

**e.printStackTrace();**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**} finally {**

**try {**

**if (ois != null)**

**ois.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**return array;**

**}**

**}**

**package** logic;

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.UIManager;

**import** java.awt.Toolkit;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuItem;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** javax.swing.JButton;

**import** javax.swing.JFileChooser;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTabbedPane;

**import** javax.swing.GroupLayout;

**import** javax.swing.JTable;

**import** javax.swing.GroupLayout.Alignment;

**import** javax.swing.LayoutStyle.ComponentPlacement;

**import** javax.swing.ImageIcon;

**import** javax.swing.filechooser.FileNameExtensionFilter;

**import** java.awt.event.MouseAdapter;

**import** java.awt.event.MouseEvent;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** javax.swing.JLabel;

**import** javax.swing.JSeparator;

**import** javax.swing.JTextField;

**import** java.awt.event.KeyAdapter;

**import** java.awt.event.KeyEvent;

**import** javax.swing.JInternalFrame;

**import** java.awt.GridLayout;

**import** javax.swing.JList;

**public** **class** DE\_ModelDelegate **extends** JFrame {

**private** **static** **final** **long** *serialVersionUID* = 1L;

/\*\*

\* Об'єкт, що представлює собою меню "Файл"

\*/

**private** JMenu menu;

/\*\*

\* Об'єкт, що представлює собою меню "Операції"

\*/

**private** JMenu operations;

/\*\*

\* Об'єкт, що представлює собою меню "Допомога"

\*/

**private** JMenu help;

/\*\*

\* Об'єкт, що представлює собою панель меню

\*/

**private** JMenuBar menuBar;

/\*\*

\* Об'єкт, що представлює собою кнопку "Створити"

\*/

**private** JMenuItem create;

/\*\*

\* Об'єкт, що представлює собою кнопку "Відкрити"

\*/

**private** JMenuItem open;

/\*\*

\* Об'єкт, що представлює собою кнопку "Зберегти"

\*/

**private** JMenuItem save;

/\*\*

\* Об'єкт, що представлює собою кнопку "Зберегти як"

\*/

**private** JMenuItem save\_as;

/\*\*

\* Об'єкт, що представлює собою кнопку на бічній панелі "Створити"

\*/

**private** JButton button\_create\_newDiag;

/\*\*

\* Об'єкт, що представлює собою кнопку на бічній панелі "Відкрити"

\*/

**private** JButton button\_open\_diagram;

/\*\*

\* Об'єкт, що представлює собою кнопку на бічній панелі "Зберегти"

\*/

**private** JButton button\_save;

/\*\*

\* Об'єкт, що представлює собою кнопку на бічній панелі "Додати сектор"

\*/

**private** JButton button\_add\_sector;

/\*\*

\* Об'єкт, що представлює собою кнопку "Видалити сектор"

\*/

**private** JButton button\_delete\_sector;

/\*\*

\* Об'єкт, що представлює собою кнопку "Очистити діаграму"

\*/

**private** JButton button\_clear\_diagram;

/\*\*

\* Об'єкт, що представлює собою кнопку "Редагувати діаграму"

\*/

**private** JButton button\_edit\_sectors;

/\*\*

\* Об'єкт, що представлює собою панель вкладок

\*/

**private** JTabbedPane tabbedPane;

/\*\*

\* Об'єкт, що представлює собою діалогове вікно для вікдриття файлу

\*/

**private** JFileChooser fileChooser;

/\*\*

\* Об'єкт, що представлює собою фільтр розширень, що допустимі для відкриття

\*/

**private** FileNameExtensionFilter filter;

/\*\*

\* Об'єкт класу CSVProcessor

\*/

**private** CSVProcessor csvProcessor;

/\*\*

\* Об'єкт, що представлює собою сховище даних секторів

\*/

**private** ArrayList<String> diagramData;

/\*\*

\* Об'єкт, що представлює собою зону відображення діаграми

\*/

**private** JPanel work\_panel;

/\*\*

\* Об'єкт, що представлює собою зону відображення списку секторів

\*/

**private** JPanel diagramData\_panel;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem\_1;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem\_2;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem\_3;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem\_4;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem\_5;

/\*\*

\* Об'єкт, що представлює собою пункт меню

\*/

**private** JMenuItem menuItem\_6;

/\*\*

\* Об'єкт, що представлює собою розділювач в меню

\*/

**private** JSeparator separator\_3;

/\*\*

\* Об'єкт, що відповідає за відображення діаграми

\*/

**private** JPanel DiagramPanel;

/\*\*

\* Об'єкт, що відображає віконечко додавання секторів

\*/

**private** JPanel Sector\_adding\_Panel;

/\*\*

\* лічильник діаграм

\*/

**private** **int** diagram\_count = 1;

/\*\*

\* Зона, де відображається рисунок діаграми

\*/

**private** JPanel diagramView\_panel;

**public** **static** **void** main(String[] args) {

**try** {

UIManager.*setLookAndFeel*(UIManager.*getSystemLookAndFeelClassName*());

} **catch** (Throwable e) {

e.printStackTrace();

}

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

DE\_ModelDelegate frame = **new** DE\_ModelDelegate();

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Конструктор, де будується вікно програмного додатку

\*/

**public** DE\_ModelDelegate() {

setBackground(Color.*WHITE*);

setIconImage(Toolkit.*getDefaultToolkit*().getImage(DE\_ModelDelegate.**class**.getResource("/resources/main\_icon\_16x16.png")));

setFont(**null**);

setTitle("Редактор кільцевих діаграм");

setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

setBounds(100, 100, 614, 412);

csvProcessor = **new** CSVProcessor(1024);

menuBar = **new** JMenuBar();

menuBar.setFont(**new** Font("Segoe UI", Font.*PLAIN*, 12));

menuBar.setBackground(Color.*WHITE*);

setJMenuBar(menuBar);

menu = **new** JMenu("Файл");

menuBar.add(menu);

create = **new** JMenuItem("Створити");

create.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mousePressed(MouseEvent e) {

tabbedPane.addTab("newDiag" + Integer.*toString*(diagram\_count++), **new** JPanel());

diagramData\_panel = **new** Sector\_adding\_Panel(0, **new** DE\_ModelDelegate());

}

});

menu.add(create);

open = **new** JMenuItem("Відкрити");

menu.add(open);

JSeparator separator\_1 = **new** JSeparator();

menu.add(separator\_1);

save = **new** JMenuItem("Зберегти");

menu.add(save);

save\_as = **new** JMenuItem("Зберегти як");

menu.add(save\_as);

JSeparator separator\_2 = **new** JSeparator();

menu.add(separator\_2);

menuItem = **new** JMenuItem("Закрити");

menu.add(menuItem);

operations = **new** JMenu("Операції");

menuBar.add(operations);

menuItem\_1 = **new** JMenuItem("Додати сектор");

operations.add(menuItem\_1);

menuItem\_2 = **new** JMenuItem("Видалити сектор");

operations.add(menuItem\_2);

menuItem\_4 = **new** JMenuItem("Редагувати сектори");

operations.add(menuItem\_4);

separator\_3 = **new** JSeparator();

operations.add(separator\_3);

menuItem\_3 = **new** JMenuItem("Очистити діаграму");

operations.add(menuItem\_3);

help = **new** JMenu("Допомога");

menuBar.add(help);

menuItem\_6 = **new** JMenuItem("Автор");

help.add(menuItem\_6);

menuItem\_5 = **new** JMenuItem("Про програму");

help.add(menuItem\_5);

tabbedPane = **new** JTabbedPane(JTabbedPane.*TOP*);

work\_panel = **new** JPanel();

button\_create\_newDiag = **new** JButton("");

button\_create\_newDiag.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent arg0) {

tabbedPane.addTab("newDiag" + Integer.*toString*(diagram\_count++), **new** JPanel());

}

});

button\_create\_newDiag.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/CreateNewPageIcon.gif")));

button\_open\_diagram = **new** JButton("");

button\_open\_diagram.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/\u043F\u043F\u043F.png")));

button\_open\_diagram.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

fileChooser = **new** JFileChooser();

filter = **new** FileNameExtensionFilter(**null**, "csv");

fileChooser.setFileFilter(filter);

**int** returnVal = fileChooser.showOpenDialog(DE\_ModelDelegate.**this**);

**if**(returnVal == JFileChooser.*APPROVE\_OPTION*) {

**try** {

diagramData = csvProcessor.read\_CSV(fileChooser.getSelectedFile());

diagram\_count++;

tabbedPane.addTab("newDiag" + Integer.*toString*(diagram\_count++), **new** JPanel());

JLabel lblNewLabel = **new** JLabel(diagramData.get(0));

work\_panel.add(lblNewLabel);

} **catch** (CSVParseException | IOException e1) {

e1.printStackTrace();

}

}

}

});

button\_save = **new** JButton("");

button\_save.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/save\_16.png")));

button\_add\_sector = **new** JButton("");

button\_add\_sector.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mousePressed(MouseEvent arg0) {

}

});

button\_add\_sector.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/add\_16x16.png")));

button\_delete\_sector = **new** JButton("");

button\_delete\_sector.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/delete.gif")));

button\_clear\_diagram = **new** JButton("");

button\_clear\_diagram.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/trash\_16.png")));

button\_edit\_sectors = **new** JButton("");

button\_edit\_sectors.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/edit.png")));

JButton button\_save\_as = **new** JButton("");

button\_save\_as.setIcon(**new** ImageIcon(DE\_ModelDelegate.**class**.getResource("/resources/icon\_saveAs.png")));

GroupLayout groupLayout = **new** GroupLayout(getContentPane());

groupLayout.setHorizontalGroup(

groupLayout.createParallelGroup(Alignment.*LEADING*)

.addGroup(groupLayout.createSequentialGroup()

.addContainerGap()

.addGroup(groupLayout.createParallelGroup(Alignment.*LEADING*)

.addComponent(button\_open\_diagram, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_save, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_create\_newDiag, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_save\_as, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_add\_sector, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_delete\_sector, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_edit\_sectors, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*)

.addComponent(button\_clear\_diagram, GroupLayout.*PREFERRED\_SIZE*, 32, GroupLayout.*PREFERRED\_SIZE*))

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(work\_panel, GroupLayout.*PREFERRED\_SIZE*, 550, Short.*MAX\_VALUE*)

.addGap(48))

.addComponent(tabbedPane, Alignment.*TRAILING*, GroupLayout.*DEFAULT\_SIZE*, 646, Short.*MAX\_VALUE*)

);

groupLayout.setVerticalGroup(

groupLayout.createParallelGroup(Alignment.*LEADING*)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(tabbedPane, GroupLayout.*PREFERRED\_SIZE*, 27, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addGroup(groupLayout.createParallelGroup(Alignment.*LEADING*)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(button\_create\_newDiag, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_open\_diagram, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_save, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_save\_as, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_add\_sector, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_delete\_sector, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_edit\_sectors, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(button\_clear\_diagram, GroupLayout.*PREFERRED\_SIZE*, 30, GroupLayout.*PREFERRED\_SIZE*))

.addComponent(work\_panel, GroupLayout.*PREFERRED\_SIZE*, 310, Short.*MAX\_VALUE*))

.addGap(39))

);

diagramView\_panel = **new** JPanel();

diagramData\_panel = **new** JPanel();

GroupLayout gl\_work\_panel = **new** GroupLayout(work\_panel);

gl\_work\_panel.setHorizontalGroup(

gl\_work\_panel.createParallelGroup(Alignment.*TRAILING*)

.addGroup(gl\_work\_panel.createSequentialGroup()

.addComponent(diagramView\_panel, GroupLayout.*DEFAULT\_SIZE*, 297, Short.*MAX\_VALUE*)

.addPreferredGap(ComponentPlacement.*RELATED*)

.addComponent(diagramData\_panel, GroupLayout.*PREFERRED\_SIZE*, 237, GroupLayout.*PREFERRED\_SIZE*)

.addContainerGap())

);

gl\_work\_panel.setVerticalGroup(

gl\_work\_panel.createParallelGroup(Alignment.*LEADING*)

.addGroup(gl\_work\_panel.createSequentialGroup()

.addGroup(gl\_work\_panel.createParallelGroup(Alignment.*LEADING*)

.addComponent(diagramView\_panel, GroupLayout.*DEFAULT\_SIZE*, 310, Short.*MAX\_VALUE*)

.addComponent(diagramData\_panel, GroupLayout.*PREFERRED\_SIZE*, 310, GroupLayout.*PREFERRED\_SIZE*))

.addContainerGap())

);

diagramData\_panel.setLayout(**new** GridLayout(0, 1, 0, 0));

work\_panel.setLayout(gl\_work\_panel);

getContentPane().setLayout(groupLayout);

}

}

**package** logic;

/\*\*

\* Клас виключення.

\* **@author** Aruy

\*/

/\*\*

\* Клас виключень, що викидається при помилці

\* **@author** Aruy

\*

\*/

**public** **class** CSVParseException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** *serialVersionUID* = 1L;

/\*\*

\* Конструктор, реалізовує виведення помилки на екран

\*/

**public** CSVParseException() {

System.*err*.println ("Помилка CSVParseException");

}

}

package logic;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

public class DiagramPanel extends JPanel

{

/\*\*

\* Кількість сеткорів в діаграмі

\*/

private int[] sectors\_values;

/\*\*

\* Кольори діаграми

\*/

private Color[] colors;

/\*\*

\* Конструктор, що відповідає за рисування діаграми

\* @param sec\_val кількість секторів

\* @param color\_val масив кольорів

\*/

public DiagramPanel(int[] sec\_val, Color[] color\_val) {

this.sectors\_values = new int[sec\_val.length];

this.colors = new Color[color\_val.length];

this.sectors\_values = sec\_val;

this.colors = color\_val;

}

/\* Метод, що прорисовує діарграму

\* (non-Javadoc)

\* @see javax.swing.JComponent#paint(java.awt.Graphics)

\*/

public void paint(Graphics g) {

int sum = 0;

for (int i = 0; i < colors.length; i++) {

g.setColor(colors[i]);

g.fillRect(10, 24 + i\*26, 20, 20);

g.fillArc((int)(150 + Math.round(30\*Math.cos(Math.toRadians((2\*sum + sectors\_values[i])/2)))),

(int)(150 - Math.round(30\*Math.sin(Math.toRadians((2\*sum + sectors\_values[i])/2)))), 300, 300, sum, sectors\_values[i]);

sum +=sectors\_values[i];

}

}

}

package logic;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import java.awt.Dimension;

import java.awt.GridLayout;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* Клас, що реалізує вікно додавання секторів

\* @author Aruy

\*

\*/

public class Sector\_adding\_Panel extends JPanel {

/\*\*

\* Конструктор вікна для додавання секторів

\* @param num\_of\_sect кількість секторів

\* @param frame батьківське вікно

\*/

public Sector\_adding\_Panel(int num\_of\_sect, JFrame frame) {

super(new GridLayout(1,0));

String[] columnNames = {"Колір",

"Назва",

"Розмір",

"Відсоток" };

Object[][] data = new Object[num\_of\_sect][4];

final JTable table = new JTable(data, columnNames);

table.setPreferredScrollableViewportSize(new Dimension(500, 70));

table.setFillsViewportHeight(true);

JScrollPane scrollPane = new JScrollPane(table);

.

add(scrollPane);

createAndShowGUI(frame);

}

/\*\*

\* Створення і відображення вікна

\*/

public void createAndShowGUI(JFrame frame) {

Sector\_adding\_Panel newContentPane = new Sector\_adding\_Panel(4, frame);

newContentPane.setOpaque(true); frame.setContentPane(newContentPane);

frame.pack();

frame.setVisible(true);

}

}

package logic;

import javax.swing.JTabbedPane;

import javax.swing.ImageIcon;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JFrame;

import javax.swing.JComponent;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.GridLayout;

import java.awt.event.KeyEvent;

public class TabbedPane extends JPanel {

public TabbedPane() {

super(new GridLayout(1, 1));

JTabbedPane tabbedPane = new JTabbedPane();

ImageIcon icon = createImageIcon("images/middle.gif");

JComponent panel1 = makeTextPanel("Panel #1");

tabbedPane.addTab("Tab 1", icon, panel1,

"Does nothing");

tabbedPane.setMnemonicAt(0, KeyEvent.VK\_1);

JComponent panel2 = makeTextPanel("Panel #2");

tabbedPane.addTab("Tab 2", icon, panel2,

"Does twice as much nothing");

tabbedPane.setMnemonicAt(1, KeyEvent.VK\_2);

JComponent panel3 = makeTextPanel("Panel #3");

tabbedPane.addTab("Tab 3", icon, panel3,

"Still does nothing");

tabbedPane.setMnemonicAt(2, KeyEvent.VK\_3);

JComponent panel4 = makeTextPanel(

"Panel #4 (has a preferred size of 410 x 50).");

panel4.setPreferredSize(new Dimension(410, 50));

tabbedPane.addTab("Tab 4", icon, panel4,

"Does nothing at all");

tabbedPane.setMnemonicAt(3, KeyEvent.VK\_4);

add(tabbedPane);

tabbedPane.setTabLayoutPolicy(JTabbedPane.SCROLL\_TAB\_LAYOUT);

}

/\*\*

\* Створення горизонтальної панелі

\* @param text

\* @return

\*/

protected JComponent makeTextPanel(String text) {

JPanel panel = new JPanel(false);

JLabel filler = new JLabel(text);

filler.setHorizontalAlignment(JLabel.CENTER);

panel.setLayout(new GridLayout(1, 1));

panel.add(filler);

return panel;

}

/\*\*

\* Загружає іконку по заданому шляху

\* @param path шлях, за яким знаходиться іконка

\* @return повертає об'єкт типу ImageIcon

\*/

protected static ImageIcon createImageIcon(String path) {

java.net.URL imgURL = TabbedPane.class.getResource(path);

if (imgURL != null) {

return new ImageIcon(imgURL);

} else {

System.err.println("Couldn't find file: " + path);

return null;

}

}

/\*\*

\* Створення панелі вкладок для головного вікна

\*/

private static void createAndShowGUI() {

JFrame frame = new JFrame("TabbedPaneDemo");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.add(new TabbedPane(), BorderLayout.CENTER);

frame.pack();

frame.setVisible(true);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

UIManager.put("swing.boldMetal", Boolean.FALSE);

createAndShowGUI();

}

});

}

}

package components;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import javax.swing.colorchooser.\*;

public class ColorChooser extends JPanel

implements ActionListener,

ChangeListener {

public JLabel banner;

public JColorChooser tcc;

public ColorChooser() {

super(new BorderLayout());

banner = new JLabel("Welcome to the Tutorial Zone!",

JLabel.CENTER);

banner.setForeground(Color.yellow);

banner.setBackground(Color.blue);

banner.setOpaque(true);

banner.setFont(new Font("SansSerif", Font.BOLD, 24));

banner.setPreferredSize(new Dimension(100, 65));

JPanel bannerPanel = new JPanel(new BorderLayout());

bannerPanel.add(banner, BorderLayout.CENTER);

bannerPanel.setBorder(BorderFactory.createTitledBorder("Banner"));

JPanel panel = new JPanel(); //use FlowLayout

JButton bcc = new JButton("Show Color Chooser...");

bcc.addActionListener(this);

panel.add(bcc);

panel.setBorder(BorderFactory.createTitledBorder(

"Choose Background Color"));

tcc = new JColorChooser();

tcc.getSelectionModel().addChangeListener(this);

tcc.setBorder(BorderFactory.createTitledBorder("Choose Text Color"));

tcc.setPreviewPanel(new JPanel());

AbstractColorChooserPanel panels[] = { new CrayonPanel() };

tcc.setChooserPanels(panels);

tcc.setColor(banner.getForeground());

add(bannerPanel, BorderLayout.PAGE\_START);

add(panel, BorderLayout.CENTER);

add(tcc, BorderLayout.PAGE\_END);

}

public void actionPerformed(ActionEvent e) {

Color newColor = JColorChooser.showDialog(

ColorChooser.this,

"Choose Background Color",

banner.getBackground());

if (newColor != null) {

banner.setBackground(newColor);

}

}

public void stateChanged(ChangeEvent e) {

Color newColor = tcc.getColor();

banner.setForeground(newColor);

}

/\*\*

\* Create the GUI and show it. For thread safety,

\* this method should be invoked from the

\* event-dispatching thread.

\*/

private static void createAndShowGUI() {

JFrame frame = new JFrame("ColorChooserDemo2");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JComponent newContentPane = new ColorChooser();

newContentPane.setOpaque(true); //content panes must be opaque

frame.setContentPane(newContentPane);

frame.pack();

frame.setVisible(true);

}

public static void main(String[] args) {

javax.swing.SwingUtilities.invokeLater(new Runnable() {

public void run() {

createAndShowGUI();

}

});

}

}

package components;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import javax.swing.colorchooser.\*;

/\*\*

\* Клас, що реалізовує вікно для вибору кольору для сектора

\* @author Aruy

\*

\*/

public class ColorChooser extends JPanel

implements ChangeListener {

/\*\*

\* Об'єкт, що забезпечую механізм вибору кольору

\*/

protected JColorChooser tcc;

/\*\*

\* Заголовок над палітрою

\*/

protected JLabel banner;

/\*\*

\* Конструктор класу

\*/

public ColorChooser() {

super(new BorderLayout());

banner = new JLabel("Виберіть колір сектора",

JLabel.CENTER);

banner.setForeground(Color.yellow);

banner.setBackground(Color.blue);

banner.setOpaque(true);

banner.setFont(new Font("SansSerif", Font.BOLD, 24));

banner.setPreferredSize(new Dimension(100, 65));

JPanel bannerPanel = new JPanel(new BorderLayout());

bannerPanel.add(banner, BorderLayout.CENTER);

bannerPanel.setBorder(BorderFactory.createTitledBorder(""));

tcc = new JColorChooser(banner.getForeground());

tcc.getSelectionModel().addChangeListener(this);

tcc.setBorder(BorderFactory.createTitledBorder(

"Палітри"));

add(bannerPanel, BorderLayout.CENTER);

add(tcc, BorderLayout.PAGE\_END);

}

/\* Прослуховувач, що реагує на зміну вибору кольору

\* (non-Javadoc)

\* @see javax.swing.event.ChangeListener#stateChanged(javax.swing.event.ChangeEvent)

\*/

public void stateChanged(ChangeEvent e) {

Color newColor = tcc.getColor();

banner.setForeground(newColor);

}

/\*\*

\* Створення і відображення вікна

\*/

private static void createAndShowGUI() {

JFrame frame = new JFrame("Вікно вибору кольору");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JComponent newContentPane = new ColorChooser();

newContentPane.setOpaque(true);

frame.setContentPane(newContentPane);

frame.pack();

frame.setVisible(true);

}

Додаток Б. Таблиця відповідності елементів дизайну

Таблиця 2.1. Відповідність елементів інтерфейсу до класів

|  |  |
| --- | --- |
| ***Елемент інтерфейсу*** | ***Класи пакету java.swing*** |
| Головне вікно | java.swing .JFrame |
| Панель «Меню» | java.swing .JMenuBar |
| Панель «Вкладки» | java.swing .JTabbedPane |
| Область відображення діаграми | java.swing .JPanel |
| Діаграма | java.awt.Graphics |
| Список секторів | java.swing .JTable |
| Кнопка «Файл» | java.swing .JMenu |
| Кнопка «Операції» | java.swing .JMenu |
| Кнопка «Про програму» | java.swing .JMenu |
| Кнопка «Новий файл» | java.swing .JMenuItem |
| Кнопка «Зберегти файл» | java.swing .JMenuItem |
| Кнопка «Зберегти як…» | java.swing .JMenuItem |
| Кнопка «Відкрити файл» | java.swing .JMenuItem |
| Кнопка «Закрити файл» | java.swing .JMenuItem |
| Кнопка «Додати сектор» | java.swing .JMenuItem |
| Кнопка «Видалити сектор» | java.swing .JMenuItem |
| Кнопка «Змінити параметри сектора» | java.swing .JMenuItem |
| Кнопка «Очистити діаграму» | java.swing .JMenuItem |
| Кнопка «Автор» | java.swing .JMenuItem |
| Кнопка «Про програму» | java.swing .JMenuItem |
| Бічна панель кнопок | java.swing .JPanel |
| Кнопка «Новий файл» | java.swing .JButton |
| Кнопка «Зберегти файл» | java.swing .JButton |
| Кнопка «Зберегти як…» | java.swing .JButton |
| Кнопка «Відкрити файл» | java.swing .JButton |
| Кнопка «Додати сектор» | java.swing .JButton |
| Кнопка «Видалити сектор» | java.swing .JButton |
| Кнопка «Редагування секторів» | java.swing .JButton |
| Кнопка «Очистити діаграму» | java.swing .JButton |

Додаток В. Діаграма класів програмного додатку “Редактор кільцевих діаграм”

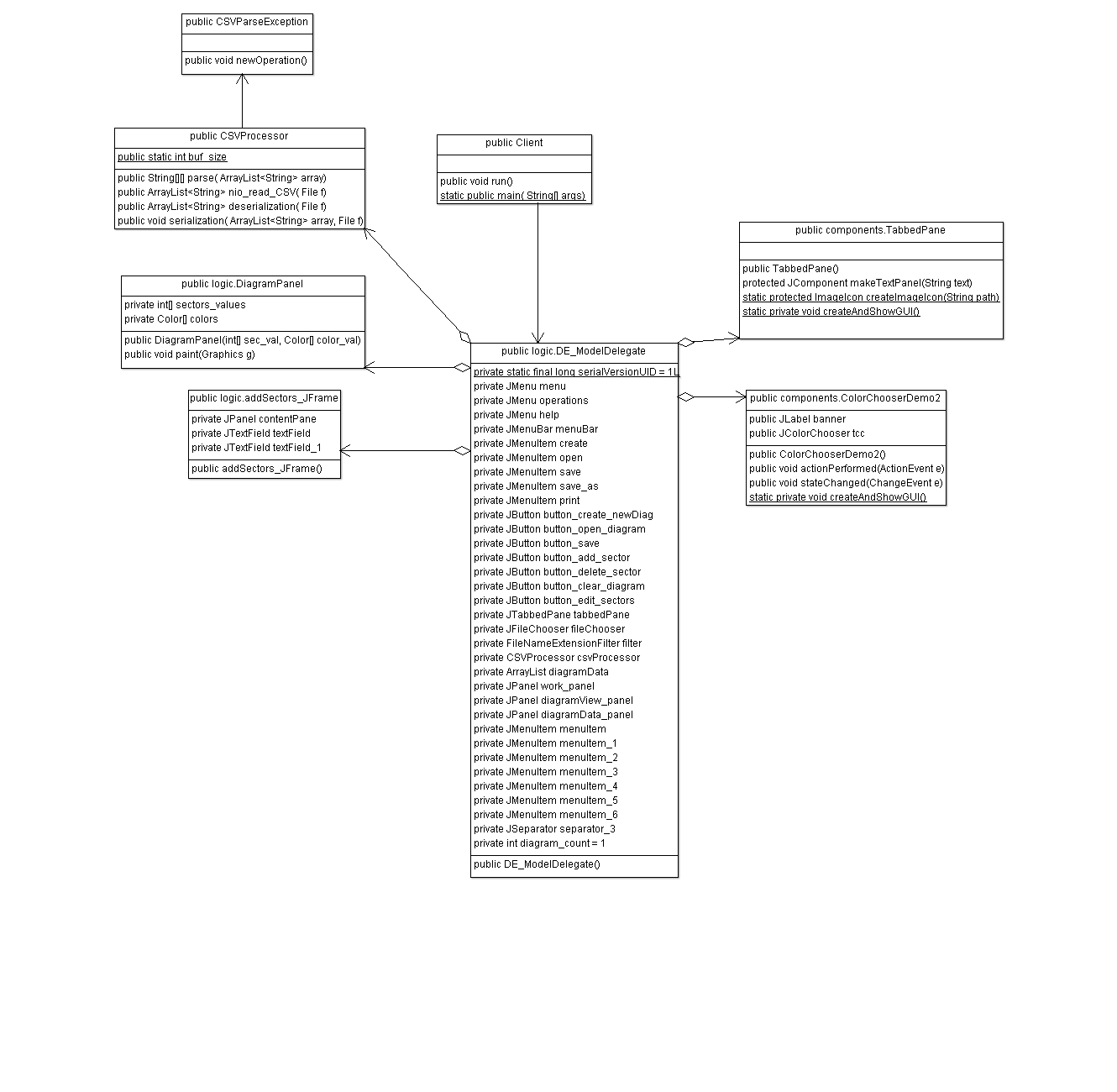


Рис. 3.1 Діаграма класів програмного додатку “Редактор кільцевих діаграм”