Міністерство освіти та науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

Курсова робота

З дисципліни

«Інженерія програмного забезпечення»

*Виконав:*

Студент ФІОТ

Групи ІО-12

Нестерук Ю.О.

м. Київ

2013 р.

ЗМІСТ

[ВСТУП 3](#_Toc319596709)

[1 Технічне завдання 4](#_Toc319596710)

[1.1 Загальне завдання 4](#_Toc319596711)

[1.2 Функціональність 4](#_Toc319596712)

[1.3 Вимоги до реалізації 4](#_Toc319596713)

[1.4 Варіант завдання 5](#_Toc319596714)

[2 Проектування інтерфейсу користувача 6](#_Toc319596715)

2.1 UML-діаграма класів……………………………………...……………...…………………..……..6

2.2 [Графічний інтерфейс користувача .](#_Toc319596716)7

2.3 [Розробка орієнтирів 8](#_Toc319596717)

2.4 [Діаграма граничних класів 9](#_Toc319596718)

[3 Розробка програмного забезпечення 10](#_Toc319596722)

3.1 [Скріншоти програми 11](#_Toc319596723)

3.2 [Специфікація класів 13](#_Toc319596724)

[ВИСНОВКИ](#_Toc319596725) 29

[СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 3](#_Toc319596726)0

[ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЕКТУ 3](#_Toc319596727)2

[ДОДАТОК Б. СТРУКТУРА ПРОЕКТУ 52](#_Toc319596728)

ВСТУП

У курсовій роботі розробляється пакет утилітних класів для роботи з растровими зображеннями. Також необхідно реалізувати процеси обробки зображень для їх подальшого використання інтерпритатором.

У роботі розглядаються основні принципи роботи з зображеннями як з масивами чисел, та їх растеризація.

Розробка програмного забезпечення виконується на мові програмування Java. Робота містить повну документацію, а так же програмний код проекту та UML діаграму класів.

# Технічне завдання

## Загальне завдання

Необхідно побудувати пакет утилітних класів для роботи з растровими зображеннями кольорової моделі RGB. Повинні бути реалізовані методи для завантаження і збереження зображень, представлення їх у матричній формі, знаходження яскравостей їх пікселів, растеризації їх сіткового представлення.

## Функціональність

Бібліотека класів повинна мати наступну функціональність:

* можливість завантаження/збереження зображень за вказаним шляхом;
* представлення зображень у матричному вигляді (двовимірним масивом чисел);
* знаходження матриці яскравостей пікселів зображення, можливість вказувати власні коефіцієнти для RGB каналів;
* растеризація сіткового представлення зображень у різних режимах відображення;
* реалізація процесів, які може виконувати інтерпритатор.

## Вимоги до реалізації

* мова програмування Java з використанням бібліотеки AWT;
* проект має бути повністю задокументований за допомогою JavaDoc;
* проект має повністю відповідати правилам CheckStyle;
* ?

## Варіант завдання

?

# Проектування Бібліотеки класів

## 2.1 Пакет ImageUtil

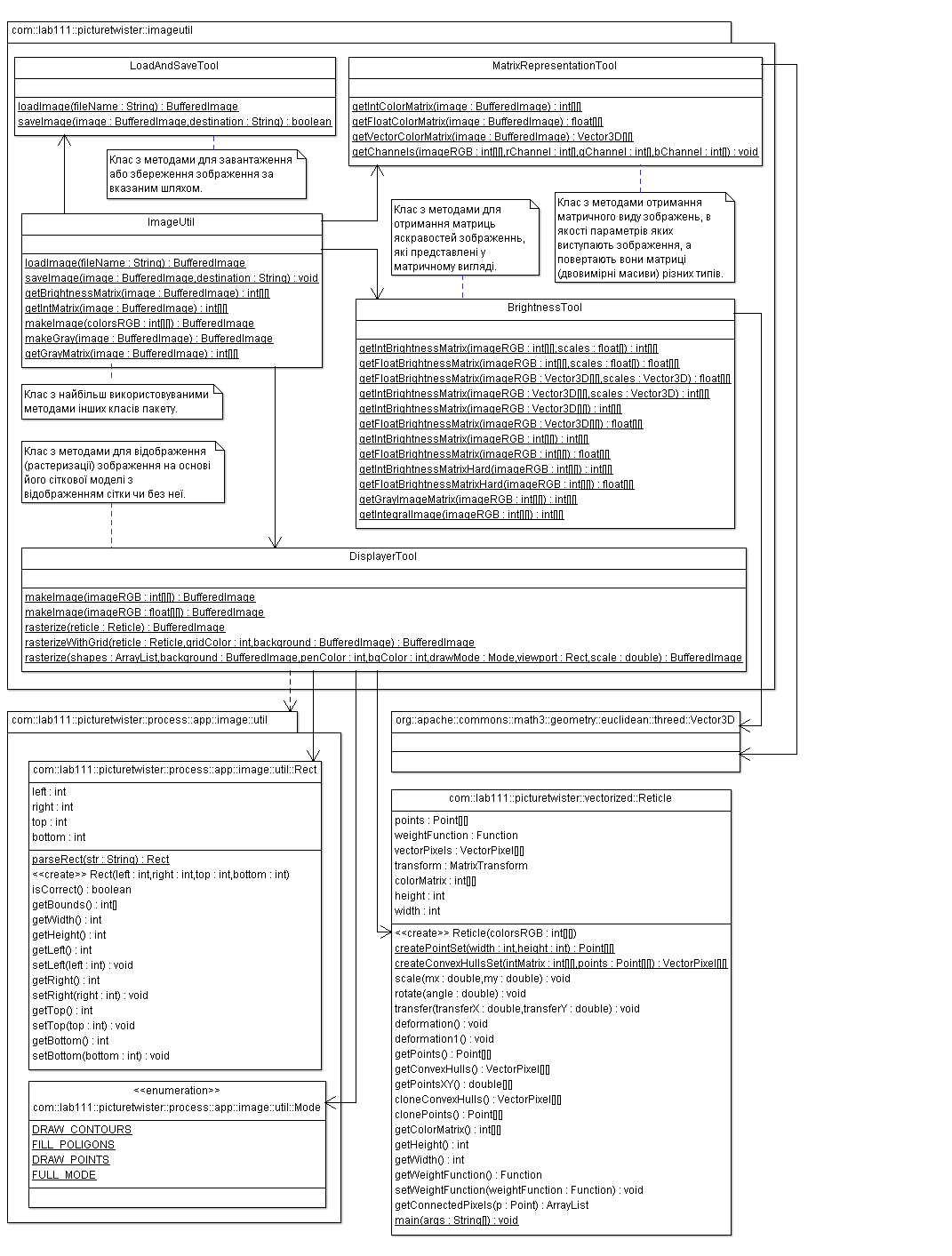


Рисунок 2.1.Діаграма класів пакету ImageUtil

Пакет ImageUtil містить наступні класи для роботи з растровими зображеннями:

* **LoadAndSaveTool** – містить методи для завантаження\збереження зображень.
* **MatrixRepresentationTool** – містить методи для отримання матричного представлення зображень.
* **BrightnessTool** – містить методи для отримання матриці яскравостей пікселів зображення на основі його матричного представлення**.**
* **DisplayerTool** – містить методи для растеризації зображення на основі його матричного представлення або сіткової моделі (Reticle).
* **ImageUtil** – містить ті методи вищевказаних класів, які використовуються найчастіше.

## Опис класів пакету ImageUtil

### Клас

**2.2 Графічний інтерфейс користувача**

**Призначення:**

Основне вікно програми використовується для створення і редагування кругових діаграм.

1. Прецедент починається, коли користувач запускає програму

2. Користувач може виконати одну з дій:

1. Відкрити CSV файл

2. Вийти з програми.

3. Побудувати діаграму.

4. Додати рядки або стовпчики

5. Зберегти дані як CSV файл

6. Зберегти діграму у форматі PNG

Прецедент закінчується, при закрітті Програми.

**2.3 Розробка орієнтирів**

1. Прецедент починається коли користувач запускає програму

2. Користувач може виконати одну з дій:

1. Відкрити CSV файл [Користувач повинен мати можливість відображення тільки файлів CSV. При натисканні викликається файловий діалог, в якому користувач вказує файл для завантаження] (Виконується в 60% випадків);

2. Зберегти дані як CSV файл [Користувач повинен мати можливість зберегти файл після його редагування] (Виконується в 50% випадків);

3. Зберегти діаграму в форматі PNG [Користувач повинен мати можливість зберегти діаграми в PNG форматі для можливості подальшого використання] (Виконується в 20% випадків);

4. Побудувати діаграму [Користувач повинен мати можливість будувати діаграми за данням таблиці]. (Виконується в 99% випадків);

5. Змінити дані в таблиці[Автоматично змінюється графічна область] (Виконується в 75% випадків)

3. Прецедент закінчується, коли користувач завершує роботу з додатком.

**2.4 Діаграма граничних класів:**

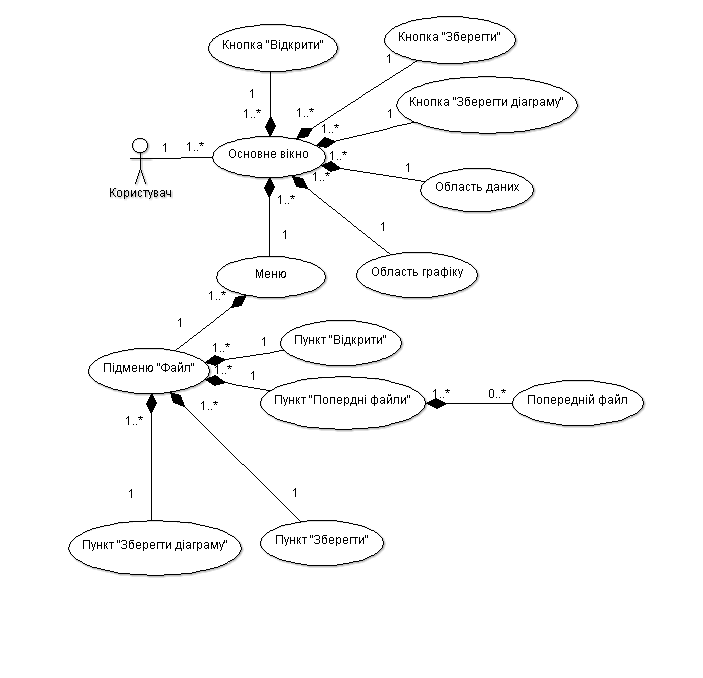
****

Рисунок 2.2. Діаграма граничних класів

# Розробка програмного забезпечення

**3.1Таблиця відповідності елементів дизайну і класів з бібліотеки Swing**

Таблиця 3.1. Відповідності елементів

|  |  |
| --- | --- |
| Програма | Swing |
| БарМеню | javax.swing.JMenuBar |
| Меню «File» | javax.swing.JMenu |
| Компонент меню «Open» | Javax.swing.JMenuItem |
| Компонент меню «Save csv» | Javax.swing.JMenuItem |
| Компонент меню «Save png » | Javax.swing.JMenuItem |
| Файловый діалог | javax.swing.JFileChooser |
| Панель таблиці | Javax.swing.JPanel |
| Таблиця | Javax.swing.JTable |
| Панель малювання | Javax.swing.JPanel |

**3.2 Скріншоти програми**

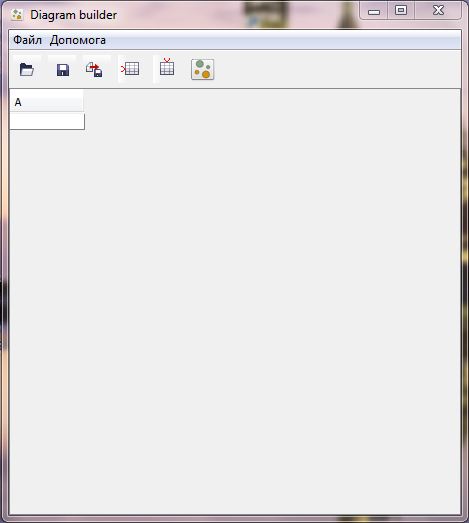
****

Рисунок 3.1. Створення нової діаграми

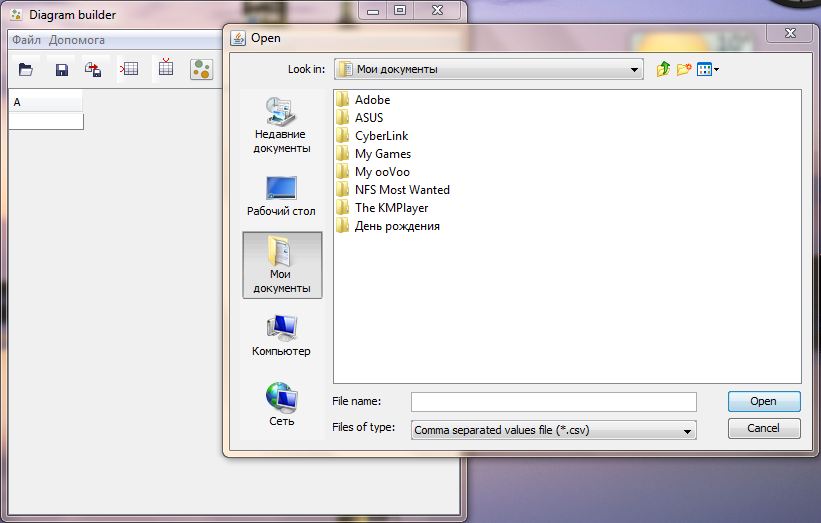
****

Рисунок 3.2. Відкриття файлу

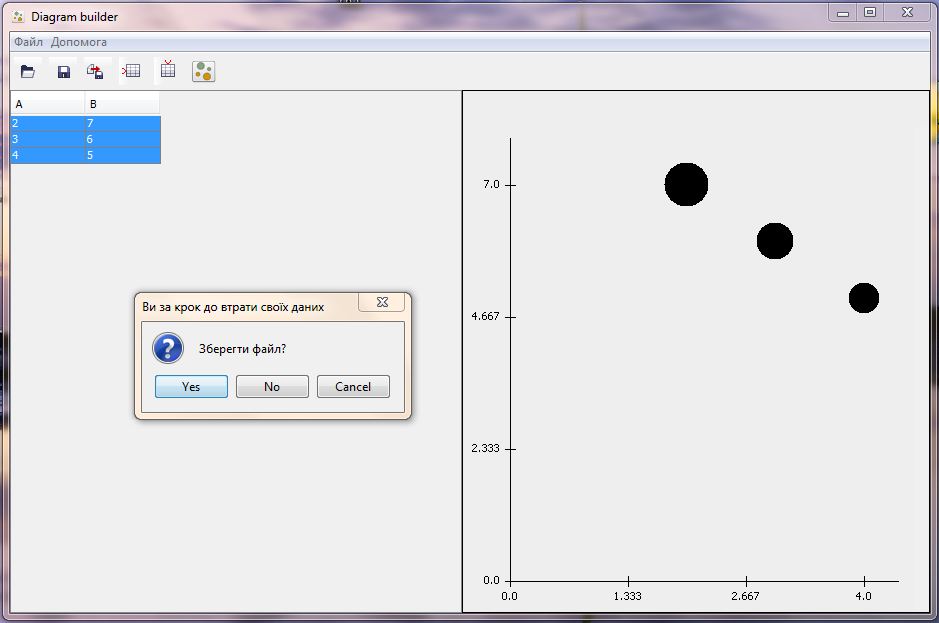


Рисунок 3.3. Побудована діаграма

Специфікація класів

## diagram\_builder Class CSVProcessor

java.lang.Object

extended by **diagram\_builder.CSVProcessor**

public class **CSVProcessor**

extends java.lang.Object

Читає та записує файли формату \*.csv

**Author:**

Стремецький Олексій

|  |  |
| --- | --- |
| **Field Summary** | |
| private  double[][] | [**parsedTable**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#parsedTable)            Розібрана таблиця |
| java.lang.String | [**serializedFileName**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#serializedFileName)            Ім'я сереалізованого файлу |
| private  java.util.ArrayList<java.lang.String> | [**table**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#table)            Завантажена таблиця |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**CSVProcessor**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#CSVProcessor())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| java.util.ArrayList<java.lang.String> | [**deserialize**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#deserialize())()            Десереалізувати ArrayList |
| private  int | [**findPos**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#findPos(java.lang.String, int))(java.lang.String temp, int tokens)            Знаходження зміщення помилки |
| double[][] | [**getParsedTable**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#getParsedTable())()            Повертає розібраний масив |
| java.util.ArrayList<java.lang.String> | [**loadFile**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#loadFile(java.io.File))(java.io.File file)            Завантажити CSV файл |
| java.util.ArrayList<java.lang.String> | [**loadFile**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#loadFile(java.lang.String))(java.lang.String fileName)            Завантажити CSV файл |
| double[][] | [**parse**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#parse())()            Розбирає завантажений ArrayList |
| void | [**saveFile**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#saveFile(java.util.ArrayList, java.io.File))(java.util.ArrayList<java.lang.String> table, java.io.File file)            Збреження CSV файлу |
| void | [**saveFile**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#saveFile(java.util.ArrayList, java.lang.String))(java.util.ArrayList<java.lang.String> table, java.lang.String fileName)            Збреження CSV файлу |
| void | [**serialize**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html#serialize(java.util.ArrayList))(java.util.ArrayList<java.lang.String> table)            Сереалізування ArrayList |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Field Detail** |

### serializedFileName

public final java.lang.String **serializedFileName**

Ім'я сереалізованого файлу

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.CSVProcessor.serializedFileName)

### table

private java.util.ArrayList<java.lang.String> **table**

Завантажена таблиця

### parsedTable

private double[][] **parsedTable**

Розібрана таблиця

|  |
| --- |
| **Constructor Detail** |

### CSVProcessor

public **CSVProcessor**()

|  |
| --- |
| **Method Detail** |

### loadFile

public java.util.ArrayList<java.lang.String> **loadFile**(java.lang.String fileName)

throws java.io.IOException

Завантажити CSV файл

**Parameters:**

fileName - ім'я файлу

**Returns:**

ArrayList з рядків CSV файлу

**Throws:**

java.io.IOException - виключення вводу-виводу

### loadFile

public java.util.ArrayList<java.lang.String> **loadFile**(java.io.File file)

throws java.io.IOException

Завантажити CSV файл

**Parameters:**

file - дескриптор файлу

**Returns:**

ArrayList з рядків CSV файлу

**Throws:**

java.io.IOException - виключення вводу-виводу

### deserialize

public java.util.ArrayList<java.lang.String> **deserialize**()

throws java.io.FileNotFoundException,

java.io.IOException,

java.lang.ClassNotFoundException

Десереалізувати ArrayList

**Returns:**

ArrayList з рядків CSV файлу

**Throws:**

java.io.FileNotFoundException - виключення незнайденого файлу

java.io.IOException - виключення вводу-виводу

java.lang.ClassNotFoundException - виключення незнайденого класу

### saveFile

public void **saveFile**(java.util.ArrayList<java.lang.String> table,

java.lang.String fileName)

throws java.io.IOException

Збреження CSV файлу

**Parameters:**

table - ArrayList з рядків CSV файлу

fileName - ім'я файлу

**Throws:**

java.io.IOException - виключення вводу-виводу

### saveFile

public void **saveFile**(java.util.ArrayList<java.lang.String> table,

java.io.File file)

throws java.io.IOException

Збреження CSV файлу

**Parameters:**

table - ArrayList з рядків CSV файлу

file - дескриптор файлу

**Throws:**

java.io.IOException - виключення вводу-виводу

### serialize

public void **serialize**(java.util.ArrayList<java.lang.String> table)

throws java.io.FileNotFoundException,

java.io.IOException

Сереалізування ArrayList

**Parameters:**

table - ArrayList з рядків CSV файлу

**Throws:**

java.io.FileNotFoundException - виключення незнайденого файлу

java.io.IOException - виключення вводу-виводу

### parse

public double[][] **parse**()

throws [CSVParseException](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVParseException.html)

Розбирає завантажений ArrayList

**Returns:**

масив з масивів стрічок

**Throws:**

[CSVParseException](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVParseException.html) - виключення розбору CSV

### findPos

private int **findPos**(java.lang.String temp,

int tokens)

Знаходження зміщення помилки

**Parameters:**

temp - стрічка з помилкою

tokens - очікувана кількість знаків

**Returns:**

error зміщення

### getParsedTable

public double[][] **getParsedTable**()

Повертає розібраний масив

**Returns:**

розібраний масив

## diagram\_builder Class CSVParseException

java.lang.Object

extended by java.lang.Throwable

extended by java.lang.Exception

extended by java.text.ParseException

extended by **diagram\_builder.CSVParseException**

**All Implemented Interfaces:**

java.io.Serializable

public class **CSVParseException**

extends java.text.ParseException

Виключення розбору CSV файлу

**Author:**

Стремецький Олексій

**See Also:**

[Serialized Form](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\serialized-form.html#diagram_builder.CSVParseException)

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**CSVParseException**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVParseException.html#CSVParseException(java.lang.String, int))(java.lang.String s, int errorOffset)            Конструктор за замовчуванням |  |

|  |
| --- |
| **Method Summary** |

|  |
| --- |
| **Methods inherited from class java.text.ParseException** |
| getErrorOffset |

|  |
| --- |
| **Methods inherited from class java.lang.Throwable** |
| fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### CSVParseException

public **CSVParseException**(java.lang.String s,

int errorOffset)

Конструктор за замовчуванням

**Parameters:**

s - стрічка, що викликала виключення

errorOffset - номер помилкового символу

## diagram\_builder Class DiagramBuilderFrame

java.lang.Object

extended by java.awt.Component

extended by java.awt.Container

extended by java.awt.Window

extended by java.awt.Frame

extended by javax.swing.JFrame

extended by **diagram\_builder.DiagramBuilderFrame**

**All Implemented Interfaces:**

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

public class **DiagramBuilderFrame**

extends javax.swing.JFrame

Головний фрейм, містить компоненти для завантаження, редактування, збереження \*.csv файлів та побудови і збереження графіків

**Author:**

Стремецький Олексій

**See Also:**

[Serialized Form](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\serialized-form.html#diagram_builder.DiagramBuilderFrame)

|  |
| --- |
| **Nested Class Summary** |

|  |
| --- |
| **Nested classes/interfaces inherited from class javax.swing.JFrame** |
| javax.swing.JFrame.AccessibleJFrame |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Frame** |
| java.awt.Frame.AccessibleAWTFrame |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Window** |
| java.awt.Window.AccessibleAWTWindow |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Container** |
| java.awt.Container.AccessibleAWTContainer |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy |

|  |  |
| --- | --- |
| **Field Summary** | |
| private  java.awt.event.ActionListener | [**addColListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#addColListener)            Слухач подій кнопки панелі "Додати стовпчик" |
| private  java.awt.event.ActionListener | [**addRowListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#addRowListener)            Слухач подій кнопки панелі "Додати рядок" |
| private  javax.swing.filechooser.FileFilter | [**csvFileFilter**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#csvFileFilter)            Фільтр файлів \*.csv |
| private  [DiagramPanel](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html) | [**drawer**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#drawer)            Панель для малювання бульбашкової діаграми |
| private  java.awt.event.ActionListener | [**drawListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#drawListener)            Слухач подій кнопки панелі "Намалювати діаграму" |
| private  javax.swing.JFileChooser | [**fileChooser**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#fileChooser)            Діалог завантаження/збереження |
| private  java.awt.event.ActionListener | [**fileExitListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#fileExitListener)            Слухач подій кнопки меню "Вийти" |
| private  java.awt.event.ActionListener | [**fileOpenListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#fileOpenListener)            Слухач подій кнопки меню "Відкрити" |
| private  java.awt.event.ActionListener | [**fileSaveDiagramListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#fileSaveDiagramListener)            Слухач подій кнопки меню "Зберегти діаграму" |
| private  java.awt.event.ActionListener | [**fileSaveListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#fileSaveListener)            Слухач подій кнопки меню "Зберегти" |
| private  java.awt.event.ActionListener | [**helpAboutListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#helpAboutListener)            Слухач подій кнопки меню "Про програму" |
| private  javax.swing.filechooser.FileFilter | [**pngFileFilter**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#pngFileFilter)            Фільтр файлів \*.png |
| private  [CSVProcessor](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html) | [**processor**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#processor)            Оброблювач \*.csv файлів |
| private  int[][] | [**selected**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#selected)            Виділені стовпці та рядки |
| private static long | [**serialVersionUID**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#serialVersionUID)            Ідентифікатор класу |
| private  javax.swing.JTable | [**table**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#table)            Таблиця значень |
| private  javax.swing.event.TableModelListener | [**tableListener**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#tableListener)            Слухач зміни данних таблиці |
| private  java.awt.event.WindowAdapter | [**windowAdapter**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#windowAdapter)            Адаптер вікна |

|  |
| --- |
| **Fields inherited from class javax.swing.JFrame** |
| accessibleContext, EXIT\_ON\_CLOSE, rootPane, rootPaneCheckingEnabled |

|  |
| --- |
| **Fields inherited from class java.awt.Frame** |
| CROSSHAIR\_CURSOR, DEFAULT\_CURSOR, E\_RESIZE\_CURSOR, HAND\_CURSOR, ICONIFIED, MAXIMIZED\_BOTH, MAXIMIZED\_HORIZ, MAXIMIZED\_VERT, MOVE\_CURSOR, N\_RESIZE\_CURSOR, NE\_RESIZE\_CURSOR, NORMAL, NW\_RESIZE\_CURSOR, S\_RESIZE\_CURSOR, SE\_RESIZE\_CURSOR, SW\_RESIZE\_CURSOR, TEXT\_CURSOR, W\_RESIZE\_CURSOR, WAIT\_CURSOR |

|  |
| --- |
| **Fields inherited from class java.awt.Component** |
| BOTTOM\_ALIGNMENT, CENTER\_ALIGNMENT, LEFT\_ALIGNMENT, RIGHT\_ALIGNMENT, TOP\_ALIGNMENT |

|  |
| --- |
| **Fields inherited from interface javax.swing.WindowConstants** |
| DISPOSE\_ON\_CLOSE, DO\_NOTHING\_ON\_CLOSE, HIDE\_ON\_CLOSE |

|  |
| --- |
| **Fields inherited from interface java.awt.image.ImageObserver** |
| ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**DiagramBuilderFrame**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#DiagramBuilderFrame())()            Конструктор за замовчуванням |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| private  void | [**centerWindow**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#centerWindow())()            Центрування вікна |
| private  void | [**draw**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#draw())()            Підготовка до малювання діаграми |
| private  void | [**fileSave**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#fileSave())()            Збереження файлу |
| private  void | [**showMessage**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramBuilderFrame.html#showMessage(java.lang.String, int))(java.lang.String message, int type)            Відображення повідомлення |

|  |
| --- |
| **Field Detail** |

### serialVersionUID

private static final long **serialVersionUID**

Ідентифікатор класу

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramBuilderFrame.serialVersionUID)

### table

private javax.swing.JTable **table**

Таблиця значень

### fileChooser

private javax.swing.JFileChooser **fileChooser**

Діалог завантаження/збереження

### processor

private [CSVProcessor](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\CSVProcessor.html) **processor**

Оброблювач \*.csv файлів

### drawer

private [DiagramPanel](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html) **drawer**

Панель для малювання бульбашкової діаграми

### selected

private int[][] **selected**

Виділені стовпці та рядки

### windowAdapter

private java.awt.event.WindowAdapter **windowAdapter**

Адаптер вікна

### fileExitListener

private java.awt.event.ActionListener **fileExitListener**

Слухач подій кнопки меню "Вийти"

### fileOpenListener

private java.awt.event.ActionListener **fileOpenListener**

Слухач подій кнопки меню "Відкрити"

### fileSaveListener

private java.awt.event.ActionListener **fileSaveListener**

Слухач подій кнопки меню "Зберегти"

### fileSaveDiagramListener

private java.awt.event.ActionListener **fileSaveDiagramListener**

Слухач подій кнопки меню "Зберегти діаграму"

### addRowListener

private java.awt.event.ActionListener **addRowListener**

Слухач подій кнопки панелі "Додати рядок"

### addColListener

private java.awt.event.ActionListener **addColListener**

Слухач подій кнопки панелі "Додати стовпчик"

### csvFileFilter

private javax.swing.filechooser.FileFilter **csvFileFilter**

Фільтр файлів \*.csv

### pngFileFilter

private javax.swing.filechooser.FileFilter **pngFileFilter**

Фільтр файлів \*.png

### drawListener

private java.awt.event.ActionListener **drawListener**

Слухач подій кнопки панелі "Намалювати діаграму"

### tableListener

private javax.swing.event.TableModelListener **tableListener**

Слухач зміни данних таблиці

### helpAboutListener

private java.awt.event.ActionListener **helpAboutListener**

Слухач подій кнопки меню "Про програму"

|  |
| --- |
| **Constructor Detail** |

### DiagramBuilderFrame

**DiagramBuilderFrame**()

Конструктор за замовчуванням

|  |
| --- |
| **Method Detail** |

### centerWindow

private void **centerWindow**()

Центрування вікна

### draw

private void **draw**()

Підготовка до малювання діаграми

### fileSave

private void **fileSave**()

Збереження файлу

### showMessage

private void **showMessage**(java.lang.String message,

int type)

Відображення повідомлення

**Parameters:**

message - повідомлення

type - тип повідомлення

## diagram\_builder Class DiagramPanel

java.lang.Object

extended by java.awt.Component

extended by java.awt.Container

extended by javax.swing.JComponent

extended by javax.swing.JPanel

extended by **diagram\_builder.DiagramPanel**

**All Implemented Interfaces:**

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

public class **DiagramPanel**

extends javax.swing.JPanel

Панель для малювання бульбашкової діаграми

**Author:**

Стремецький Олексій

**See Also:**

[Serialized Form](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\serialized-form.html#diagram_builder.DiagramPanel)

|  |
| --- |
| **Nested Class Summary** |

|  |
| --- |
| **Nested classes/interfaces inherited from class javax.swing.JPanel** |
| javax.swing.JPanel.AccessibleJPanel |

|  |
| --- |
| **Nested classes/interfaces inherited from class javax.swing.JComponent** |
| javax.swing.JComponent.AccessibleJComponent |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Container** |
| java.awt.Container.AccessibleAWTContainer |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy |

|  |  |
| --- | --- |
| **Field Summary** | |
| private  int | [**BORDER**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#BORDER)            Розмір відступу |
| private  double | [**bubbleR**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#bubbleR)            Максимальний радіус бульбашки |
| private  java.awt.Rectangle | [**plotBound**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#plotBound)            Межі малювання графіка |
| private  double | [**scaleRatio**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#scaleRatio)            Пропорція масштабування |
| private static long | [**serialVersionUID**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#serialVersionUID)            Ідентифікатор класу |
| private  int | [**TEXT\_BORDER**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#TEXT_BORDER)            Розмір відступу для тексту |
| private  int | [**TOPLEFT\_BORDER**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#TOPLEFT_BORDER)            Повний розмір відступу |
| private  double[] | [**x**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#x)            Масив координат X |
| private  int | [**X\_MARK**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#X_MARK)            Розмір помітки X |
| private  double | [**xStep**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#xStep)            Крок сітки X |
| private  int | [**xStepPixel**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#xStepPixel)            Крок сітки X в пікселях |
| private  double[] | [**y**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#y)            Масив координат Y |
| private  int | [**Y\_MARK**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#Y_MARK)            Розмір помітки Y |
| private  double | [**yStep**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#yStep)            Крок сітки Y |
| private  int | [**yStepPixel**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#yStepPixel)            Крок сітки Y в пікселях |

|  |
| --- |
| **Fields inherited from class javax.swing.JComponent** |
| accessibleContext, listenerList, TOOL\_TIP\_TEXT\_KEY, ui, UNDEFINED\_CONDITION, WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT, WHEN\_FOCUSED, WHEN\_IN\_FOCUSED\_WINDOW |

|  |
| --- |
| **Fields inherited from class java.awt.Component** |
| BOTTOM\_ALIGNMENT, CENTER\_ALIGNMENT, LEFT\_ALIGNMENT, RIGHT\_ALIGNMENT, TOP\_ALIGNMENT |

|  |
| --- |
| **Fields inherited from interface java.awt.image.ImageObserver** |
| ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**DiagramPanel**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#DiagramPanel(double[], double[]))(double[] x, double[] y)            Конструктор за замовчуванням |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| private  void | [**drawGrid**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#drawGrid(java.awt.Graphics))(java.awt.Graphics g)            Малює сітку у контексті |
| private  int | [**getMaxValuePos**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#getMaxValuePos(double[]))(double[] array)            Знаходить позицію найбільшого елемента |
| private  int | [**getPlotBound**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#getPlotBound(int))(int res)            Зменшує область малювання на розмір відступу |
| private  void | [**getRes**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#getRes(double[], double[], java.awt.Rectangle))(double[] coord1, double[] coord2, java.awt.Rectangle bounds)            Обчислює розміри для графіка |
| void | [**paintComponent**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#paintComponent(java.awt.Graphics))(java.awt.Graphics g)            Перемальовує діаграму у контексті |
| void | [**setX**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#setX(double[]))(double[] x)            Встановлює координати X |
| void | [**setY**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\DiagramPanel.html#setY(double[]))(double[] y)            Встановл.є координати Y |

|  |
| --- |
| **Field Detail** |

### serialVersionUID

private static final long **serialVersionUID**

Ідентифікатор класу

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramPanel.serialVersionUID)

### x

private double[] **x**

Масив координат X

### y

private double[] **y**

Масив координат Y

### xStep

private double **xStep**

Крок сітки X

### yStep

private double **yStep**

Крок сітки Y

### xStepPixel

private int **xStepPixel**

Крок сітки X в пікселях

### yStepPixel

private int **yStepPixel**

Крок сітки Y в пікселях

### plotBound

private java.awt.Rectangle **plotBound**

Межі малювання графіка

### BORDER

private final int **BORDER**

Розмір відступу

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramPanel.BORDER)

### TEXT\_BORDER

private final int **TEXT\_BORDER**

Розмір відступу для тексту

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramPanel.TEXT_BORDER)

### X\_MARK

private final int **X\_MARK**

Розмір помітки X

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramPanel.X_MARK)

### Y\_MARK

private final int **Y\_MARK**

Розмір помітки Y

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramPanel.Y_MARK)

### TOPLEFT\_BORDER

private final int **TOPLEFT\_BORDER**

Повний розмір відступу

**See Also:**

[Constant Field Values](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\constant-values.html#diagram_builder.DiagramPanel.TOPLEFT_BORDER)

### bubbleR

private double **bubbleR**

Максимальний радіус бульбашки

### scaleRatio

private double **scaleRatio**

Пропорція масштабування

|  |
| --- |
| **Constructor Detail** |

### DiagramPanel

**DiagramPanel**(double[] x,

double[] y)

Конструктор за замовчуванням

**Parameters:**

x - Координати x

y - Координати y

|  |
| --- |
| **Method Detail** |

### paintComponent

public void **paintComponent**(java.awt.Graphics g)

Перемальовує діаграму у контексті

**Overrides:**

paintComponent in class javax.swing.JComponent

**Parameters:**

g - контекст малювання

### drawGrid

private void **drawGrid**(java.awt.Graphics g)

Малює сітку у контексті

**Parameters:**

g - контекст малювання

### setX

public void **setX**(double[] x)

Встановлює координати X

**Parameters:**

x - координати X

### setY

public void **setY**(double[] y)

Встановл.є координати Y

**Parameters:**

y - координати Y

### getRes

private void **getRes**(double[] coord1,

double[] coord2,

java.awt.Rectangle bounds)

Обчислює розміри для графіка

**Parameters:**

coord1 - координати x

coord2 - координати y

bounds - межі малювання

### getPlotBound

private int **getPlotBound**(int res)

Зменшує область малювання на розмір відступу

**Parameters:**

res - розмір

**Returns:**

новий розмір

### getMaxValuePos

private int **getMaxValuePos**(double[] array)

Знаходить позицію найбільшого елемента

**Parameters:**

array - масив для пошуку

**Returns:**

позиція найбільшого елемента

## diagram\_builder Class Main

java.lang.Object

extended by **diagram\_builder.Main**

public class **Main**

extends java.lang.Object

Виконавчий клас

**Author:**

Стремецький Олексій

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Main**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\Main.html#Main())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| static void | [**main**](file:///C:\Users\Кристина\Desktop\diagram_builder\doc\diagram_builder\Main.html#main(java.lang.String[]))(java.lang.String[] args)            Виконавчий метод, з нього почианється виконання програми |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Main

public **Main**()

|  |
| --- |
| **Method Detail** |

### main

public static void **main**(java.lang.String[] args)

Виконавчий метод, з нього почианється виконання програми

**Parameters:**

args - параметри передені через командну стрічку

ВИСНОВКИ

В ході виконання курсової роботи придбано навички роботи з графічними пакетами Swing та AWT. Виконано взаємодію між елементами інтерфейсу, здобуто навички встановлення взаємодії між програмою і користувачем за допомогою бібліотеки java.awt.event.

Працюючи над додатками, вивчено не тільки графічні пакети, а й реалізацію таких засобів як обробка текстових даних, а саме розбір CSV файлу на окремі текстові частини та запис їх в матрицю.

Так само були використані засоби введення / виведення, за допомогою яких реалізовано считування даних з файлу і запис їх у файл, збереження діаграми як рисунка.

Важливим є засоби оголошення та виявлення виключних ситуацій, завдяки ним програма не припиняє свою роботу при невірному форматі файлу і його структури.

Вивчила організацію потоків, серіалізаціі і десеріалізаціі даних.

В результаті розробки всі пункти технічного завдання було виконано.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Герберт Шилдт SWING: руководство для начинающих – М.: “Вильямс”, 2007. – С. 704. – ISBN 0-07-226314-8.
2. 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.
3. Приемы объектно-ориентированого проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. – СПб.: Питер, 2010 – 368 с.: ил. –ISBN 978-5-469-01136-1.
4. 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.).
5. <http://www.uic.rsu.ru/doc/programming/java/TIJ2e.ru/Chapter11.html>

**ДОДАТКИ**

ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЕКТУ

package diagram\_builder;

import java.text.ParseException;

/\*\*

\* Виключення розбору CSV файлу

\*

\* @author Стремецький Олексій

\*

\*/

public class CSVParseException extends ParseException {

/\*\*

\* Конструктор за замовчуванням

\*

\* @param s

\* стрічка, що викликала виключення

\* @param errorOffset

\* номер помилкового символу

\*/

public CSVParseException(String s, int errorOffset) {

super(s, errorOffset);

}

}

package diagram\_builder;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.PrintWriter;

import java.util.ArrayList;

/\*\*

\* Читає та записує файли формату \*.csv

\*

\* @author Стремецький Олексій

\*

\*/

public class CSVProcessor {

/\*\*

\* Ім'я сереалізованого файлу

\*/

public final String serializedFileName = "serialized.dat";

/\*\*

\* Завантажена таблиця

\*/

private ArrayList<String> table = null;

/\*\*

\* Розібрана таблиця

\*/

private double[][] parsedTable;

/\*\*

\* Завантажити CSV файл

\*

\* @param fileName

\* ім'я файлу

\* @return ArrayList з рядків CSV файлу

\* @throws IOException

\* виключення вводу-виводу

\*/

public ArrayList<String> loadFile(String fileName) throws IOException {

return loadFile(new File(fileName));

}

/\*\*

\* Завантажити CSV файл

\*

\* @param file

\* дескриптор файлу

\* @return ArrayList з рядків CSV файлу

\* @throws IOException

\* виключення вводу-виводу

\*/

public ArrayList<String> loadFile(File file) throws IOException {

table = new ArrayList<String>();

BufferedReader stream = new BufferedReader(new FileReader(file));

while (stream.ready()) {

table.add(stream.readLine());

}

stream.close();

return table;

}

/\*\*

\* Десереалізувати ArrayList

\*

\* @return ArrayList з рядків CSV файлу

\* @throws FileNotFoundException

\* виключення незнайденого файлу

\* @throws IOException

\* виключення вводу-виводу

\* @throws ClassNotFoundException

\* виключення незнайденого класу

\*/

public ArrayList<String> deserialize() throws FileNotFoundException,

IOException, ClassNotFoundException {

ObjectInputStream stream = new ObjectInputStream(new FileInputStream(

serializedFileName));

table = (ArrayList<String>) stream.readObject();

stream.close();

return table;

}

/\*\*

\* Збреження CSV файлу

\*

\* @param table

\* ArrayList з рядків CSV файлу

\* @param fileName

\* ім'я файлу

\* @throws IOException

\* виключення вводу-виводу

\*/

public void saveFile(ArrayList<String> table, String fileName)

throws IOException {

saveFile(table, new File(fileName));

}

/\*\*

\* Збреження CSV файлу

\*

\* @param table

\* ArrayList з рядків CSV файлу

\* @param file

\* дескриптор файлу

\* @throws IOException

\* виключення вводу-виводу

\*/

public void saveFile(ArrayList<String> table, File file) throws IOException {

PrintWriter stream = new PrintWriter(new BufferedWriter(new FileWriter(

file)));

for (String t : table) {

stream.println(t);

}

stream.close();

}

/\*\*

\* Сереалізування ArrayList

\*

\* @param table

\* ArrayList з рядків CSV файлу

\* @throws FileNotFoundException

\* виключення незнайденого файлу

\* @throws IOException

\* виключення вводу-виводу

\*/

public void serialize(ArrayList<String> table)

throws FileNotFoundException, IOException {

ObjectOutputStream stream = new ObjectOutputStream(

new FileOutputStream(serializedFileName));

stream.writeObject(table);

stream.close();

}

/\*\*

\* Розбирає завантажений ArrayList

\*

\* @return масив з масивів стрічок

\* @throws CSVParseException

\* виключення розбору CSV

\*/

public double[][] parse() throws CSVParseException {

String[] tokens = table.get(0).split(",");

double[][] array = new double[table.size()][tokens.length];

String temp = null;

try {

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

temp = table.get(i);

tokens = temp.split(",");

if (tokens.length != array[i].length) {

throw new CSVParseException(temp, findPos(temp,

array[i].length));

} else {

for (int j = 0; j < array[i].length; j++) {

array[i][j] = Double.parseDouble(tokens[j]);

}

}

}

} catch (CSVParseException e) {

throw e;

} catch (Exception e) {

throw new CSVParseException(temp, 0);

}

parsedTable = array;

return array;

}

/\*\*

\* Знаходження зміщення помилки

\*

\* @param temp

\* стрічка з помилкою

\* @param tokens

\* очікувана кількість знаків

\* @return error зміщення

\*/

private int findPos(String temp, int tokens) {

int count = 0;

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

if (temp.charAt(i) == ',') {

count++;

}

if (count == tokens) {

return i;

}

}

return temp.length() - 1;

}

/\*\*

\* Повертає розібраний масив

\*

\* @return розібраний масив

\*/

public double[][] getParsedTable() {

return parsedTable;

}

}

package diagram\_builder;

import java.awt.BorderLayout;

import java.awt.Component;

import java.awt.Dimension;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.image.BufferedImage;

import java.awt.image.RenderedImage;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JToolBar;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import javax.swing.event.TableModelEvent;

import javax.swing.event.TableModelListener;

import javax.swing.filechooser.FileFilter;

import javax.swing.table.DefaultTableModel;

/\*\*

\* Головний фрейм, містить компоненти для завантаження, редактування, збереження

\* \*.csv файлів та побудови і збереження графіків

\*

\* @author Стремецький Олексій

\*

\*/

public class DiagramBuilderFrame extends JFrame {

/\*\*

\* Ідентифікатор класу

\*/

private static final long serialVersionUID = -5814076303962428492L;

/\*\*

\* Таблиця значень

\*/

private JTable table;

/\*\*

\* Діалог завантаження/збереження

\*/

private JFileChooser fileChooser;

/\*\*

\* Оброблювач \*.csv файлів

\*/

private CSVProcessor processor;

/\*\*

\* Панель для малювання бульбашкової діаграми

\*/

private DiagramPanel drawer;

/\*\*

\* Виділені стовпці та рядки

\*/

private int[][] selected;

/\*\*

\* Конструктор за замовчуванням

\*/

DiagramBuilderFrame() {

setLayout(null);

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

Toolkit toolkit = Toolkit.getDefaultToolkit();

setTitle("Diagram builder");

setIconImage(toolkit.getImage("icons/icon.png"));

setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

addWindowListener(windowAdapter);

setLayout(new BorderLayout());

JMenuBar menuBar = new JMenuBar();

JMenu fileMenu = new JMenu("Файл");

fileMenu.setMnemonic(KeyEvent.VK\_F);

JMenuItem fileOpen = new JMenuItem("Відкрити файл");

fileOpen.setMnemonic(KeyEvent.VK\_O);

fileOpen.addActionListener(fileOpenListener);

JMenuItem fileSave = new JMenuItem("Зберегти файл");

fileSave.setMnemonic(KeyEvent.VK\_S);

fileSave.addActionListener(fileSaveListener);

JMenuItem fileSaveDiagram = new JMenuItem("Зберегти діаграму");

fileSaveDiagram.setMnemonic(KeyEvent.VK\_D);

fileSaveDiagram.addActionListener(fileSaveDiagramListener);

JMenuItem fileExit = new JMenuItem("Вийти");

fileExit.setMnemonic(KeyEvent.VK\_X);

fileExit.addActionListener(fileExitListener);

fileMenu.add(fileOpen);

fileMenu.add(fileSave);

fileMenu.add(fileSaveDiagram);

fileMenu.addSeparator();

fileMenu.add(fileExit);

JMenu helpMenu = new JMenu("Допомога");

helpMenu.setMnemonic(KeyEvent.VK\_H);

JMenuItem helpAbout = new JMenuItem("Про програму");

helpAbout.setMnemonic(KeyEvent.VK\_A);

helpAbout.addActionListener(helpAboutListener);

helpMenu.add(helpAbout);

menuBar.add(fileMenu);

menuBar.add(helpMenu);

setJMenuBar(menuBar);

JToolBar toolBar = new JToolBar(JToolBar.CENTER);

toolBar.setFloatable(false);

JButton btnFileOpen = new JButton(

new ImageIcon("icons/btnFileOpen.png"));

btnFileOpen.setSize(32, 32);

btnFileOpen.setToolTipText("Відкрити файл");

btnFileOpen.addActionListener(fileOpenListener);

toolBar.add(btnFileOpen);

JButton btnFileSave = new JButton(

new ImageIcon("icons/btnFileSave.png"));

btnFileSave.setSize(32, 32);

btnFileSave.setToolTipText("Зберегти файл");

btnFileSave.addActionListener(fileSaveListener);

toolBar.add(btnFileSave);

JButton btnFileSaveDiagram = new JButton(new ImageIcon(

"icons/btnFileSaveDiagram.png"));

btnFileSaveDiagram.setSize(32, 32);

btnFileSaveDiagram.setToolTipText("Зберегти діаграму");

btnFileSaveDiagram.addActionListener(fileSaveDiagramListener);

toolBar.add(btnFileSaveDiagram);

JButton btnAddRow = new JButton(new ImageIcon("icons/btnAddRow.png"));

btnAddRow.setSize(32, 32);

btnAddRow.setToolTipText("Додати до таблиці рядок");

btnAddRow.addActionListener(addRowListener);

toolBar.add(btnAddRow);

JButton btnAddCol = new JButton(new ImageIcon("icons/btnAddCol.png"));

btnAddCol.setSize(32, 32);

btnAddCol.setToolTipText("Додати до таблиці стовпчик");

btnAddCol.addActionListener(addColListener);

toolBar.add(btnAddCol);

JButton btnDraw = new JButton(new ImageIcon("icons/btnDraw.png"));

btnDraw.setSize(32, 32);

btnDraw.setToolTipText("Намалювати графік");

btnDraw.addActionListener(drawListener);

toolBar.add(btnDraw);

add(toolBar, BorderLayout.NORTH);

table = new JTable(1, 1);

table.setCellSelectionEnabled(true);

table.getModel().addTableModelListener(tableListener);

JScrollPane scrollPane = new JScrollPane(table);

table.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);

add(scrollPane, BorderLayout.WEST);

pack();

centerWindow();

fileChooser = new JFileChooser();

fileChooser.setAcceptAllFileFilterUsed(false);

processor = new CSVProcessor();

setVisible(true);

}

/\*\*

\* Адаптер вікна

\*/

private WindowAdapter windowAdapter = new WindowAdapter() {

/\*\*

\* Обробка події закриття вікна

\*/

@Override

public void windowClosing(WindowEvent e) {

super.windowClosing(e);

int result = JOptionPane.showConfirmDialog(null, "Зберегти файл?",

"Ви за крок до втрати своїх даних",

JOptionPane.YES\_NO\_CANCEL\_OPTION);

if (result == JOptionPane.OK\_OPTION) {

fileSave();

} else if (result == JOptionPane.NO\_OPTION) {

dispose();

}

}

};

/\*\*

\* Слухач подій кнопки меню "Вийти"

\*/

private ActionListener fileExitListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

WindowEvent closingEvent = new WindowEvent(

DiagramBuilderFrame.this, WindowEvent.WINDOW\_CLOSING);

Toolkit.getDefaultToolkit().getSystemEventQueue()

.postEvent(closingEvent);

}

};

/\*\*

\* Слухач подій кнопки меню "Відкрити"

\*/

private ActionListener fileOpenListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

fileChooser.removeChoosableFileFilter(pngFileFilter);

fileChooser.addChoosableFileFilter(csvFileFilter);

if (fileChooser.showOpenDialog((Component) e.getSource()) == JFileChooser.APPROVE\_OPTION) {

setTitle(fileChooser.getSelectedFile() + " - Diagram Builder");

Thread t = new Thread(new Runnable() {

/\*\*

\* Запуск в окремому потоці

\*/

@Override

public void run() {

double[][] result;

Double[][] model;

int maxJ = 0;

try {

processor.loadFile(fileChooser.getSelectedFile());

result = processor.parse();

model = new Double[result.length][];

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

model[i] = new Double[result[i].length];

for (int j = 0; j < result[i].length; j++) {

model[i][j] = result[i][j];

if (j > maxJ) {

maxJ = j;

}

}

}

table.setModel(new DefaultTableModel(model,

new Object[maxJ + 1]));

} catch (IOException e) {

e.printStackTrace();

showMessage("Помилка завантаження файлу",

JOptionPane.ERROR\_MESSAGE);

} catch (CSVParseException e) {

e.printStackTrace();

showMessage("Схоже, що це не \*.csv файл",

JOptionPane.ERROR\_MESSAGE);

}

}

});

t.start();

}

}

};

/\*\*

\* Слухач подій кнопки меню "Зберегти"

\*/

private ActionListener fileSaveListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

fileSave();

}

};

/\*\*

\* Слухач подій кнопки меню "Зберегти діаграму"

\*/

private ActionListener fileSaveDiagramListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

fileChooser.removeChoosableFileFilter(csvFileFilter);

fileChooser.addChoosableFileFilter(pngFileFilter);

if (fileChooser.showSaveDialog((Component) e.getSource()) == JFileChooser.APPROVE\_OPTION) {

repaint();

BufferedImage img = (BufferedImage) drawer.createImage(

drawer.getWidth(), drawer.getHeight());

drawer.paint(img.getGraphics());

String fileName = fileChooser.getSelectedFile().getPath();

if (!fileName.toLowerCase().endsWith(".png")) {

fileName += ".png";

fileChooser.setSelectedFile(new File(fileName));

}

try {

ImageIO.write((RenderedImage) img, "png",

fileChooser.getSelectedFile());

} catch (IOException e1) {

showMessage("Зображення не збережено",

JOptionPane.ERROR\_MESSAGE);

e1.printStackTrace();

}

}

}

};

/\*\*

\* Слухач подій кнопки панелі "Додати рядок"

\*/

private ActionListener addRowListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

DefaultTableModel tm = (DefaultTableModel) table.getModel();

tm.addRow(new Object[1]);

}

};

/\*\*

\* Слухач подій кнопки панелі "Додати стовпчик"

\*/

private ActionListener addColListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

DefaultTableModel tm = (DefaultTableModel) table.getModel();

tm.addColumn((char) (table.getColumnName(tm.getColumnCount() - 1)

.charAt(0) + 1));

}

};

/\*\*

\* Фільтр файлів \*.csv

\*/

private FileFilter csvFileFilter = new FileFilter() {

/\*\*

\* Отримання опису

\*/

@Override

public String getDescription() {

return "Comma separated values file (\*.csv)";

}

/\*\*

\* Перевірка на відповідність

\*/

@Override

public boolean accept(File f) {

return f.getName().toLowerCase().endsWith(".csv")

|| f.isDirectory();

}

};

/\*\*

\* Фільтр файлів \*.png

\*/

private FileFilter pngFileFilter = new FileFilter() {

/\*\*

\* Отримання опису

\*/

@Override

public String getDescription() {

return "Portable network graphics file (\*.png)";

}

/\*\*

\* Перевірка на відповідність

\*/

@Override

public boolean accept(File f) {

return f.getName().toLowerCase().endsWith(".png")

|| f.isDirectory();

}

};

/\*\*

\* Слухач подій кнопки панелі "Намалювати діаграму"

\*/

private ActionListener drawListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

selected = new int[2][];

selected[0] = table.getSelectedColumns();

selected[1] = table.getSelectedRows();

if (selected[0].length == 0 || selected[1].length == 0) {

showMessage("Спочатку оберіть дані",

JOptionPane.WARNING\_MESSAGE);

return;

}

draw();

}

};

/\*\*

\* Слухач зміни данних таблиці

\*/

private TableModelListener tableListener = new TableModelListener() {

/\*\*

\* Обробка події

\*/

@Override

public void tableChanged(TableModelEvent e) {

if (selected != null) {

boolean selCol = false;

boolean selRow = false;

if (e.getType() == TableModelEvent.UPDATE) {

for (int i = 0; i < selected[0].length; i++) {

if (e.getColumn() == selected[0][i]) {

selCol = true;

break;

}

}

for (int i = 0; i < selected[1].length; i++) {

if (e.getFirstRow() == selected[1][i]) {

selRow = true;

break;

}

}

if (selCol && selRow) {

draw();

}

}

}

}

};

/\*\*

\* Слухач подій кнопки меню "Про програму"

\*/

private ActionListener helpAboutListener = new ActionListener() {

/\*\*

\* Обробка події

\*/

@Override

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null,

"Створено Стремецьким Олексієм aka Fors ©2012", getTitle(),

JOptionPane.INFORMATION\_MESSAGE);

}

};

/\*\*

\* Центрування вікна

\*/

private void centerWindow() {

Toolkit toolkit = getToolkit();

Dimension dim = toolkit.getScreenSize();

setLocation(dim.width / 2 - getWidth() / 2, dim.height / 2

- getHeight() / 2);

}

/\*\*

\* Підготовка до малювання діаграми

\*/

private void draw() {

if (selected != null) {

double[] x = null;

double[] y = null;

if (selected[0].length == 2) {

x = new double[selected[1].length];

y = new double[selected[1].length];

for (int i = 0; i < selected[1].length; i++) {

x[i] = Double.parseDouble(String.valueOf(table.getValueAt(

selected[1][i], selected[0][0])));

y[i] = Double.parseDouble(String.valueOf(table.getValueAt(

selected[1][i], selected[0][1])));

}

} else if (selected[1].length == 2) {

x = new double[selected[0].length];

y = new double[selected[0].length];

for (int i = 0; i < selected[0].length; i++) {

x[i] = Double.parseDouble(String.valueOf(table.getValueAt(

selected[1][0], selected[0][i])));

y[i] = Double.parseDouble(String.valueOf(table.getValueAt(

selected[1][1], selected[0][i])));

}

} else {

showMessage("Необідно обрати 2 стовпчики або 2 рядки",

JOptionPane.WARNING\_MESSAGE);

return;

}

if (drawer == null) {

int state = getExtendedState();

drawer = new DiagramPanel(x, y);

add(drawer, BorderLayout.CENTER);

drawer.setPreferredSize(new Dimension(getWidth(), getHeight()));

pack();

if (state == MAXIMIZED\_BOTH) {

setExtendedState(MAXIMIZED\_BOTH);

} else {

centerWindow();

}

} else {

drawer.setX(x);

drawer.setY(y);

repaint();

}

}

}

/\*\*

\* Збереження файлу

\*/

private void fileSave() {

fileChooser.removeChoosableFileFilter(pngFileFilter);

fileChooser.addChoosableFileFilter(csvFileFilter);

if (fileChooser.showSaveDialog(this) == JFileChooser.APPROVE\_OPTION) {

Thread t = new Thread(new Runnable() {

/\*\*

\* Запуск в окремому потоці

\*/

@Override

public void run() {

try {

ArrayList<String> output = new ArrayList<String>();

String element;

for (int i = 0; i < table.getRowCount(); i++) {

element = new String();

for (int j = 0; j < table.getColumnCount(); j++) {

element += table.getValueAt(i, j) + ",";

}

output.add(element.substring(0,

element.length() - 1));

}

String fileName = fileChooser.getSelectedFile()

.getPath();

if (!fileName.toLowerCase().endsWith(".csv")) {

fileName += ".csv";

fileChooser.setSelectedFile(new File(fileName));

}

processor.saveFile(output,

fileChooser.getSelectedFile());

} catch (IOException e) {

e.printStackTrace();

showMessage("Помилка збереження файлу",

JOptionPane.ERROR\_MESSAGE);

}

}

});

t.start();

}

}

/\*\*

\* Відображення повідомлення

\*

\* @param message

\* повідомлення

\* @param type

\* тип повідомлення

\*/

private void showMessage(String message, int type) {

JOptionPane.showMessageDialog(null, message,

"Нажаль щось пішло не так...", type);

}

}

package diagram\_builder;

import java.awt.Color;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Rectangle;

import javax.swing.JPanel;

import javax.swing.border.LineBorder;

/\*\*

\* Панель для малювання бульбашкової діаграми

\*

\* @author Стремецький Олексій

\*

\*/

public class DiagramPanel extends JPanel {

/\*\*

\* Ідентифікатор класу

\*/

private static final long serialVersionUID = -3580518691795972626L;

/\*\*

\* Масив координат X

\*/

private double[] x;

/\*\*

\* Масив координат Y

\*/

private double[] y;

/\*\*

\* Крок сітки X

\*/

private double xStep;

/\*\*

\* Крок сітки Y

\*/

private double yStep;

/\*\*

\* Крок сітки X в пікселях

\*/

private int xStepPixel;

/\*\*

\* Крок сітки Y в пікселях

\*/

private int yStepPixel;

/\*\*

\* Межі малювання графіка

\*/

private Rectangle plotBound;

/\*\*

\* Розмір відступу

\*/

private final int BORDER = 16;

/\*\*

\* Розмір відступу для тексту

\*/

private final int TEXT\_BORDER = 32;

/\*\*

\* Розмір помітки X

\*/

private final int X\_MARK = 5;

/\*\*

\* Розмір помітки Y

\*/

private final int Y\_MARK = 5;

/\*\*

\* Повний розмір відступу

\*/

private final int TOPLEFT\_BORDER = BORDER + TEXT\_BORDER;

/\*\*

\* Максимальний радіус бульбашки

\*/

private double bubbleR;

/\*\*

\* Пропорція масштабування

\*/

private double scaleRatio;

/\*\*

\* Конструктор за замовчуванням

\*

\* @param x

\* Координати x

\* @param y

\* Координати y

\*/

DiagramPanel(double[] x, double[] y) {

if (x.length == y.length) {

setX(x);

setY(y);

} else if (x.length > y.length) {

this.x = new double[y.length];

System.arraycopy(x, 0, this.x, 0, y.length);

this.setY(y);

} else if (y.length > x.length) {

this.y = new double[x.length];

System.arraycopy(y, 0, this.y, 0, x.length);

this.setX(x);

}

setBorder(new LineBorder(Color.black, 1, false));

}

/\*\*

\* Перемальовує діаграму у контексті

\*

\* @param g

\* контекст малювання

\*/

public void paintComponent(Graphics g) {

getRes(x, y, new Rectangle(getX(), getY(), getWidth(), getHeight()));

drawGrid(g);

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

g.fillOval(

(int) (TOPLEFT\_BORDER + x[i] / xStep \* xStepPixel - bubbleR

\* y[i] \* scaleRatio / 2),

(int) (TOPLEFT\_BORDER + plotBound.height - y[i] / yStep

\* yStepPixel - bubbleR \* y[i] \* scaleRatio / 2),

(int) (bubbleR \* y[i] \* scaleRatio),

(int) (bubbleR \* y[i] \* scaleRatio));

}

}

/\*\*

\* Малює сітку у контексті

\*

\* @param g

\* контекст малювання

\*/

private void drawGrid(Graphics g) {

g.setColor(getBackground());

g.fillRect(getX(), getY(), getWidth(), getHeight());

g.setColor(Color.BLACK);

FontMetrics fm = g.getFontMetrics();

int textHeight = fm.getHeight();

String s;

g.drawLine(TOPLEFT\_BORDER, TOPLEFT\_BORDER, TOPLEFT\_BORDER,

TOPLEFT\_BORDER + plotBound.height);

g.drawLine(TOPLEFT\_BORDER, TOPLEFT\_BORDER + plotBound.height,

TOPLEFT\_BORDER + plotBound.width, TOPLEFT\_BORDER

+ plotBound.height);

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

g.drawLine(TOPLEFT\_BORDER + i \* xStepPixel, TOPLEFT\_BORDER

+ plotBound.height - Y\_MARK, TOPLEFT\_BORDER + i

\* xStepPixel, TOPLEFT\_BORDER + plotBound.height + Y\_MARK);

s = String.valueOf(Math.rint(xStep \* i \* 1000) / 1000);

g.drawString(s, TOPLEFT\_BORDER + i \* xStepPixel - fm.stringWidth(s)

/ 2, TOPLEFT\_BORDER + plotBound.height + Y\_MARK

+ textHeight);

g.drawLine(TOPLEFT\_BORDER - X\_MARK, TOPLEFT\_BORDER

+ plotBound.height - yStepPixel \* i, TOPLEFT\_BORDER

+ X\_MARK, TOPLEFT\_BORDER + plotBound.height - yStepPixel

\* i);

s = String.valueOf(Math.rint(yStep \* i \* 1000) / 1000);

g.drawString(String.valueOf(Math.rint(yStep \* i \* 1000) / 1000),

TOPLEFT\_BORDER - X\_MARK \* 2 - fm.stringWidth(s),

TOPLEFT\_BORDER + plotBound.height - i \* yStepPixel

+ textHeight / 4);

}

}

/\*\*

\* Встановлює координати X

\*

\* @param x

\* координати X

\*/

public void setX(double[] x) {

this.x = x;

}

/\*\*

\* Встановл.є координати Y

\*

\* @param y

\* координати Y

\*/

public void setY(double[] y) {

this.y = y;

}

/\*\*

\* Обчислює розміри для графіка

\*

\* @param coord1

\* координати x

\* @param coord2

\* координати y

\* @param bounds

\* межі малювання

\*/

private void getRes(double[] coord1, double[] coord2, Rectangle bounds) {

int maxX = getMaxValuePos(coord1);

int maxY = getMaxValuePos(coord2);

int x = getPlotBound(bounds.width);

int y = getPlotBound(bounds.height);

xStep = coord1[maxX] / coord1.length;

yStep = coord2[maxY] / coord2.length;

bubbleR = y \* 0.1;

scaleRatio = (double) 1 / coord2[maxY];

plotBound = new Rectangle(TOPLEFT\_BORDER, TOPLEFT\_BORDER, x, y);

xStepPixel = (int) ((plotBound.width - bubbleR \* scaleRatio

\* coord2[maxX]) / (coord1.length));

yStepPixel = (int) ((plotBound.height - bubbleR) / (coord2.length));

}

/\*\*

\* Зменшує область малювання на розмір відступу

\*

\* @param res

\* розмір

\* @return новий розмір

\*/

private int getPlotBound(int res) {

return res - TOPLEFT\_BORDER - 2 \* BORDER;

}

/\*\*

\* Знаходить позицію найбільшого елемента

\*

\* @param array

\* масив для пошуку

\* @return позиція найбільшого елемента

\*/

private int getMaxValuePos(double[] array) {

int pos = 0;

double maxValue = array[pos];

for (int i = 1; i < array.length; i++) {

if (array[i] > maxValue) {

maxValue = array[i];

pos = i;

}

}

return pos;

}

}

package diagram\_builder;

/\*\*

\* Виконавчий клас

\*

\* @author Стремецький Олексій

\*

\*/

public class Main {

/\*\*

\* Виконавчий метод, з нього почианється виконання програми

\*

\* @param args

\* параметри передені через командну стрічку

\*/

public static void main(String[] args) {

new DiagramBuilderFrame();

}

}

ДОДАТОК Б. СТРУКТУРА ПРОЕКТУ

