# *Національний технічний університет України*

***«Київський політехнічний інститут»***

#### ***Факультет інформатики та обчислювальної техніки***

### ***Кафедра обчислювальної техніки***

## 

Курсова робота

*з дисципліни "****Програмна інженерія****"*

***Виконав: Куцовол Віктор Вікторович***

***Факультет***  *ІОТ*

***Група*** *ІО-82,* ***Залікова книжка №*** *8213*

***Номер технічного завдання*** *– 5G*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(підпис керівника)*

***Київ - 2010р.***

Зміст

|  |  |  |
| --- | --- | --- |
|  | Призначення розроблюваного об’єкта. | 3 |
|  | Вхідні дані для розробки. | 3 |
|  | Функціональність. | 3 |
|  | Вимоги до реалізації. | 4 |
|  | Перелік текстової і графічної документації. | 4 |

#### **Призначення розроблюваного об’єкту.**

В курсовій роботі нам необхідно створити програму візуалізації табличних даних у вигляді діаграми. Практичне застосування даного автомата можливе в області обчислювальної техніки.

#### **Вхідні дані.**

В якості вхідних даних виступає таблиця, що знаходиться у CSV-файлі.

Вірний формат вихідного CSV-файлу в даному проекті наступний:

* Файл складається з необмеженої кількості рядків.
* Роздільні знаки конфігуруються за замовчуванням або користувачем.

Таблиця 1.1. Варіант завдання.

|  |  |  |
| --- | --- | --- |
| № | **Назва діаграми** | **Вид редагування** |
| **1** | Кругова | Редагування за допомогою миші в графічній області з синхронізацією табличної області. |

#### **Функціональність.**

Програма візуалізації має містити наступну функціональність:

* можливість завантаження/збереження даних у табличний формі в форматі CSV;
* синтаксичний розбір і верифікація формату CSV з відбудовою внутрішньої моделі даних, в разі помилок – формування виключення (Exception);
* інтерфейс користувача (на основі компонентів бібліотеки SWING), який містить дві основні області – табличну і графічну, а також допоміжні компоненти – меню, панелі, кнопки, діалоги, тощо. Таблична область містить дані, що завантажені з CSV-файлу, а графічна – їх інтерпретацію у вигляді діаграми;
* можливість редагування даних;
* можливість збереження результату графічної інтерпретації – діаграми у вигляді файлу формату JPEG, PNG, GIF;
* можливість одночасної роботи з декількома діаграмами.

#### **Вимоги до реалізації.**

Вимоги до реалізації:

* мова програмування Java з використанням 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;
* обґрунтована насиченість інтерфейсу елементами позитивно впливає на оцінку;
* можливість роботи програми з більше ніж з однією серією даних (в одному чи різних CSV-файлах) позитивно впливає на оцінку.

#### **Перелік текстової документації.**

1. Титульний лист
2. Технічне завдання
3. Пояснювальна записка

Пояснювальна записка

Зміст

|  |  |  |
| --- | --- | --- |
| 1. | Вступ. | 6 |
| 2. | Модель інтерфейсу користувача. | 6 |
| 3. | UML-діаграми класів. | 9 |
| 4. | Роздруківка документації JavaDoc. | 11 |
| 5. | Вихідний код всіх класів. | 49 |
| 6. | Висновки. | 75 |
| 7. | Список літератури. | 75 |

#### **Вступ**.

У даній курсовій роботі необхідно створити программу візуалізації табличних даних у вигляді діаграми.

#### **Модель інтерфейсу користувача..**

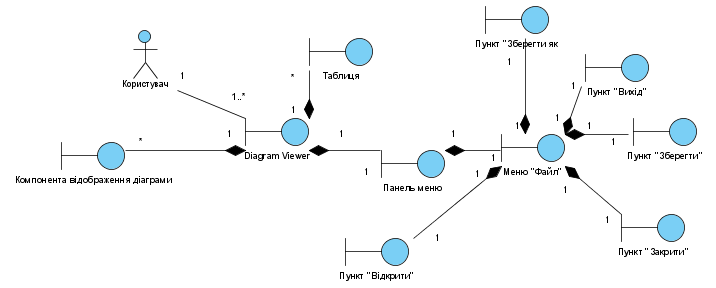
Призначення:

Головне вікно програми VisualDataManager використовується для завантаження \ збереження електронних таблиць, відображення їх як в табличному так і в графічному вигляді, а також для зміни даних в таблиці.

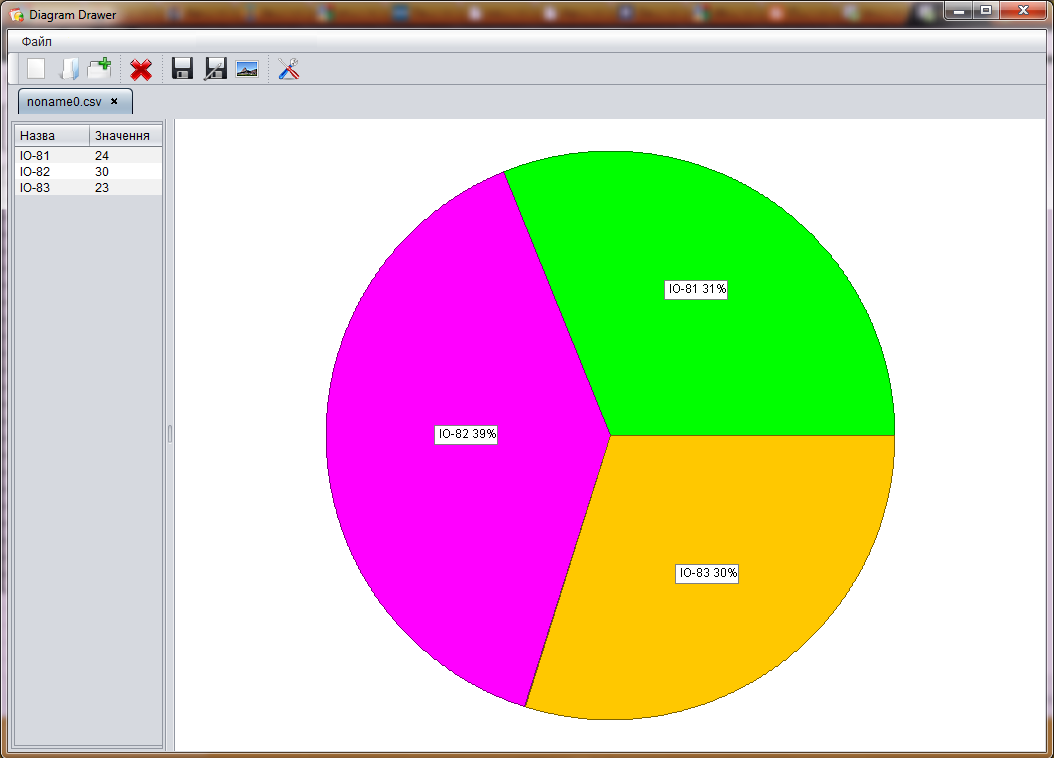
Початковий опис прецеденту інтерфейсу користувача:

1. Прецедент починається коли користувач запускає програму. [*Кнопки збереження необхідно зробити неактивними, а таблицю та графік приховати для зменшення завантаженості інтерфейсу*]
2. Користувач може виконати одну з наступних дій:
   1. Відкрити таблицю з файлу. [*При натисненні на кнопку запустити файловий діалог з шляхом, який використовувався минулого разу, в панелі стану вивести підказку про призначення кнопки.*]{*При першому запуску програми шлях файлового діалогу має дорівнювати шляху даних по замовчуванню.*}*(Виконується в середньому в 25% випадків.)*
   2. Зберегти відкриту таблицю з перезаписом в той же файл. [В панелі стану вивести підказку про призначенняя кнопки. ] {*Таблица должна иметь фиксированную ширину и быть выровнена по левому краю окна. Количество отображаемых ячеек должно содержать в среднем 5 шт.*} *(Виконується в середньому в 22% випадків)*
   3. Зберегти відкриту таблицю завданням шляху збереження. [*При натисненні на кнопку запустити файловий діалог з шляхом, який використовувався минулого разу, в панелі стану вивести підказку про призначення кнопки.*] { *При першому запуску програми шлях файлового діалогу має дорівнювати шляху даних по замовчуванню* }*( Виконується в середньому в 10% випадків)*
   4. Змінити значення даних в таблиці, перетягуючи границі даних на графіку. [*При перетягуванні даних відображаються х значення.*] {*Графік має не фіксований розмір, вирівняний по всій клієнтській області вікна.*}*( Виконується в середньому в 65% випадків)*
   5. Зберегти зображення відкритого графіку у вигляді картинки. [*При натисненні на кнопку запустити файловий діалог з шляхом, який використовувався минулого разу, в панелі стану вивести підказку про призначення кнопки.*] { *При першому запуску програми шлях файлового діалогу має дорівнювати шляху даних по замовчуванню* }*( Виконується в середньому в 10% випадків)*
3. Прецедент закінчується коли користувач закінчує роботу з програмою.

Діаграма граничних класів

****

Зовнішній вигляд програми

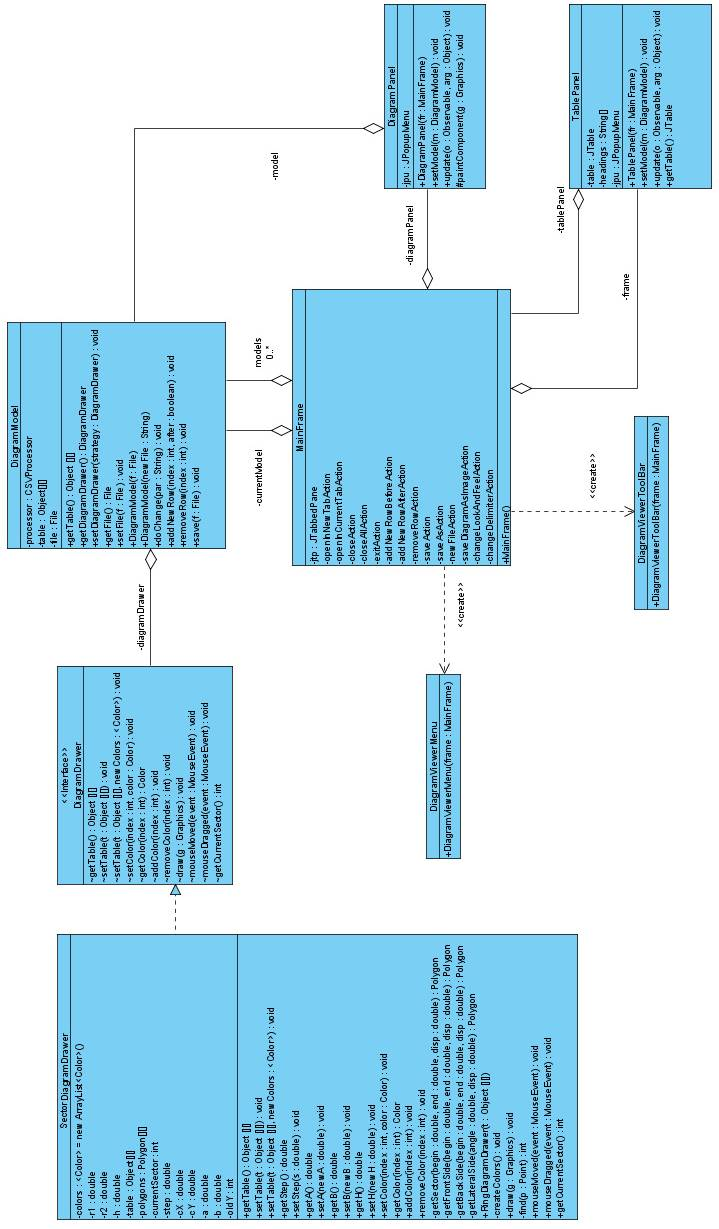
****

Компоненти SWING, що використовуються

|  |  |
| --- | --- |
| Diagram Viewer | java.awt.JFrame |
| Компонента відображення діаграми | javax.swing.JPanel |
| Таблиця | javax.swing.JTable |
| Панель меню | javax.swing.JMenuBar |
| Меню «Файл» | javax.swing.JMenu |
| Пункт «Вихід» | javax.swing.JMenuItem |
| Пункт «Закрити все» | javax.swing.JMenuItem |
| Пункт «Закрити» | javax.swing.|JMenuItem |
| Пункт «Відкрити файл» | javax.swing.JMenuItem |
| Пункт «Відкрити файл у новій вкладці» | javax.swing.JMenuItem |

**3. UML-Діаграми класів**

|  |
| --- |
|  |
| sc3 |
| sc1 |



1. **Роздруківка документації JavaDoc.**

**DiagramViewer**

**Abstract**

## 1. Package com.­csv

[Exception CSVParseException](file:///D:\KPI\EclDocs\DiagramViewer\index.html#class-com.csv.CSVParseException)

[Class CSVParseThread](file:///D:\KPI\EclDocs\DiagramViewer\index.html#class-com.csv.CSVParseThread)

[Class CSVProcessor](file:///D:\KPI\EclDocs\DiagramViewer\index.html#class-com.csv.CSVProcessor)

**Exception CSVParseException**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\index.html#d0e25)

[CSVParseException()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#constructor-com.csv.CSVParseException)

[CSVParseException(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#constructor-com.csv.CSVParseException-java.lang.String)

Клас виключення розбору CSV файлів.

**Synopsis**

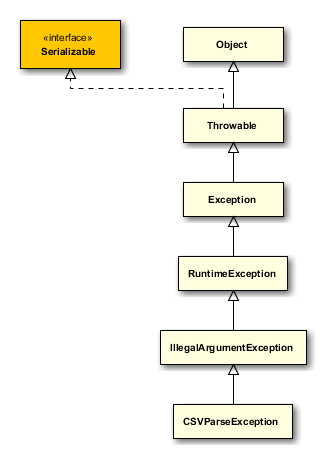
**public** **class** CSVParseException **extends** IllegalArgumentException {

  // Public Constructors

**public** CSVParseException();  
  **public** CSVParseException(String message);  
}

*Author*

Куцовол Віктор



**CSVParseException()**

**public** CSVParseException();

Конструктор, що створює виключення CSVParseException з порожнім деталізуючим повідомленням.

**CSVParseException(String)**

**public** CSVParseException(String message);

Конструктор, що створює виключення CSVParseException з власним деталізуючим повідомленням.

|  |  |
| --- | --- |
| Parameters | |
| message | деталізуюче повідомлення |

**Class CSVParseThread**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\index.html#d0e144)

[CSVParseThread(CSVProcessor)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#constructor-com.csv.CSVParseThread-com.csv.CSVProcessor)

[getTable()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVParseThread.getTable)

[getThread()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVParseThread.getThread)

[run()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVParseThread.run)

Клас потоку розбору CSV файла.

**Synopsis**

**public** **class** CSVParseThread **implements** Runnable {

  // Public Constructors

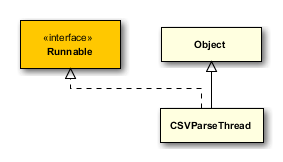
**public** CSVParseThread(CSVProcessor pro);

  // Public Methods

**public** **final** Object[][] getTable();  
  **public** **final** Thread getThread();  
  **public** **final** void run();  
}

*Author*

Куцовол Віктор



**CSVParseThread(CSVProcessor)**

**public** CSVParseThread(CSVProcessor pro);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| pro | CSV процесор |

**getTable()**

**public** **final** Object[][] getTable();

Повертає результуючу таблицю.

|  |  |
| --- | --- |
| Parameters | |
| *return* | результуючу таблицю |

**getThread()**

**public** **final** Thread getThread();

Повертає потік.

|  |  |
| --- | --- |
| Parameters | |
| *return* | потік |

**run()**

**public** **final** void run();

**Specified by:** Method run in interface Runnable

Виконується при запуску потоку.

**Class CSVProcessor**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\index.html#d0e386)

[CSVProcessor(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#constructor-com.csv.CSVProcessor-java.lang.String)

[deserialize()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.deserialize)

[deserialize(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.deserialize-java.lang.String)

[getCsvStrings()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.getCsvStrings)

[getDelimiter()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.getDelimiter)

[load(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.load-java.lang.String)

[parse()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.parse)

[save(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.save-java.lang.String)

[save(String, Object[][])](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.save-java.lang.String-java.lang.Object_A_A)

[serialize()](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.serialize)

[serialize(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.serialize-java.lang.String)

[setDelimiter(String)](file:///D:\KPI\EclDocs\DiagramViewer\index.html#method-com.csv.CSVProcessor.setDelimiter-java.lang.String)

Оброблювач CSV файлів.

**Synopsis**

**public** **class** CSVProcessor {

  // Public Constructors

**public** CSVProcessor(String del);

  // Public Methods

**public** **final** void deserialize()

**throws**Exception;  
  **public** **final** void deserialize(String fileName)

**throws**Exception;  
  **public** **final** ArrayList<String> getCsvStrings();  
  **public** **final** String getDelimiter();  
  **public** **final** void load(String fileName)

**throws**IOException;  
  **public** **synchronized** **final** Object[][] parse();  
  **public** **final** void save(String fileName)

**throws**IOException;  
  **public** **final** void save(String fileName, Object[][] table)

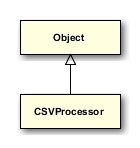
**throws**IOException;  
  **public** **final** void serialize()

**throws**Exception;  
  **public** **final** void serialize(String fileName)

**throws**Exception;  
  **public** **final** void setDelimiter(String del);  
}

*Author*

Куцовол Віктор



**CSVProcessor(String)**

**public** CSVProcessor(String del);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| del | відокремлювач |

**deserialize()**

**public** **final** void deserialize()

**throws**Exception;

Десеріалізує дані з файлу "serialized.dat".

**Exceptions**

Exception

виникає внаслідок помилки десеріалізації

**deserialize(String)**

**public** **final** void deserialize(String fileName)

**throws**Exception;

Десеріалізує дані з файлу.

|  |  |
| --- | --- |
| Parameters | |
| fileName | ім'я файлу для десеріалізації |

**Exceptions**

Exception

виникає внаслідок помилки десеріалізації

**getCsvStrings()**

**public** **final** ArrayList<String> getCsvStrings();

Повертає агрегований об'єкт.

|  |  |
| --- | --- |
| Parameters | |
| *return* | агрегований об'єкт класу ArrayList |

**getDelimiter()**

**public** **final** String getDelimiter();

Повертає відокремлювач.

|  |  |
| --- | --- |
| Parameters | |
| *return* | відокремлювач |

**load(String)**

**public** **final** void load(String fileName)

**throws**IOException;

Вивантажує з файлу рядки в агрегований об'єкт.

|  |  |
| --- | --- |
| Parameters | |
| fileName | ім'я файлу |

**Exceptions**

IOException

виникає внаслідок помилки вводу/виводу

**parse()**

**public** **synchronized** **final** Object[][] parse();

Виконує розбір даних агрегованого об'єкту.

|  |  |
| --- | --- |
| Parameters | |
| *return* | двовимірний масив |

**save(String)**

**public** **final** void save(String fileName)

**throws**IOException;

Завантажує в файл рядки з агрегованого об'єкту.

|  |  |
| --- | --- |
| Parameters | |
| fileName | ім'я файлу |

**Exceptions**

IOException

виникає внаслідок помилки вводу/виводу

**save(String, Object[][])**

**public** **final** void save(String fileName, Object[][] table)

**throws**IOException;

Зберігає в агрегований об'єкт двовимірний масив у форматі CSV, а потім завантажує в файл рядки з цього об'єкту.

|  |  |
| --- | --- |
| Parameters | |
| fileName | імя файлу |
| table | двовимірний масив |

**Exceptions**

IOException

виникає внаслідок помилки вводу/виводу

**serialize()**

**public** **final** void serialize()

**throws**Exception;

Серіалізує дані в файл "serialized.dat".

**Exceptions**

Exception

виникає внаслідок помилки серіалізації

**serialize(String)**

**public** **final** void serialize(String fileName)

**throws**Exception;

Серіалізує дані в файл.

|  |  |
| --- | --- |
| Parameters | |
| fileName | ім'я файлу для серіалізації |

**Exceptions**

Exception

виникає внаслідок помилки серіалізації

**setDelimiter(String)**

**public** **final** void setDelimiter(String del);

Встановлює відокремлювач.

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| del | | новий відокремлювач |
|  | | |

**2. Package com.gui**

**Class AddNewRowAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02.html#d0e1184)

[AddNewRowAction(MainFrame, String, boolean)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02.html#constructor-com.gui.AddNewRowAction-com.gui.MainFrame-java.lang.String-boolean)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02.html#method-com.gui.AddNewRowAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує додавання нового рядка.

**Synopsis**

**public** **class** AddNewRowAction **extends** AbstractAction {

  // Public Constructors

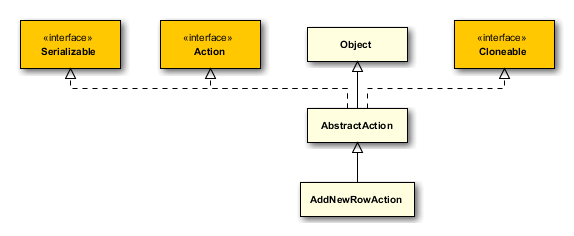
**public** AddNewRowAction(MainFrame fr, String name, boolean after);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**AddNewRowAction(MainFrame, String, boolean)**

**public** AddNewRowAction(MainFrame fr, String name, boolean after);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |
| after | ознака додавання після позиції рядка |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.2.** Class ButtonTabComponent

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s02.html#d0e1373)

[ButtonTabComponent(MainFrame)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s02.html#constructor-com.gui.ButtonTabComponent-com.gui.MainFrame)

Клас, що використовується як tabComponent. Створює JLabel, що використовується для закриття вкладки

**Synopsis**

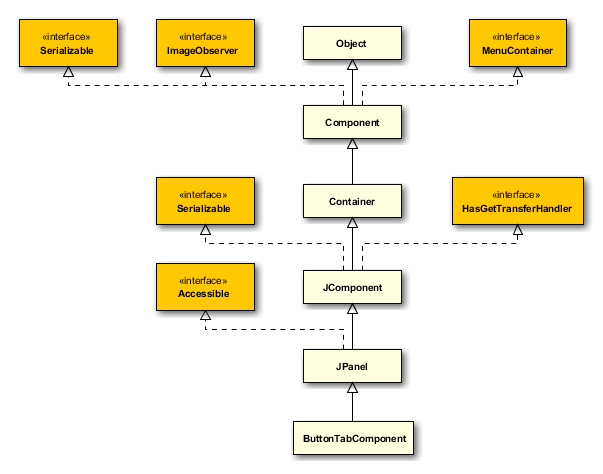
**public** **class** ButtonTabComponent **extends** JPanel {

  // Public Constructors

**public** ButtonTabComponent(MainFrame fr);  
}

*Author*

Куцовол Віктор



**ButtonTabComponent(MainFrame)**

**public** ButtonTabComponent(MainFrame fr);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |

### **2.3.** **Class CSVFileFilter**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s03.html#d0e1467)

[accept(File)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s03.html#method-com.gui.CSVFileFilter.accept-java.io.File)

[getDescription()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s03.html#method-com.gui.CSVFileFilter.getDescription)

Фільтер CSV файлів.

**Synopsis**

**public** **class** CSVFileFilter **extends** FileFilter {

  // Public Constructors

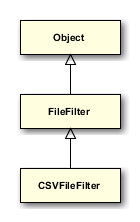
**public** CSVFileFilter();

  // Public Methods

**public** **final** boolean accept(File f);  
  **public** **final** String getDescription();  
}

*Author*

Куцовол Віктор



**accept(File)**

**public** **final** boolean accept(File f);

Повертає значення true, якщо в якості параметра йому було передано CSV файл.

|  |  |
| --- | --- |
| Parameters | |
| f | файл |
| *return* | true, якщо в якості параметра йому було передано CSV файл |

**getDescription()**

**public** **final** String getDescription();

Повертає опис фільтра.

|  |  |
| --- | --- |
| Parameters | |
| *return* | рядок опису |

### **2.4.** **Class ChangeDelimiterAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s04.html#d0e1637)

[ChangeDelimiterAction(MainFrame, String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s04.html#constructor-com.gui.ChangeDelimiterAction-com.gui.MainFrame-java.lang.String)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s04.html#method-com.gui.ChangeDelimiterAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує зміну відокремлювача CSV файлу.

**Synopsis**

**public** **class** ChangeDelimiterAction **extends** AbstractAction {

  // Public Constructors

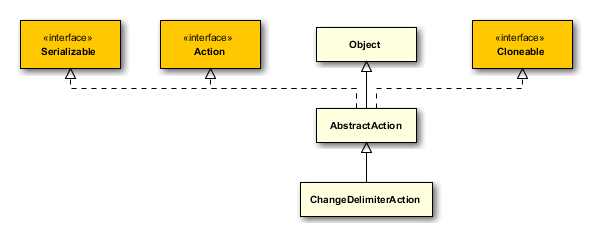
**public** ChangeDelimiterAction(MainFrame fr, String name);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**ChangeDelimiterAction(MainFrame, String)**

**public** ChangeDelimiterAction(MainFrame fr, String name);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.5.** **Class ChangeLookAndFeelAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s05.html#d0e1809)

[ChangeLookAndFeelAction(MainFrame, String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s05.html#constructor-com.gui.ChangeLookAndFeelAction-com.gui.MainFrame-java.lang.String)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s05.html#method-com.gui.ChangeLookAndFeelAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує зміну стилю вікна.

**Synopsis**

**public** **class** ChangeLookAndFeelAction **extends** AbstractAction {

  // Public Constructors

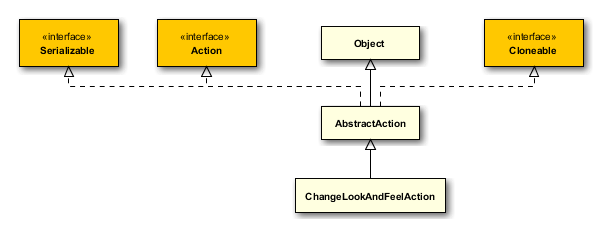
**public** ChangeLookAndFeelAction(MainFrame fr, String name);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**ChangeLookAndFeelAction(MainFrame, String)**

**public** ChangeLookAndFeelAction(MainFrame fr, String name);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.6.** **Class CloseAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s06.html#d0e1981)

[CloseAction(MainFrame, String, boolean)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s06.html#constructor-com.gui.CloseAction-com.gui.MainFrame-java.lang.String-boolean)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s06.html#method-com.gui.CloseAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує закриття файлу.

**Synopsis**

**public** **class** CloseAction **extends** AbstractAction {

  // Public Constructors

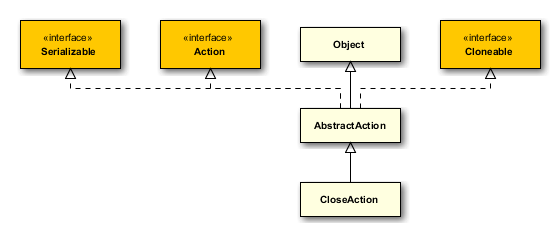
**public** CloseAction(MainFrame fr, String name, boolean closeAll);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**CloseAction(MainFrame, String, boolean)**

**public** CloseAction(MainFrame fr, String name, boolean closeAll);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |
| closeAll | ознака закриття усіх вкладок |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.7.** **Interface DiagramDrawer**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#d0e2170)

[addColor(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.addColor-int)

[draw(Graphics)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.draw-java.awt.Graphics)

[getColor(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.getColor-int)

[getCurrentSector()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.getCurrentSector)

[getTable()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.getTable)

[mouseDragged(MouseEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.mouseDragged-java.awt.event.MouseEvent)

[mouseMoved(MouseEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.mouseMoved-java.awt.event.MouseEvent)

[removeColor(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.removeColor-int)

[setColor(int, Color)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.setColor-int-java.awt.Color)

[setTable(Object[][])](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.setTable-java.lang.Object_A_A)

[setTable(Object[][], ArrayList<Color>)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.setTable-java.lang.Object_A_A-java.util.ArrayList)

Інтерфейс відображувача діаграм.

**Synopsis**

**public** **interface** DiagramDrawer {

  // Public Methods

**public** void addColor(int index);  
  **public** void draw(Graphics g);  
  **public** Color getColor(int index);  
  **public** int getCurrentSector();  
  **public** Object[][] getTable();  
  **public** void mouseDragged(MouseEvent event);  
  **public** void mouseMoved(MouseEvent event);  
  **public** void removeColor(int index);  
  **public** void setColor(int index, Color color);  
  **public** void setTable(Object[][] t);  
  **public** void setTable(Object[][] t, ArrayList<Color> newColors);  
}

*Author*

Куцовол Віктор



**addColor(int)**

**public** void addColor(int index);

Додає колір сектору.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектру |

**draw(Graphics)**

**public** void draw(Graphics g);

Малює діаграму використовуючи даний графічний контекст.

|  |  |
| --- | --- |
| Parameters | |
| g | графічний контекст |

**getColor(int)**

**public** Color getColor(int index);

Повертає колір сектору за його індексом.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектору |
| *return* | колір сектору |

**getCurrentSector()**

**public** int getCurrentSector();

Повертає поточний сектор.

|  |  |
| --- | --- |
| Parameters | |
| *return* | the currentSector поточний сектор |

**getTable()**

**public** Object[][] getTable();

Повертає таблицю значень.

|  |  |
| --- | --- |
| Parameters | |
| *return* | таблиця значень. |

**mouseDragged(MouseEvent)**

**public** void mouseDragged(MouseEvent event);

Виконується при одночасному натисканні та русі миші.

|  |  |
| --- | --- |
| Parameters | |
| event | подія миші |

**mouseMoved(MouseEvent)**

**public** void mouseMoved(MouseEvent event);

Виконується при русі миші.

|  |  |
| --- | --- |
| Parameters | |
| event | подія миші |

**removeColor(int)**

**public** void removeColor(int index);

Видаляє колір сектору.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектру |

**setColor(int, Color)**

**public** void setColor(int index, Color color);

Встановлює колір сектору за його індексом.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектору |
| color | колір сектору |

**setTable(Object[][])**

**public** void setTable(Object[][] t);

Встановлює таблицю значень.

|  |  |
| --- | --- |
| Parameters | |
| t | таблиця значень |

**setTable(Object[][], ArrayList<Color>)**

**public** void setTable(Object[][] t, ArrayList<Color> newColors);

Встановлює таблицю значень і список кольорів секторів.

|  |  |
| --- | --- |
| Parameters | |
| t | таблиця значень |
| newColors | список кольорів секторів |

### **2.8.** **Class DiagramModel**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#d0e2843)

[DiagramModel(File)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#constructor-com.gui.DiagramModel-java.io.File)

[DiagramModel(String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#constructor-com.gui.DiagramModel-java.lang.String)

[addNewRow(int, boolean)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.addNewRow-int-boolean)

[doChange(String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.doChange-java.lang.String)

[getDiagramDrawer()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.getDiagramDrawer)

[getFile()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.getFile)

[getTable()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.getTable)

[removeRow(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.removeRow-int)

[save(File)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.save-java.io.File)

[setDiagramDrawer(DiagramDrawer)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.setDiagramDrawer-com.gui.DiagramDrawer)

[setFile(File)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s08.html#method-com.gui.DiagramModel.setFile-java.io.File)

Клас моделі діаграми.

**Synopsis**

**public** **class** DiagramModel **extends** Observable {

  // Public Constructors

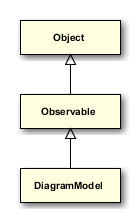
**public** DiagramModel(File f);  
  **public** DiagramModel(String newFile);

  // Public Methods

**public** **final** void addNewRow(int index, boolean after);  
  **public** **final** void doChange(String par);  
  **public** **final** DiagramDrawer getDiagramDrawer();  
  **public** **final** File getFile();  
  **public** **final** Object[][] getTable();  
  **public** **final** void removeRow(int index);  
  **public** **final** void save(File f);  
  **public** **final** void setDiagramDrawer(DiagramDrawer strategy);  
  **public** **final** void setFile(File f);  
}

*Author*

Куцовол Віктор



**DiagramModel(File)**

**public** DiagramModel(File f);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| f | CSV файл |

**DiagramModel(String)**

**public** DiagramModel(String newFile);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| newFile | ім'я нового файлу |

**addNewRow(int, boolean)**

**public** **final** void addNewRow(int index, boolean after);

Додає новий рядок.

|  |  |
| --- | --- |
| Parameters | |
| index | позиція рядка |
| after | ознака додавання після позиції рядка |

**doChange(String)**

**public** **final** void doChange(String par);

Повідомляє про зміни.

|  |  |
| --- | --- |
| Parameters | |
| par | характер змін |

**getDiagramDrawer()**

**public** **final** DiagramDrawer getDiagramDrawer();

Повертає стратегію відображувача діаграм.

|  |  |
| --- | --- |
| Parameters | |
| *return* | стратегію відображувача діаграм |

**getFile()**

**public** **final** File getFile();

Повертає файл.

|  |  |
| --- | --- |
| Parameters | |
| *return* | файл |

**getTable()**

**public** **final** Object[][] getTable();

Повертає таблицю значень.

|  |  |
| --- | --- |
| Parameters | |
| *return* | таблицю знчень |

**removeRow(int)**

**public** **final** void removeRow(int index);

Видаляє рядок.

|  |  |
| --- | --- |
| Parameters | |
| index | позиція рядка |

**save(File)**

**public** **final** void save(File f);

Зберігає файл.

|  |  |
| --- | --- |
| Parameters | |
| f | файл |

**setDiagramDrawer(DiagramDrawer)**

**public** **final** void setDiagramDrawer(DiagramDrawer strategy);

Встановлює стратегію відображувача діаграм.

|  |  |
| --- | --- |
| Parameters | |
| strategy | стратегія відображувача діаграм |

**setFile(File)**

**public** **final** void setFile(File f);

Встановлює файл.

|  |  |
| --- | --- |
| Parameters | |
| f | файл |

### **2.9.** **Class DiagramPanel**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s09.html#d0e3517)

[DiagramPanel(MainFrame)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s09.html#constructor-com.gui.DiagramPanel-com.gui.MainFrame)

[setModel(DiagramModel)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s09.html#method-com.gui.DiagramPanel.setModel-com.gui.DiagramModel)

[update(Observable, Object)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s09.html#method-com.gui.DiagramPanel.update-java.util.Observable-java.lang.Object)

Клас панелі діаграми.

**Synopsis**

**public** **class** DiagramPanel **extends** JPanel

**implements** Observer {

  // Public Constructors

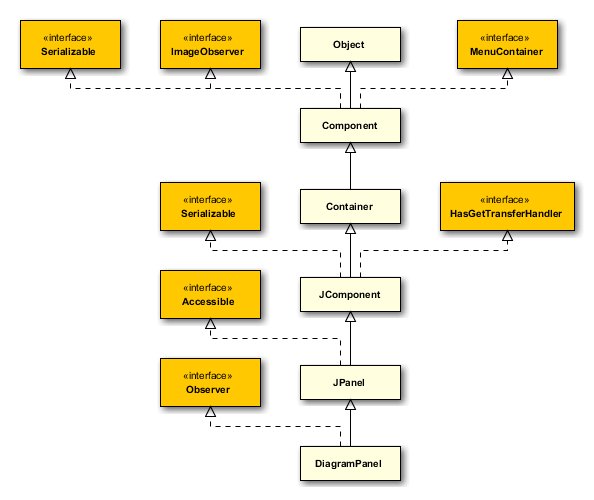
**public** DiagramPanel(MainFrame fr);

  // Public Methods

**public** **final** void setModel(DiagramModel m);  
  **public** **final** void update(Observable o, Object arg);  
}

*Author*

Куцовол Віктор



**DiagramPanel(MainFrame)**

**public** DiagramPanel(MainFrame fr);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |

**setModel(DiagramModel)**

**public** **final** void setModel(DiagramModel m);

Встановлює модель діаграми.

|  |  |
| --- | --- |
| Parameters | |
| m | модель діаграми |

**update(Observable, Object)**

**public** **final** void update(Observable o, Object arg);

**Specified by:** Method update in interface Observer

Виконується, коли змінюється об'єкт, за яким ведеться нагляд.

|  |  |
| --- | --- |
| Parameters | |
| o | об'єкт, за яким ведеться нагляд |
| arg | значення, яке пересилається від об'єкта, за яким ведеться нагляд |

### **2.10.** **Class DiagramViewerMenu**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s10.html#d0e3760)

[DiagramViewerMenu(MainFrame)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s10.html#constructor-com.gui.DiagramViewerMenu-com.gui.MainFrame)

Клас меню.

**Synopsis**

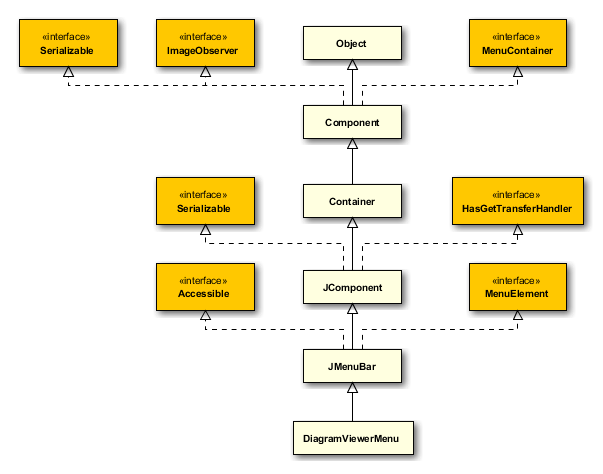
**public** **class** DiagramViewerMenu **extends** JMenuBar {

  // Public Constructors

**public** DiagramViewerMenu(MainFrame frame);  
}

*Author*

Куцовол Віктор



**DiagramViewerMenu(MainFrame)**

**public** DiagramViewerMenu(MainFrame frame);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| frame | посилання на головне вікно програми |

### **2.11.** **Class DiagramViewerToolBar**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s11.html#d0e3854)

[DiagramViewerToolBar(MainFrame)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s11.html#constructor-com.gui.DiagramViewerToolBar-com.gui.MainFrame)

Клас панелі інструментів.

**Synopsis**

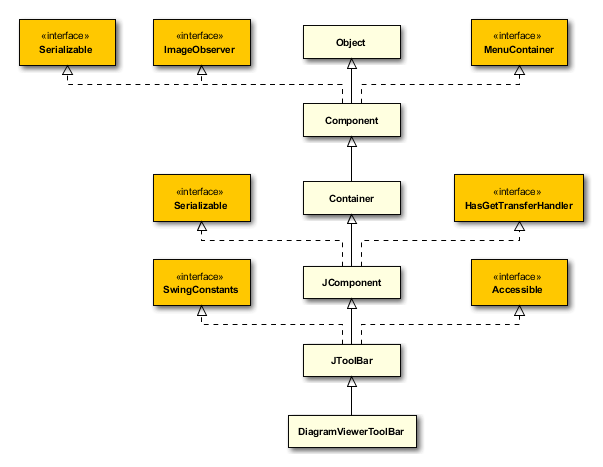
**public** **class** DiagramViewerToolBar **extends** JToolBar {

  // Public Constructors

**public** DiagramViewerToolBar(MainFrame frame);  
}

*Author*

Куцовол Віктор



**DiagramViewerToolBar(MainFrame)**

**public** DiagramViewerToolBar(MainFrame frame);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| frame | посилання на головне вікно програми |

### **2.12.** **Class ExitAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s12.html#d0e3948)

[ExitAction(MainFrame, String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s12.html#constructor-com.gui.ExitAction-com.gui.MainFrame-java.lang.String)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s12.html#method-com.gui.ExitAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує закриття програми.

**Synopsis**

**public** **class** ExitAction **extends** AbstractAction {

  // Public Constructors

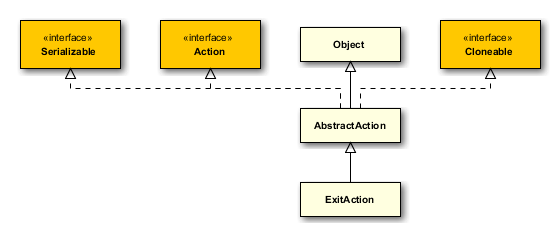
**public** ExitAction(MainFrame fr, String name);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**ExitAction(MainFrame, String)**

**public** ExitAction(MainFrame fr, String name);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.13.** **Class JPEGFileFilter**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s13.html#d0e4120)

[accept(File)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s13.html#method-com.gui.JPEGFileFilter.accept-java.io.File)

[getDescription()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s13.html#method-com.gui.JPEGFileFilter.getDescription)

Фільтер JPEG файлів.

**Synopsis**

**public** **class** JPEGFileFilter **extends** FileFilter {

  // Public Constructors

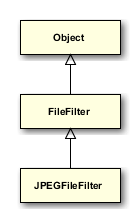
**public** JPEGFileFilter();

  // Public Methods

**public** **final** boolean accept(File f);  
  **public** **final** String getDescription();  
}

*Author*

Куцовол Віктор



**accept(File)**

**public** **final** boolean accept(File f);

Повертає значення true, якщо в якості параметра йому було передано JPEG файл.

|  |  |
| --- | --- |
| Parameters | |
| f | файл |
| *return* | true, якщо в якості параметра йому було передано JPEG файл |

**getDescription()**

**public** **final** String getDescription();

Повертає опис фільтра.

|  |  |
| --- | --- |
| Parameters | |
| *return* | рядок опису |

### **2.14.** **Class MainFrame**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#d0e4290)

[MainFrame()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#constructor-com.gui.MainFrame)

[getAddNewRowAfterAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getAddNewRowAfterAction)

[getAddNewRowBeforeAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getAddNewRowBeforeAction)

[getChangeDelimiterAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getChangeDelimiterAction)

[getChangeLookAndFeelAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getChangeLookAndFeelAction)

[getCloseAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getCloseAction)

[getCloseAllAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getCloseAllAction)

[getCurrentModel()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getCurrentModel)

[getDiagramPanel()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getDiagramPanel)

[getExitAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getExitAction)

[getJtp()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getJtp)

[getModels()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getModels)

[getNewFileAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getNewFileAction)

[getOpenInCurrentTabAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getOpenInCurrentTabAction)

[getOpenInNewTabAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getOpenInNewTabAction)

[getRemoveRowAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getRemoveRowAction)

[getSaveAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getSaveAction)

[getSaveAsAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getSaveAsAction)

[getSaveDiagramAsImageAction()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getSaveDiagramAsImageAction)

[getTablePanel()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.getTablePanel)

[setCurrentModel(DiagramModel)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s14.html#method-com.gui.MainFrame.setCurrentModel-com.gui.DiagramModel)

Клас головного вікна.

**Synopsis**

**public** **class** MainFrame **extends** JFrame {

  // Public Constructors

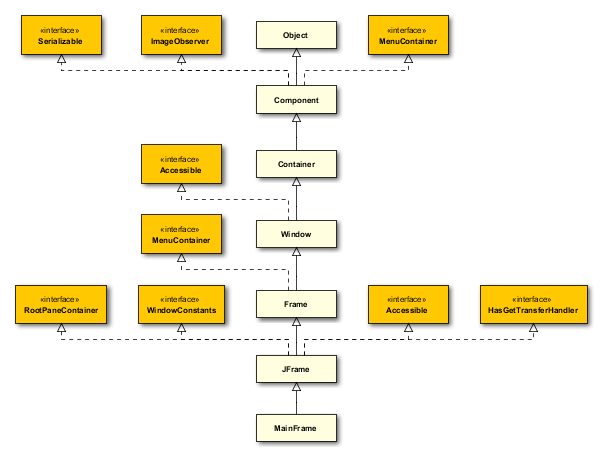
**public** MainFrame();

  // Public Methods

**public** **final** AddNewRowAction getAddNewRowAfterAction();  
  **public** **final** AddNewRowAction getAddNewRowBeforeAction();  
  **public** **final** ChangeDelimiterAction getChangeDelimiterAction();  
  **public** **final** ChangeLookAndFeelAction getChangeLookAndFeelAction();  
  **public** **final** CloseAction getCloseAction();  
  **public** **final** CloseAction getCloseAllAction();  
  **public** **final** DiagramModel getCurrentModel();  
  **public** **final** DiagramPanel getDiagramPanel();  
  **public** **final** ExitAction getExitAction();  
  **public** **final** JTabbedPane getJtp();  
  **public** **final** ArrayList<DiagramModel> getModels();  
  **public** **final** NewFileAction getNewFileAction();  
  **public** **final** OpenAction getOpenInCurrentTabAction();  
  **public** **final** OpenAction getOpenInNewTabAction();  
  **public** **final** RemoveRowAction getRemoveRowAction();  
  **public** **final** SaveAction getSaveAction();  
  **public** **final** SaveAction getSaveAsAction();  
  **public** **final** SaveDiagramAsImageAction getSaveDiagramAsImageAction();  
  **public** **final** TablePanel getTablePanel();  
  **public** **final** void setCurrentModel(DiagramModel model);  
}

*Author*

Куцовол Віктор



**MainFrame()**

**public** MainFrame();

Конструктор.

**getAddNewRowAfterAction()**

**public** **final** AddNewRowAction getAddNewRowAfterAction();

Повертає дію, що виконує додавання нового рядка після поточного.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує додавання нового рядка після поточного. |

**getAddNewRowBeforeAction()**

**public** **final** AddNewRowAction getAddNewRowBeforeAction();

Повертає дію, що виконує додавання нового рядка перед поточним.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує додавання нового рядка перед поточним |

**getChangeDelimiterAction()**

**public** **final** ChangeDelimiterAction getChangeDelimiterAction();

Повертає дію, що виконує зміну відокремлювача CSV файлу.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує зміну відокремлювача CSV файлу |

**getChangeLookAndFeelAction()**

**public** **final** ChangeLookAndFeelAction getChangeLookAndFeelAction();

Повертає дію, що виконує зміну стилю вікна.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує зміну стилю вікна |

**getCloseAction()**

**public** **final** CloseAction getCloseAction();

Повертає дію, що закриває файл.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що закриває файл |

**getCloseAllAction()**

**public** **final** CloseAction getCloseAllAction();

Повертає дію, що закриває усі файли.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що закриває усі файли |

**getCurrentModel()**

**public** **final** DiagramModel getCurrentModel();

Повертає поточну модель.

|  |  |
| --- | --- |
| Parameters | |
| *return* | поточну модель діаграми |

**getDiagramPanel()**

**public** **final** DiagramPanel getDiagramPanel();

Повертає панель діаграми.

|  |  |
| --- | --- |
| Parameters | |
| *return* | панель діаграми |

**getExitAction()**

**public** **final** ExitAction getExitAction();

Повертає дію, що виконує закриття програми.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує закриття програми |

**getJtp()**

**public** **final** JTabbedPane getJtp();

Повертає панель вкладок.

|  |  |
| --- | --- |
| Parameters | |
| *return* | панель вкладок |

**getModels()**

**public** **final** ArrayList<DiagramModel> getModels();

Повертає масив моделей діаграми.

|  |  |
| --- | --- |
| Parameters | |
| *return* | масив моделей діаграми |

**getNewFileAction()**

**public** **final** NewFileAction getNewFileAction();

Повертає дію, що виконує створення нового файлу.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує створення нового файлу |

**getOpenInCurrentTabAction()**

**public** **final** OpenAction getOpenInCurrentTabAction();

Повертає дію, що відкриває файл у поточній вкладці.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що відкриває файл у поточній вкладці |

**getOpenInNewTabAction()**

**public** **final** OpenAction getOpenInNewTabAction();

Повертає дію, що відкриває файл у новій вкладці.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що відкриває файл у новій вкладці |

**getRemoveRowAction()**

**public** **final** RemoveRowAction getRemoveRowAction();

Повертає дію, що виконує видалення рядка.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує видалення рядка |

**getSaveAction()**

**public** **final** SaveAction getSaveAction();

Повертає дію, що виконує зберігання у цьому ж файлі.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує зберігання у цьому ж файлі |

**getSaveAsAction()**

**public** **final** SaveAction getSaveAsAction();

Повертає дію, що виконує зберігання в іншому файлі.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує зберігання в іншому файлі |

**getSaveDiagramAsImageAction()**

**public** **final** SaveDiagramAsImageAction getSaveDiagramAsImageAction();

Повертає дію, що виконує зберігання діагрми у файлі формату JPEG.

|  |  |
| --- | --- |
| Parameters | |
| *return* | дію, що виконує зберігання діагрми у файлі формату JPEG |

**getTablePanel()**

**public** **final** TablePanel getTablePanel();

Повертає панель таблиці.

|  |  |
| --- | --- |
| Parameters | |
| *return* | панель таблиці |

**setCurrentModel(DiagramModel)**

**public** **final** void setCurrentModel(DiagramModel model);

Встановлює поточну модель діаграми.

|  |  |
| --- | --- |
| Parameters | |
| model | поточна модель діаграми |

### **2.15.** **Class NewFileAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s15.html#d0e5407)

[NewFileAction(MainFrame, String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s15.html#constructor-com.gui.NewFileAction-com.gui.MainFrame-java.lang.String)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s15.html#method-com.gui.NewFileAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує створення нового файлу.

**Synopsis**

**public** **class** NewFileAction **extends** AbstractAction {

  // Public Constructors

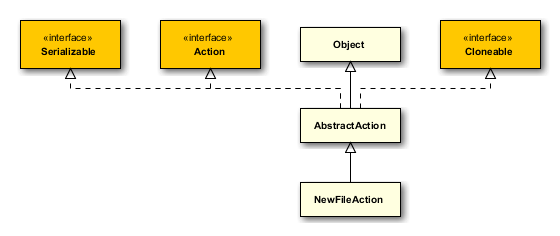
**public** NewFileAction(MainFrame fr, String name);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**NewFileAction(MainFrame, String)**

**public** NewFileAction(MainFrame fr, String name);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.16.** **Class OpenAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s16.html#d0e5579)

[OpenAction(MainFrame, String, boolean)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s16.html#constructor-com.gui.OpenAction-com.gui.MainFrame-java.lang.String-boolean)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s16.html#method-com.gui.OpenAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує відкриття файлу.

**Synopsis**

**public** **class** OpenAction **extends** AbstractAction {

  // Public Constructors

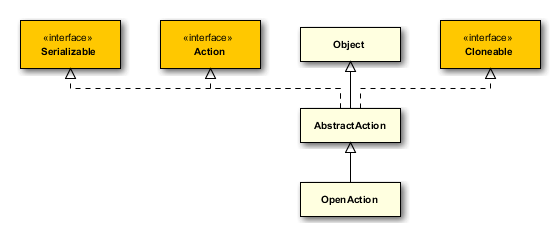
**public** OpenAction(MainFrame fr, String name, boolean inNewTab);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**OpenAction(MainFrame, String, boolean)**

**public** OpenAction(MainFrame fr, String name, boolean inNewTab);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |
| inNewTab | ознака відкриття у новій вкладці |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.17.** **Class RemoveRowAction**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s17.html#d0e5768)

[RemoveRowAction(MainFrame, String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s17.html#constructor-com.gui.RemoveRowAction-com.gui.MainFrame-java.lang.String)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s17.html#method-com.gui.RemoveRowAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує видалення рядка.

**Synopsis**

**public** **class** RemoveRowAction **extends** AbstractAction {

  // Public Constructors

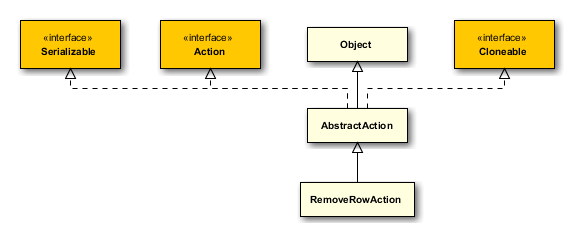
**public** RemoveRowAction(MainFrame fr, String name);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**RemoveRowAction(MainFrame, String)**

**public** RemoveRowAction(MainFrame fr, String name);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.18.** **Class SectorDiagramDrawer**

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#d0e5940)

[RingDiagramDrawer(Object[][])](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#constructor-com.gui.RingDiagramDrawer-java.lang.Object_A_A)

[addColor(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.addColor-int)

[draw(Graphics)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.draw-java.awt.Graphics)

[getA()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.getA)

[getB()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.getB)

[getColor(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.getColor-int)

[getCurrentSector()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.getCurrentSector)

[getStep()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.getStep)

[getTable()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.getTable)

[mouseDragged(MouseEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.mouseDragged-java.awt.event.MouseEvent)

[mouseMoved(MouseEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.mouseMoved-java.awt.event.MouseEvent)

[removeColor(int)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.removeColor-int)

[setA(double)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.setA-double)

[setB(double)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.setB-double)

[setColor(int, Color)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.setColor-int-java.awt.Color)

[setStep(double)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.setStep-double)

[setTable(Object[][])](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.setTable-java.lang.Object_A_A)

[setTable(Object[][], ArrayList<Color>)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s18.html#method-com.gui.RingDiagramDrawer.setTable-java.lang.Object_A_A-java.util.ArrayList)

Клас відображувача кільцевої діаграми.

**Synopsis**

**public** **class** SectorDiagramDrawer **implements** DiagramDrawer {

  // Public Constructors

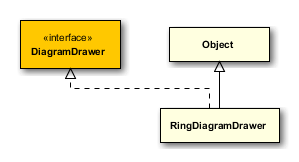
**public** RingDiagramDrawer(Object[][] t);

  // Public Methods

**public** **final** void addColor(int index);  
  **public** **final** void draw(Graphics g);  
  **public** **final** double getA();  
  **public** **final** double getB();  
  **public** **final** Color getColor(int index);  
  **public** **final** int getCurrentSector();  
  **public** **final** double getStep();  
  **public** **final** Object[][] getTable();  
  **public** **final** void mouseDragged(MouseEvent event);  
  **public** **final** void mouseMoved(MouseEvent event);  
  **public** **final** void removeColor(int index);  
  **public** **final** void setA(double newA);  
  **public** **final** void setB(double newB);  
  **public** **final** void setColor(int index, Color color);  
  **public** **final** void setStep(double s);  
  **public** **final** void setTable(Object[][] t);  
  **public** **final** void setTable(Object[][] t, ArrayList<Color> newColors);  
}

*Author*

Куцовол Віктор



**SectorDiagramDrawer(Object[][])**

**public** SectorDiagramDrawer(Object[][] t);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| t | таблиця значень |

**addColor(int)**

**public** **final** void addColor(int index);

**Specified by:** Method [addColor](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.addColor-int) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Додає колір сектору.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектру |

**draw(Graphics)**

**public** **final** void draw(Graphics g);

**Specified by:** Method [draw](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.draw-java.awt.Graphics) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Малює діаграму використовуючи даний графічний контекст.

|  |  |
| --- | --- |
| Parameters | |
| g | графічний контекст |

**getA()**

**public** **final** double getA();

Повертає коефіцієнт а еліпса.

|  |  |
| --- | --- |
| Parameters | |
| *return* | коефіцієнт а еліпса |

**getB()**

**public** **final** double getB();

Повертає коефіцієнт b еліпса.

|  |  |
| --- | --- |
| Parameters | |
| *return* | коефіцієнт b еліпса |

**getColor(int)**

**public** **final** Color getColor(int index);

**Specified by:** Method [getColor](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.getColor-int) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Повертає колір сектору за його індексом.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектору |
| *return* | колір сектору |

**getCurrentSector()**

**public** **final** int getCurrentSector();

**Specified by:** Method [getCurrentSector](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.getCurrentSector) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Повертає поточний сектор.

|  |  |
| --- | --- |
| Parameters | |
| *return* | the currentSector поточний сектор |

**getStep()**

**public** **final** double getStep();

Повертає крок малювання.

|  |  |
| --- | --- |
| Parameters | |
| *return* | крок малювання |

**getTable()**

**public** **final** Object[][] getTable();

**Specified by:** Method [getTable](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.getTable) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Повертає таблицю значень.

|  |  |
| --- | --- |
| Parameters | |
| *return* | таблиця значень. |

**mouseDragged(MouseEvent)**

**public** **final** void mouseDragged(MouseEvent event);

**Specified by:** Method [mouseDragged](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.mouseDragged-java.awt.event.MouseEvent) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Виконується при одночасному натисканні та русі миші.

|  |  |
| --- | --- |
| Parameters | |
| event | подія миші |

**mouseMoved(MouseEvent)**

**public** **final** void mouseMoved(MouseEvent event);

**Specified by:** Method [mouseMoved](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.mouseMoved-java.awt.event.MouseEvent) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Виконується при русі миші.

|  |  |
| --- | --- |
| Parameters | |
| event | подія миші |

**removeColor(int)**

**public** **final** void removeColor(int index);

**Specified by:** Method [removeColor](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.removeColor-int) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Видаляє колір сектору.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектру |

**setA(double)**

**public** **final** void setA(double newA);

Встановлює коефіцієнт а еліпса.

|  |  |
| --- | --- |
| Parameters | |
| newA | коефіцієнт а еліпса |

**setB(double)**

**public** **final** void setB(double newB);

Встановлює коефіцієнт b еліпса.

|  |  |
| --- | --- |
| Parameters | |
| newB | коефіцієнт b еліпса |

**setColor(int, Color)**

**public** **final** void setColor(int index, Color color);

**Specified by:** Method [setColor](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.setColor-int-java.awt.Color) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Встановлює колір сектору за його індексом.

|  |  |
| --- | --- |
| Parameters | |
| index | індекс сектору |
| color | колір сектору |

**setStep(double)**

**public** **final** void setStep(double s);

Встановлює крок малювання.

|  |  |
| --- | --- |
| Parameters | |
| s | крок малювання |

**setTable(Object[][])**

**public** **final** void setTable(Object[][] t);

**Specified by:** Method [setTable](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.setTable-java.lang.Object_A_A) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Встановлює таблицю значень.

|  |  |
| --- | --- |
| Parameters | |
| t | таблиця значень |

**setTable(Object[][], ArrayList<Color>)**

**public** **final** void setTable(Object[][] t, ArrayList<Color> newColors);

**Specified by:** Method [setTable](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html#method-com.gui.DiagramDrawer.setTable-java.lang.Object_A_A-java.util.ArrayList) in interface [DiagramDrawer](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s07.html)

Встановлює таблицю значень та список кольорів секторів.

|  |  |
| --- | --- |
| Parameters | |
| t | таблиця значень |
| newColors | список кольорів секторів |

### **2.19.** Class SaveAction

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s19.html#d0e7145)

[SaveAction(MainFrame, String, boolean)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s19.html#constructor-com.gui.SaveAction-com.gui.MainFrame-java.lang.String-boolean)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s19.html#method-com.gui.SaveAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує зберігання файлу.

**Synopsis**

**public** **class** SaveAction **extends** AbstractAction {

  // Public Constructors

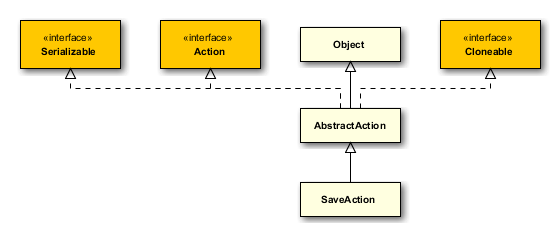
**public** SaveAction(MainFrame fr, String name, boolean saveAs);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**SaveAction(MainFrame, String, boolean)**

**public** SaveAction(MainFrame fr, String name, boolean saveAs);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |
| saveAs | флаг збереження файлу під іншим іменем. |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.20.** Class SaveDiagramAsImageAction

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s20.html#d0e7334)

[SaveDiagramAsImageAction(MainFrame, String)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s20.html#constructor-com.gui.SaveDiagramAsImageAction-com.gui.MainFrame-java.lang.String)

[actionPerformed(ActionEvent)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s20.html#method-com.gui.SaveDiagramAsImageAction.actionPerformed-java.awt.event.ActionEvent)

Клас, що виконує зберігання діагрми у файлі формату JPEG.

**Synopsis**

**public** **class** SaveDiagramAsImageAction **extends** AbstractAction {

  // Public Constructors

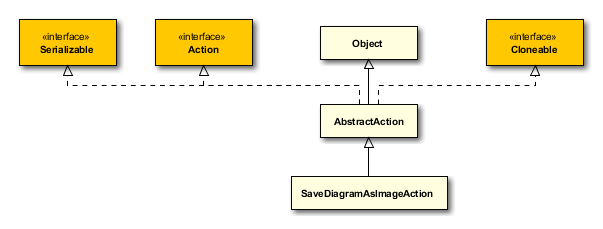
**public** SaveDiagramAsImageAction(MainFrame fr, String name);

  // Public Methods

**public** **final** void actionPerformed(ActionEvent e);  
}

*Author*

Куцовол Віктор



**SaveDiagramAsImageAction(MainFrame, String)**

**public** SaveDiagramAsImageAction(MainFrame fr, String name);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |
| name | назва дії |

**actionPerformed(ActionEvent)**

**public** **final** void actionPerformed(ActionEvent e);

Виконується при виникненні події.

|  |  |
| --- | --- |
| Parameters | |
| e | подія |

### **2.21.** Class TablePanel

[TablePanel(MainFrame)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s21.html#constructor-com.gui.TablePanel-com.gui.MainFrame)

[getTable()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s21.html#method-com.gui.TablePanel.getTable)

[setModel(DiagramModel)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s21.html#method-com.gui.TablePanel.setModel-com.gui.DiagramModel)

[update(Observable, Object)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s02s21.html#method-com.gui.TablePanel.update-java.util.Observable-java.lang.Object)

**public** **class** TablePanel **extends** JPanel

**implements** Observer {

  // Public Constructors

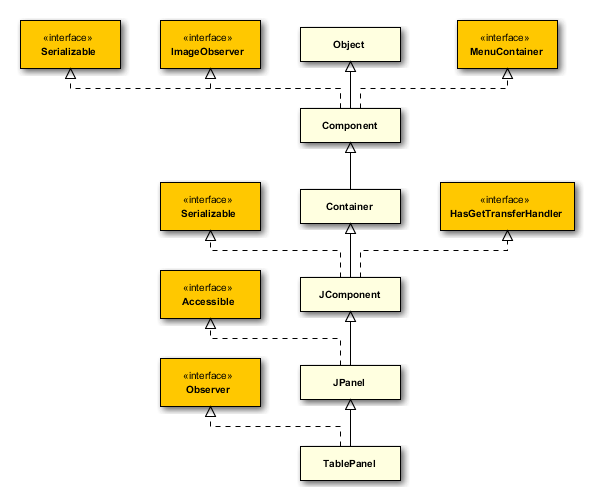
**public** TablePanel(MainFrame fr);

  // Public Methods

**public** **final** JTable getTable();  
  **public** **final** void setModel(DiagramModel m);  
  **public** **final** void update(Observable o, Object arg);  
}

*Author*

Куцовол Віктор



**TablePanel(MainFrame)**

**public** TablePanel(MainFrame fr);

Конструктор.

|  |  |
| --- | --- |
| Parameters | |
| fr | посилання на фрейм |

**getTable()**

**public** **final** JTable getTable();

Повертає таблицю.

|  |  |
| --- | --- |
| Parameters | |
| *return* | таблицю |

**setModel(DiagramModel)**

**public** **final** void setModel(DiagramModel m);

Встановлює модель діаграми.

|  |  |
| --- | --- |
| Parameters | |
| m | модель діаграми |

**update(Observable, Object)**

**public** **final** void update(Observable o, Object arg);

**Specified by:** Method update in interface Observer

Виконується, коли змінюється об'єкт, за яким ведеться нагляд.

|  |  |
| --- | --- |
| Parameters | |
| o | об'єкт, за яким ведеться нагляд |
| arg | значення, яке пересилається від об'єкта, за яким ведеться нагляд |

**3. Package com.main**

### **3.1.** Class AppSettings

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03.html#d0e7799)

[get(Object)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03.html#method-com.main.AppSettings.get-java.lang.Object)

[getInstance()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03.html#method-com.main.AppSettings.getInstance)

[put(Object, Object)](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03.html#method-com.main.AppSettings.put-java.lang.Object-java.lang.Object)

[save()](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03.html#method-com.main.AppSettings.save)

Класс настроек. Налаштування зберігаються у файлі "settings.txt".

**Synopsis**

**public** **final** **class** AppSettings {

  // Public Static Methods

**public** **static** AppSettings getInstance();

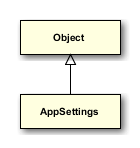
  // Public Methods

**public** Object get(Object key);  
  **public** void put(Object key, Object value);  
  **public** void save()

**throws**IOException;  
}

*Author*

Куцовол Віктор



**get(Object)**

**public** Object get(Object key);

Повертає значення за ключем.

|  |  |
| --- | --- |
| Parameters | |
| key | ключ |
| *return* | значення за ключем |

**getInstance()**

**public** **static** AppSettings getInstance();

Повертає екземпляр AppSettings.

|  |  |
| --- | --- |
| Parameters | |
| *return* | екземпляр AppSettings |

**put(Object, Object)**

**public** void put(Object key, Object value);

Додає до хештейбла значення та зв'язує з його ключем.

|  |  |
| --- | --- |
| Parameters | |
| key | ключ |
| value | значення |

**save()**

**public** void save()

**throws**IOException;

Зберігає налаштування у файлі "settings.txt".

**Exceptions**

IOException

виникає внаслідок помилки вводу/виводу

### **3.2.** Class Main

[Synopsis](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03s02.html#d0e8074)

[main(String[])](file:///D:\KPI\EclDocs\DiagramViewer\ar01s03s02.html#method-com.main.Main.main-java.lang.String_A)

Головний клас.

**Synopsis**

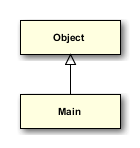
**public** **final** **class** Main {

  // Public Static Methods

**public** **static** void main(String[] args);  
}

*Author*

Куцовол Віктор



**main(String[])**

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

Виконується при запуску програми.

|  |  |
| --- | --- |
| Parameters | |
| args | параметри з командного рядка |

1. **Вихідний код всіх класів.**

**class CSVProcessor**

package com.csv;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileInputStream;

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.util.ArrayList;

import java.util.Iterator;

import java.util.StringTokenizer;

/\*\*

\* Оброблювач CSV файлів.

\*

\* @author Куцовол Віктор

\*

\*/

public class CSVProcessor {

/\*\*

\* Агрегований об'єкт класу ArrayList, в якому знаходяться рядки CSV-файлу.

\*

\* @serial

\*/

private ArrayList<String> csvStrings = new ArrayList<String>();

/\*\*

\* Об'єкт відокремлювач.

\*/

private String delimiter;

/\*\*

\* Повертає відокремлювач.

\*

\* @return відокремлювач

\*/

public final String getDelimiter() {

return delimiter;

}

/\*\*

\* Встановлює відокремлювач.

\*

\* @param del

\* новий відокремлювач

\*/

public final void setDelimiter(final String del) {

delimiter = del;

}

/\*\*

\* Конструктор.

\*

\* @param del

\* відокремлювач

\*/

public CSVProcessor(final String del) {

super();

delimiter = del;

}

/\*\*

\* Завантажує в файл рядки з агрегованого об'єкту.

\*

\* @param fileName

\* ім'я файлу

\* @throws IOException

\* виникає внаслідок помилки вводу/виводу

\*/

public final void save(final String fileName) throws IOException {

BufferedWriter wf = new BufferedWriter(new FileWriter(fileName));

Iterator<String> iterator = csvStrings.iterator();

while (iterator.hasNext()) {

wf.write(iterator.next());

wf.newLine();

}

wf.close();

}

/\*\*

\* Вивантажує з файлу рядки в агрегований об'єкт.

\*

\* @param fileName

\* ім'я файлу

\* @throws IOException

\* виникає внаслідок помилки вводу/виводу

\*/

public final void load(final String fileName) throws IOException {

BufferedReader rf = new BufferedReader(new FileReader(fileName));

csvStrings.clear();

String line;

while ((line = rf.readLine()) != null) {

csvStrings.add(line);

}

rf.close();

}

/\*\*

\* Серіалізує дані в файл.

\*

\* @throws Exception

\* виникає внаслідок помилки серіалізації

\* @param fileName

\* ім'я файлу для серіалізації

\*/

public final void serialize(final String fileName) throws Exception {

FileOutputStream fos = new FileOutputStream(fileName);

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(csvStrings);

oos.flush();

oos.close();

}

/\*\*

\* Серіалізує дані в файл "serialized.dat".

\*

\* @throws Exception

\* виникає внаслідок помилки серіалізації

\*/

public final void serialize() throws Exception {

serialize("serialized.dat");

}

/\*\*

\* Десеріалізує дані з файлу.

\*

\* @throws Exception

\* виникає внаслідок помилки десеріалізації

\* @param fileName

\* ім'я файлу для десеріалізації

\*/

@SuppressWarnings("unchecked")

public final void deserialize(final String fileName) throws Exception {

FileInputStream fis = new FileInputStream(fileName);

ObjectInputStream ois = new ObjectInputStream(fis);

csvStrings = (ArrayList<String>) ois.readObject();

ois.close();

}

/\*\*

\* Десеріалізує дані з файлу "serialized.dat".

\*

\* @throws Exception

\* виникає внаслідок помилки десеріалізації

\*/

public final void deserialize() throws Exception {

deserialize("serialized.dat");

}

/\*\*

\* Повертає агрегований об'єкт.

\*

\* @return агрегований об'єкт класу ArrayList

\*/

public final ArrayList<String> getCsvStrings() {

return csvStrings;

}

/\*\*

\* Виконує розбір даних агрегованого об'єкту.

\*

\* @return двовимірний масив

\*/

public final synchronized Object[][] parse() {

if (csvStrings.size() != 0) {

if (csvStrings.size() == 1) {

throw new CSVParseException(

"Невірний формат даних. Відсутні значення.");

}

Object[][] table = new Object[2][];

StringTokenizer t = new StringTokenizer(csvStrings.get(0),

delimiter);

table[0] = new Object[t.countTokens()];

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

table[0][i] = t.nextElement();

}

if (!csvStrings.get(1).contains(delimiter)

&& table[0].length != 1) {

throw new CSVParseException("Даний файл не є CSV.");

}

t = new StringTokenizer(csvStrings.get(1), delimiter);

table[1] = new Object[t.countTokens()];

if (table[1].length > table[0].length) {

throw new CSVParseException(

"Невірний формат даних. Зайві значення.");

}

if (table[1].length < table[0].length) {

throw new CSVParseException(

"Невірний формат даних. Не вистачає значень.");

}

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

table[1][i] = t.nextToken();

try {

Double.valueOf((String) table[1][i]);

} catch (NumberFormatException e) {

throw new CSVParseException(

"Невірний формат даних. Значення не є числа.");

}

}

return table;

} else {

return new Object[2][0];

}

}

/\*\*

\* Зберігає в агрегований об'єкт двовимірний масив у форматі CSV, а потім

\* завантажує в файл рядки з цього об'єкту.

\*

\* @param fileName

\* імя файлу

\* @param table

\* двовимірний масив

\* @throws IOException

\* виникає внаслідок помилки вводу/виводу

\*/

public final void save(final String fileName, final Object[][] table)

throws IOException {

csvStrings.clear();

String str;

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

str = "";

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

if (((String) table[i][j]).length() != 0) {

str += (String) table[i][j];

} else {

str += " ";

}

if (j != (table[i].length - 1)) {

str += delimiter;

}

}

csvStrings.add(str);

}

save(fileName);

}

}

**CSVParseException**

package com.csv;

/\*\*

\* Клас виключення розбору CSV файлів.

\*

\* @author Куцовол Віктор

\* \*/

public class CSVParseException extends IllegalArgumentException {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 6177850649807286847L;

/\*\*

\* Конструктор, що створює виключення CSVParseException з порожнім

\* деталізуючим повідомленням.

\*/

public CSVParseException() {

super();

}

/\*\*

\* Конструктор, що створює виключення CSVParseException з власним

\* деталізуючим повідомленням.

\*

\* @param message

\* деталізуюче повідомлення

\*/

public CSVParseException(final String message) {

super(message);

}

}

**CSVParseThread**

package com.csv;

/\*\*

\* Клас потоку розбору CSV файла.

\*

\* @author Куцовол Віктор

\*/

public class CSVParseThread implements Runnable {

/\*\*

\* Об'єкт CSV процесора.

\*/

private CSVProcessor processor;

/\*\*

\* Об'єкт потоку.

\*/

private Thread t;

/\*\*

\* Обєкт результуючої таблиці.

\*/

private Object[][] table = new Object[2][0];

/\*\*

\* Повертає потік.

\*

\* @return потік

\*/

public final Thread getThread() {

return t;

}

/\*\*

\* Конструктор.

\*

\* @param pro

\* CSV процесор

\*/

public CSVParseThread(final CSVProcessor pro) {

processor = pro;

t = new Thread(this);

t.start();

}

/\*\*

\* Виконується при запуску потоку.

\*/

public final void run() {

try {

table = processor.parse();

} catch (CSVParseException e) {

System.*out*.println(e.getMessage());

}

}

/\*\*

\* Повертає результуючу таблицю.

\*

\* @return результуючу таблицю

\*/

public final Object[][] getTable() {

return table;

}

}

**AddNewRowAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

/\*\*

\* Клас, що виконує додавання нового рядка.

\*

\* @author Куцовол Віктор

\*/

public class AddNewRowAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -4674346199290499316L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Ознака додавання після позиції рядка.

\*/

private boolean afterFlag;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\* @param after

\* ознака додавання після позиції рядка

\*/

public AddNewRowAction(final MainFrame fr, final String name,

final boolean after) {

frame = fr;

putValue(Action.NAME, name);

afterFlag = after;

setEnabled(false);

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

DiagramModel model = frame.getCurrentModel();

int index = frame.getTablePanel().getTable().getSelectedRow();

if (index == -1) {

index = model.getDiagramDrawer().getCurrentSector();

}

if (model != null && index != -1) {

frame.getCurrentModel().addNewRow(index, afterFlag);

model.doChange("add");

if (model.getTable()[0].length == 1) {

frame.getRemoveRowAction().setEnabled(false);

} else {

frame.getRemoveRowAction().setEnabled(true);

}

}

}

}

**ButtonTabComponent**

package com.gui;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Component;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import javax.swing.AbstractButton;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.plaf.basic.BasicButtonUI;

/\*\*

\*

\* Клас, що використовується як tabComponent.

\* Створює JLabel, що використовується для закриття вкладки

\*

\* @author Куцовол Віктор

\*

\*/

public class ButtonTabComponent extends JPanel {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 7317026987770482813L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\*/

@SuppressWarnings("serial")

public ButtonTabComponent(final MainFrame fr) {

super(new FlowLayout(FlowLayout.LEFT, 0, 0));

frame = fr;

final int shift = 5;

if (frame.getJtp() != null) {

setOpaque(false);

JLabel label = new JLabel() {

public String getText() {

int i = frame.getJtp().indexOfTabComponent(

ButtonTabComponent.this);

if (i != -1) {

return frame.getJtp().getTitleAt(i);

}

return null;

}

};

add(label);

label.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, shift));

JButton button = new TabButton();

add(button);

setBorder(BorderFactory.createEmptyBorder(2, 0, 0, 0));

}

}

/\*\*

\* Клас кнопки.

\*/

private class TabButton extends JButton implements ActionListener {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -1532516189094404360L;

/\*\*

\* Конструктор.

\*/

public TabButton() {

final int size = 17;

setPreferredSize(new Dimension(size, size));

setToolTipText("close this tab");

setUI(new BasicButtonUI());

setContentAreaFilled(false);

setFocusable(false);

setBorder(BorderFactory.createEtchedBorder());

setBorderPainted(false);

addMouseListener(buttonMouseListener);

setRolloverEnabled(true);

addActionListener(this);

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public void actionPerformed(final ActionEvent e) {

int i = frame.getJtp().indexOfTabComponent(ButtonTabComponent.this);

if (i != -1) {

frame.getJtp().setSelectedIndex(i);

frame.getCloseAction().actionPerformed(null);

}

}

/\*\*

\* Оновлює UI.

\*/

public void updateUI() {

}

/\*\*

\* Виконується при відображенні.

\*

\* @param g

\* графічний контекст

\*/

protected void paintComponent(final Graphics g) {

super.paintComponent(g);

Graphics2D g2 = (Graphics2D) g.create();

if (getModel().isPressed()) {

g2.translate(1, 1);

}

g2.setStroke(new BasicStroke(2));

g2.setColor(Color.BLACK);

if (getModel().isRollover()) {

g2.setColor(Color.MAGENTA);

}

final int delta = 6;

g2.drawLine(delta, delta, getWidth() - delta - 1, getHeight()

- delta - 1);

g2.drawLine(getWidth() - delta - 1, delta, delta, getHeight()

- delta - 1);

g2.dispose();

}

}

/\*\*

\* Клас оброблювача подій миші.

\*/

private static final MouseListener buttonMouseListener =

new MouseAdapter() {

public void mouseEntered(final MouseEvent e) {

Component component = e.getComponent();

if (component instanceof AbstractButton) {

AbstractButton button = (AbstractButton) component;

button.setBorderPainted(true);

}

}

public void mouseExited(final MouseEvent e) {

Component component = e.getComponent();

if (component instanceof AbstractButton) {

AbstractButton button = (AbstractButton) component;

button.setBorderPainted(false);

}

}

};

}

**ChangeDelimiterAction**

package com.gui;

import java.awt.event.ActionEvent;

import java.io.IOException;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.JOptionPane;

import com.main.AppSettings;

/\*\*

\* Клас, що виконує зміну відокремлювача CSV файлу.

\*

\* @author Куцовол Віктор

\*/

public class ChangeDelimiterAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -7942099344060457173L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\*/

public ChangeDelimiterAction(final MainFrame fr, final String name) {

frame = fr;

putValue(Action.NAME, name);

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

String newDel = (String) JOptionPane.showInputDialog(frame,

"Введіть відокремлювач: ", AppSettings.getInstance().get(

"delimiter"));

if (newDel != null) {

AppSettings.getInstance().put("delimiter", newDel);

try {

AppSettings.getInstance().save();

} catch (IOException e1) {

System.out.println("Помилка встановлення відокремлювача");

JOptionPane.showOptionDialog(frame,

"Помилка встановлення відокремлювача", "Помилка",

JOptionPane.CLOSED\_OPTION, JOptionPane.ERROR\_MESSAGE,

null, null, null);

}

}

}

}

**ChangeLookAndFeelAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.ImageIcon;

import javax.swing.JOptionPane;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.UIManager.LookAndFeelInfo;

import com.main.AppSettings;

import com.main.Main;

/\*\*

\* Клас, що виконує зміну стилю вікна.

\*

\* @author Куцовол Віктор

\*/

public class ChangeLookAndFeelAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -4073474641566445543L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\*/

public ChangeLookAndFeelAction(final MainFrame fr, final String name) {

frame = fr;

putValue(Action.NAME, name);

putValue(Action.SHORT\_DESCRIPTION, name);

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/configure.png")));

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

LookAndFeelInfo[] infos = UIManager.getInstalledLookAndFeels();

String[] styleNames = new String[infos.length];

String currentStyleName = "";

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

styleNames[i] = infos[i].getName();

if (AppSettings.getInstance().get("LookAndFeel").equals(

infos[i].getClassName())) {

currentStyleName = infos[i].getName();

}

}

String styleName = (String) JOptionPane.showInputDialog(frame,

"Виберіть стиль: ", "Налаштування стилю",

JOptionPane.QUESTION\_MESSAGE, null, styleNames,

currentStyleName);

if (styleName != null) {

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

if (styleName.equals(infos[i].getName())) {

try {

UIManager.setLookAndFeel(infos[i].getClassName());

SwingUtilities.updateComponentTreeUI(frame);

AppSettings.getInstance().put("LookAndFeel",

infos[i].getClassName());

AppSettings.getInstance().save();

} catch (Exception ex) {

System.out.println("Помилка встановлення стилю");

JOptionPane.showOptionDialog(frame,

"Помилка встановлення стилю", "Помилка",

JOptionPane.CLOSED\_OPTION,

JOptionPane.ERROR\_MESSAGE, null, null, null);

}

break;

}

}

}

}

}

**CloseAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.ImageIcon;

import com.main.Main;

/\*\*

\* Клас, що виконує закриття файлу.

\*

\* @author Куцовол Віктор

\*/

public class CloseAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 1053866910181848595L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Ознака закриття усіх вкладок.

\*/

private boolean all;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\* @param closeAll

\* ознака закриття усіх вкладок

\*/

public CloseAction(final MainFrame fr, final String name,

final boolean closeAll) {

frame = fr;

putValue(Action.NAME, name);

putValue(Action.SHORT\_DESCRIPTION, name);

if (closeAll) {

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/cancel.png")));

}

all = closeAll;

setEnabled(false);

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

if (all) {

frame.getModels().clear();

frame.getJtp().removeAll();

} else {

int index = frame.getJtp().getSelectedIndex();

frame.getModels().remove(index);

frame.getJtp().remove(index);

}

if (frame.getJtp().getTabCount() == 0) {

frame.getOpenInNewTabAction().setEnabled(false);

frame.getCloseAction().setEnabled(false);

frame.getCloseAllAction().setEnabled(false);

frame.getAddNewRowAfterAction().setEnabled(false);

frame.getAddNewRowBeforeAction().setEnabled(false);

frame.getRemoveRowAction().setEnabled(false);

frame.getSaveAction().setEnabled(false);

frame.getSaveAsAction().setEnabled(false);

frame.getSaveDiagramAsImageAction().setEnabled(false);

}

}

}

**CSVFileFilter**

package com.gui;

import java.io.File;

import javax.swing.filechooser.FileFilter;

/\*\*

\* Фільтер CSV файлів.

\*

\* @author Куцовол Віктор

\*/

public class CSVFileFilter extends FileFilter {

/\*\*

\* Повертає значення true, якщо в якості параметра йому було передано CSV

\* файл.

\*

\* @param f

\* файл

\* @return true, якщо в якості параметра йому було передано CSV файл

\*/

public final boolean accept(final File f) {

if (f.getName().endsWith(".csv")) {

return true;

}

if (f.isDirectory()) {

return true;

}

return false;

}

/\*\*

\* Повертає опис фільтра.

\*

\* @return рядок опису

\*/

public final String getDescription() {

return "CSV Files";

}

}

**DiagramDrawer**

package com.gui;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.MouseEvent;

import java.util.ArrayList;

/\*\*

\* Інтерфейс відображувача діаграм.

\*

\* @author Куцовол Віктор

\*/

public interface DiagramDrawer {

/\*\*

\* Повертає таблицю значень.

\*

\* @return таблиця значень.

\*/

Object[][] getTable();

/\*\*

\* Встановлює таблицю значень.

\*

\* @param t

\* таблиця значень

\*/

void setTable(final Object[][] t);

/\*\*

\* Встановлює таблицю значень і список кольорів секторів.

\*

\* @param t

\* таблиця значень

\* @param newColors

\* список кольорів секторів

\*/

void setTable(final Object[][] t, final ArrayList<Color> newColors);

/\*\*

\* Встановлює колір сектору за його індексом.

\*

\* @param index

\* індекс сектору

\* @param color

\* колір сектору

\*/

void setColor(final int index, final Color color);

/\*\*

\* Повертає колір сектору за його індексом.

\*

\* @param index

\* індекс сектору

\* @return колір сектору

\*/

Color getColor(final int index);

/\*\*

\* Додає колір сектору.

\*

\* @param index

\* індекс сектру

\*/

void addColor(final int index);

/\*\*

\* Видаляє колір сектору.

\*

\* @param index

\* індекс сектру

\*/

void removeColor(final int index);

/\*\*

\* Малює діаграму використовуючи даний графічний контекст.

\*

\* @param g

\* графічний контекст

\*/

void draw(final Graphics g);

/\*\*

\* Виконується при русі миші.

\*

\* @param event

\* подія миші

\*/

void mouseMoved(MouseEvent event);

/\*\*

\* Виконується при одночасному натисканні та русі миші.

\*

\* @param event

\* подія миші

\*/

void mouseDragged(MouseEvent event);

/\*\*

\* Повертає поточний сектор.

\*

\* @return the currentSector поточний сектор

\*/

int getCurrentSector();

}

**DiagramModel**

package com.gui;

import java.io.File;

import java.io.IOException;

import java.util.Arrays;

import java.util.Observable;

import javax.swing.JOptionPane;

import com.csv.CSVParseThread;

import com.csv.CSVProcessor;

import com.main.AppSettings;

/\*\*

\* Клас моделі діаграми.

\*

\* @author Куцовол Віктор

\*/

public class DiagramModel extends Observable {

/\*\*

\* Об'єкт CSV процесора.

\*/

private CSVProcessor processor;

/\*\*

\* Таблиця значень.

\*/

private Object[][] table;

/\*\*

\* Поле стратегії відображувача діаграм.

\*/

private DiagramDrawer diagramDrawer;

/\*\*

\* Повертає таблицю значень.

\*

\* @return таблицю знчень

\*/

public final Object[][] getTable() {

return table;

}

/\*\*

\* Файл.

\*/

private File file;

/\*\*

\* Повертає стратегію відображувача діаграм.

\*

\* @return стратегію відображувача діаграм

\*/

public final DiagramDrawer getDiagramDrawer() {

return diagramDrawer;

}

/\*\*

\* Встановлює стратегію відображувача діаграм.

\*

\* @param strategy

\* стратегія відображувача діаграм

\*/

public final void setDiagramDrawer(final DiagramDrawer strategy) {

diagramDrawer = strategy;

strategy.setTable(table);

}

/\*\*

\* Повертає файл.

\*

\* @return файл

\*/

public final File getFile() {

return file;

}

/\*\*

\* Встановлює файл.

\*

\* @param f

\* файл

\*/

public final void setFile(final File f) {

file = f;

}

/\*\*

\* Конструктор.

\*

\* @param f

\* CSV файл

\*/

public DiagramModel(final File f) {

file = new File("");

String del = (String) AppSettings.getInstance().get("delimiter");

processor = new CSVProcessor(del);

try {

processor.load(f.getAbsolutePath());

file = f;

} catch (IOException ex) {

System.out.println("Помилка завантаження файлу.");

JOptionPane.showOptionDialog(null, "Помилка завантаження файлу",

"Помилка", JOptionPane.CLOSED\_OPTION,

JOptionPane.ERROR\_MESSAGE, null, null, null);

}

CSVParseThread pth = new CSVParseThread(processor);

try {

pth.getThread().join();

} catch (InterruptedException e) {

System.out.println("Переривання потоку.");

JOptionPane.showOptionDialog(null, "Переривання потоку", "Помилка",

JOptionPane.CLOSED\_OPTION, JOptionPane.ERROR\_MESSAGE, null,

null, null);

}

table = pth.getTable();

if (table[0].length == 0) {

JOptionPane.showOptionDialog(null, "Помилковий формат файлу",

"Помилка", JOptionPane.CLOSED\_OPTION,

JOptionPane.ERROR\_MESSAGE, null, null, null);

}

diagramDrawer = new SectorDiagramDrawer(table);

}

/\*\*

\* Конструктор.

\*

\* @param newFile

\* ім'я нового файлу

\*/

public DiagramModel(final String newFile) {

file = new File(newFile);

String del = (String) AppSettings.getInstance().get("delimiter");

processor = new CSVProcessor(del);

table = new Object[2][1];

table[0][0] = "";

table[1][0] = "0";

diagramDrawer = new SectorDiagramDrawer(table);

}

/\*\*

\* Повідомляє про зміни.

\*

\* @param par

\* характер змін

\*/

public final void doChange(final String par) {

setChanged();

notifyObservers(par);

}

/\*\*

\* Додає новий рядок.

\*

\* @param index

\* позиція рядка

\* @param after

\* ознака додавання після позиції рядка

\*/

public final void addNewRow(final int index, final boolean after) {

table[0] = Arrays.copyOf(table[0], table[0].length + 1);

table[1] = Arrays.copyOf(table[1], table[1].length + 1);

System.arraycopy(table[0], index, table[0], index + 1, table[0].length

- index - 1);

System.arraycopy(table[1], index, table[1], index + 1, table[1].length

- index - 1);

if (after) {

table[0][index + 1] = "";

table[1][index + 1] = "0";

diagramDrawer.addColor(index + 1);

} else {

table[0][index] = "";

table[1][index] = "0";

diagramDrawer.addColor(index);

}

}

/\*\*

\* Видаляє рядок.

\*

\* @param index

\* позиція рядка

\*/

public final void removeRow(final int index) {

System.arraycopy(table[0], index + 1, table[0], index, table[0].length

- index - 1);

System.arraycopy(table[1], index + 1, table[1], index, table[1].length

- index - 1);

table[0] = Arrays.copyOf(table[0], table[0].length - 1);

table[1] = Arrays.copyOf(table[1], table[1].length - 1);

diagramDrawer.removeColor(index);

}

/\*\*

\* Зберігає файл.

\*

\* @param f

\* файл

\*/

public final void save(final File f) {

file = f;

try {

processor.save(f.getAbsolutePath(), table);

} catch (IOException e) {

System.out.println("Помилка збереження");

JOptionPane.showOptionDialog(null, "Помилка збереження", "Помилка",

JOptionPane.CLOSED\_OPTION, JOptionPane.ERROR\_MESSAGE, null,

null, null);

}

}

}

**DiagramPanel**

package com.gui;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseMotionListener;

import java.util.Observable;

import java.util.Observer;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.JColorChooser;

import javax.swing.JMenuItem;

import javax.swing.JPanel;

import javax.swing.JPopupMenu;

/\*\*

\* Клас панелі діаграми.

\*

\* @author Куцовол Віктор

\*/

public class DiagramPanel extends JPanel implements Observer {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 4025340017415526188L;

/\*\*

\* Модель діаграми.

\*/

private DiagramModel model;

/\*\*

\* Контекстне меню.

\*/

private JPopupMenu jpu;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\*/

public DiagramPanel(final MainFrame fr) {

setBackground(Color.WHITE);

setModel(fr.getCurrentModel());

addMouseMotionListener(new MouseMotionHandler());

jpu = new JPopupMenu();

jpu.add(new JMenuItem(new ChangeColor()));

jpu.add(new JMenuItem(fr.getAddNewRowBeforeAction()));

jpu.add(new JMenuItem(fr.getAddNewRowAfterAction()));

jpu.add(new JMenuItem(fr.getRemoveRowAction()));

addMouseListener(new MouseAdapter() {

public void mousePressed(final MouseEvent e) {

if (model != null && model.getDiagramDrawer() != null

&& model.getDiagramDrawer().getCurrentSector() != -1) {

if (e.isPopupTrigger()) {

jpu.show(e.getComponent(), e.getX(), e.getY());

}

}

repaint();

}

public void mouseReleased(final MouseEvent e) {

if (model != null && model.getDiagramDrawer() != null

&& model.getDiagramDrawer().getCurrentSector() != -1) {

if (e.isPopupTrigger()) {

jpu.show(e.getComponent(), e.getX(), e.getY());

}

}

repaint();

}

});

}

/\*\*

\* Встановлює модель діаграми.

\*

\* @param m

\* модель діаграми

\*/

public final void setModel(final DiagramModel m) {

model = m;

repaint();

}

/\*\*

\* Виконується, коли змінюється об'єкт, за яким ведеться нагляд.

\*

\* @param o

\* об'єкт, за яким ведеться нагляд

\* @param arg

\* значення, яке пересилається від об'єкта, за яким ведеться

\* нагляд

\*/

public final void update(final Observable o, final Object arg) {

repaint();

}

/\*\*

\* Виконується при відображенні.

\*

\* @param g

\* графічний контекст

\*/

protected final void paintComponent(final Graphics g) {

g.setClip(g.getClipBounds().x, g.getClipBounds().y, getWidth()

- getInsets().left - getInsets().right, getHeight()

- getInsets().top - getInsets().bottom);

super.paintComponent(g);

if (model != null && model.getDiagramDrawer() != null) {

model.getDiagramDrawer().draw(g);

}

}

/\*\*

\* Клас оброблювача подій миші.

\*

\* @author Куцовол Віктор

\*/

private class MouseMotionHandler implements MouseMotionListener {

/\*\*

\* Виконується при одночасному натисканні та русі миші.

\*

\* @param e

\* подія миші

\*/

public void mouseDragged(final MouseEvent e) {

if (model != null && model.getDiagramDrawer() != null) {

model.getDiagramDrawer().mouseDragged(e);

model.doChange("");

}

}

/\*\*

\* Виконується при русі миші.

\*

\* @param e

\* подія миші

\*/

public void mouseMoved(final MouseEvent e) {

if (model != null && model.getDiagramDrawer() != null) {

model.getDiagramDrawer().mouseMoved(e);

repaint();

}

}

}

/\*\*

\* Клас, що виконує закриття файлу.

\*

\* @author Куцовол Віктор

\*/

@SuppressWarnings("serial")

private class ChangeColor extends AbstractAction {

/\*\*

\* Конструктор.

\*/

public ChangeColor() {

putValue(Action.NAME, "Змінити колір...");

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

int index = model.getDiagramDrawer().getCurrentSector();

if (index != -1) {

Color color = model.getDiagramDrawer().getColor(index);

color = JColorChooser.showDialog(null,

"Вибір кольору сектора: \""

+ model.getTable()[0][index] + "\"", color);

if (color != null) {

model.getDiagramDrawer().setColor(index, color);

}

}

repaint();

}

}

}

**DiagramViewerMenu**

package com.gui;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

/\*\*

\* Клас меню.

\* @author Куцовол Віктор

\*/

public class DiagramViewerMenu extends JMenuBar {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 758708546654543876L;

/\*\*

\* Конструктор.

\*

\* @param frame

\* посилання на головне вікно програми

\*/

public DiagramViewerMenu(final MainFrame frame) {

JMenu fileMenu = new JMenu("Файл");

JMenuItem newFileItem = new JMenuItem(frame.getNewFileAction());

JMenuItem openInCurrentTabItem = new JMenuItem(frame

.getOpenInCurrentTabAction());

JMenuItem openInNewTabItem = new JMenuItem(frame

.getOpenInNewTabAction());

JMenuItem closeItem = new JMenuItem(frame.getCloseAction());

JMenuItem closeAllItem = new JMenuItem(frame.getCloseAllAction());

JMenuItem saveItem = new JMenuItem(frame.getSaveAction());

JMenuItem saveAsItem = new JMenuItem(frame.getSaveAsAction());

JMenuItem exitItem = new JMenuItem(frame.getExitAction());

fileMenu.add(newFileItem);

fileMenu.add(openInCurrentTabItem);

fileMenu.add(openInNewTabItem);

fileMenu.addSeparator();

fileMenu.add(closeItem);

fileMenu.add(closeAllItem);

fileMenu.addSeparator();

fileMenu.add(saveItem);

fileMenu.add(saveAsItem);

fileMenu.addSeparator();

fileMenu.add(exitItem);

add(fileMenu);

JMenu tableMenu = new JMenu("Таблиця");

JMenuItem addNewRowBeforeItem = new JMenuItem(frame

.getAddNewRowBeforeAction());

JMenuItem addNewRowAfterItem = new JMenuItem(frame

.getAddNewRowAfterAction());

JMenuItem removeRowItem = new JMenuItem(frame

.getRemoveRowAction());

tableMenu.add(addNewRowBeforeItem);

tableMenu.add(addNewRowAfterItem);

tableMenu.add(removeRowItem);

add(tableMenu);

JMenu diagramMenu = new JMenu("Діаграма");

JMenuItem saveDiagramAsImageItem = new JMenuItem(frame

.getSaveDiagramAsImageAction());

diagramMenu.add(saveDiagramAsImageItem);

add(diagramMenu);

JMenu settingsMenu = new JMenu("Налаштування");

JMenuItem changeLookAndFeelItem = new JMenuItem(frame

.getChangeLookAndFeelAction());

settingsMenu.add(changeLookAndFeelItem);

JMenuItem changeDelimiterItem = add(settingsMenu);

}

}

**DiagramViewerToolBar**

package com.gui;

import java.awt.Insets;

import javax.swing.JButton;

import javax.swing.JToolBar;

/\*\*

\* Клас панелі інструментів.

\*

\* @author Куцовол Віктор

\*/

public class DiagramViewerToolBar extends JToolBar {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 7019857712093372015L;

/\*\*

\* Конструктор.

\*

\* @param frame

\* посилання на головне вікно програми

\*/

public DiagramViewerToolBar(final MainFrame frame) {

super(JToolBar.HORIZONTAL);

JButton newFileButton = new JButton(frame.getNewFileAction());

newFileButton.setText(null);

JButton openInCurrentTabButton = new JButton(frame

.getOpenInCurrentTabAction());

openInCurrentTabButton.setText(null);

JButton openInNewTabButton = new JButton(frame.getOpenInNewTabAction());

openInNewTabButton.setText(null);

JButton closeAllButton = new JButton(frame.getCloseAllAction());

closeAllButton.setText(null);

JButton saveButton = new JButton(frame.getSaveAction());

saveButton.setText(null);

JButton saveAsButton = new JButton(frame.getSaveAsAction());

saveAsButton.setText(null);

JButton saveDiagramAsImageButton = new JButton(frame

.getSaveDiagramAsImageAction());

saveDiagramAsImageButton.setText(null);

JButton changeLookAndFeelButton = new JButton(frame

.getChangeLookAndFeelAction());

changeLookAndFeelButton.setText(null);

add(newFileButton);

add(openInCurrentTabButton);

add(openInNewTabButton);

addSeparator();

add(closeAllButton);

addSeparator();

add(saveButton);

add(saveAsButton);

add(saveDiagramAsImageButton);

addSeparator();

add(changeLookAndFeelButton);

setMargin(new Insets(0, 0, 0, 0));

}

}

**ExitAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

/\*\*

\* Клас, що виконує закриття програми.

\*

\* @author Куцовол Віктор

\*/

public class ExitAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -1027877877263188770L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\*/

public ExitAction(final MainFrame fr, final String name) {

frame = fr;

putValue(Action.NAME, name);

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

frame.setVisible(false);

System.exit(0);

}

}

**JPEGFileFilter**

package com.gui;

import java.io.File;

import javax.swing.filechooser.FileFilter;

/\*\*

\* Фільтер JPEG файлів.

\*

\* @author Куцовол Віктор

\*/

public class JPEGFileFilter extends FileFilter {

/\*\*

\* Повертає значення true, якщо в якості параметра йому було передано JPEG

\* файл.

\*

\* @param f

\* файл

\* @return true, якщо в якості параметра йому було передано JPEG файл

\*/

public final boolean accept(final File f) {

if (f.getName().endsWith(".jpg")) {

return true;

}

if (f.isDirectory()) {

return true;

}

return false;

}

/\*\*

\* Повертає опис фільтра.

\*

\* @return рядок опису

\*/

public final String getDescription() {

return "JPEG Files";

}

}

**MainFrame**

package com.gui;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.Image;

import java.awt.Toolkit;

import java.util.ArrayList;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JSplitPane;

import javax.swing.JTabbedPane;

import javax.swing.SwingConstants;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import com.main.AppSettings;

import com.main.Main;

/\*\*

\* Клас головного вікна.

\*

\* @author Куцовол Віктор

\*/

public class MainFrame extends JFrame {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -6097807245108559922L;

/\*\*

\* Масив моделей діаграм.

\*/

private ArrayList<DiagramModel> models = new ArrayList<DiagramModel>();

/\*\*

\* Поточна модель діаграми.

\*/

private DiagramModel currentModel;

/\*\*

\* Панель таблиці.

\*/

private TablePanel tablePanel;

/\*\*

\* Панель діаграми.

\*/

private DiagramPanel diagramPanel;

/\*\*

\* Панель вкладок.

\*/

private JTabbedPane jtp;

/\*\*

\* Дія, що відкриває файл у новій вкладці.

\*/

private OpenAction openInNewTabAction;

/\*\*

\* Дія, що відкриває файл у поточній вкладці.

\*/

private OpenAction openInCurrentTabAction;

/\*\*

\* Дія, що закриває файл.

\*/

private CloseAction closeAction;

/\*\*

\* Дія, що закриває усі файли.

\*/

private CloseAction closeAllAction;

/\*\*

\* Дія, що виконує закриття програми.

\*/

private ExitAction exitAction;

/\*\*

\* Дія, що виконує додавання нового рядка перед поточним.

\*/

private AddNewRowAction addNewRowBeforeAction;

/\*\*

\* Дія, що виконує додавання нового рядка після поточного.

\*/

private AddNewRowAction addNewRowAfterAction;

/\*\*

\* Дія, що виконує видалення рядка.

\*/

private RemoveRowAction removeRowAction;

/\*\*

\* Дія, що виконує зберігання у цьому ж файлі.

\*/

private SaveAction saveAction;

/\*\*

\* Дія, що виконує зберігання в іншому файлі.

\*/

private SaveAction saveAsAction;

/\*\*

\* Дія, що виконує створення нового файлу.

\*/

private NewFileAction newFileAction;

/\*\*

\* Дія, що виконує зберігання діагрми у файлі формату JPEG.

\*/

private SaveDiagramAsImageAction saveDiagramAsImageAction;

/\*\*

\* Дія, що виконує зміну стилю вікна.

\*/

private ChangeLookAndFeelAction changeLookAndFeelAction;

/\*\*

\* Дія, що виконує зміну відокремлювача CSV файлу.

\*/

private ChangeDelimiterAction changeDelimiterAction;

/\*\*

\* Конструктор.

\*/

public MainFrame() {

final double heightPercent = 0.9;

final double widthPercent = 0.9;

Toolkit kit = Toolkit.getDefaultToolkit();

Dimension screenSize = kit.getScreenSize();

int screenHeight = screenSize.height;

int screenWidth = screenSize.width;

setSize((int) (screenWidth \* widthPercent),

(int) (screenHeight \* heightPercent));

Image img = kit.getImage(Main.class.getResource("img/main.png"));

setIconImage(img);

try {

UIManager.setLookAndFeel((String) AppSettings.getInstance().get(

"LookAndFeel"));

SwingUtilities.updateComponentTreeUI(this);

} catch (Exception ex) {

System.out.println("Помилка встановлення стилю");

JOptionPane.showOptionDialog(this, "Помилка встановлення стилю",

"Помилка", JOptionPane.CLOSED\_OPTION,

JOptionPane.ERROR\_MESSAGE, null, null, null);

}

currentModel = null;

openInNewTabAction = new OpenAction(this,

"Відкрити файл у новій вкладці...", true);

openInCurrentTabAction = new OpenAction(this, "Відкрити файл...",

false);

closeAction = new CloseAction(this, "Закрити", false);

closeAllAction = new CloseAction(this, "Закрити все", true);

exitAction = new ExitAction(this, "Вихід");

saveAction = new SaveAction(this, "Зберегти", false);

saveAsAction = new SaveAction(this, "Зберегти як...", true);

addNewRowAfterAction = new AddNewRowAction(this,

"Додати новий після даного", true);

addNewRowBeforeAction = new AddNewRowAction(this,

"Додати новий перед даним", false);

removeRowAction = new RemoveRowAction(this, "Видалити");

newFileAction = new NewFileAction(this, "Новий");

saveDiagramAsImageAction = new SaveDiagramAsImageAction(this,

"Зберегти діаграму як зображення...");

changeLookAndFeelAction = new ChangeLookAndFeelAction(this,

"Змінити стиль...");

changeDelimiterAction = new ChangeDelimiterAction(this,

"Змінити відокремлювач CSV файлу...");

setLayout(new BorderLayout());

setJMenuBar(new DiagramViewerMenu(this));

add(new DiagramViewerToolBar(this), BorderLayout.NORTH);

diagramPanel = new DiagramPanel(this);

tablePanel = new TablePanel(this);

jtp = new JTabbedPane(SwingConstants.TOP, JTabbedPane.

SCROLL\_TAB\_LAYOUT);

jtp.addChangeListener(new ChangeListener() {

public void stateChanged(final ChangeEvent e) {

if (jtp.getSelectedIndex() == -1) {

currentModel = null;

} else {

currentModel = models.get(jtp.getSelectedIndex());

diagramPanel.setModel(currentModel);

tablePanel.setModel(currentModel);

JSplitPane jsp = new JSplitPane(

JSplitPane.HORIZONTAL\_SPLIT, true, new JScrollPane(

tablePanel), diagramPanel);

jtp.setComponentAt(jtp.getSelectedIndex(), jsp);

if (currentModel.getTable()[0].length == 1) {

removeRowAction.setEnabled(false);

} else {

removeRowAction.setEnabled(true);

}

}

}

});

add(jtp);

setExtendedState(JFrame.MAXIMIZED\_BOTH);

}

/\*\*

\* Повертає поточну модель.

\*

\* @return поточну модель діаграми

\*/

public final DiagramModel getCurrentModel() {

return currentModel;

}

/\*\*

\* Встановлює поточну модель діаграми.

\*

\* @param model

\* поточна модель діаграми

\*/

public final void setCurrentModel(final DiagramModel model) {

currentModel = model;

}

/\*\*

\* Повертає масив моделей діаграми.

\*

\* @return масив моделей діаграми

\*/

public final ArrayList<DiagramModel> getModels() {

return models;

}

/\*\*

\* Повертає панель таблиці.

\*

\* @return панель таблиці

\*/

public final TablePanel getTablePanel() {

return tablePanel;

}

/\*\*

\* Повертає панель діаграми.

\*

\* @return панель діаграми

\*/

public final DiagramPanel getDiagramPanel() {

return diagramPanel;

}

/\*\*

\* Повертає панель вкладок.

\*

\* @return панель вкладок

\*/

public final JTabbedPane getJtp() {

return jtp;

}

/\*\*

\* Повертає дію, що відкриває файл у новій вкладці.

\*

\* @return дію, що відкриває файл у новій вкладці

\*/

public final OpenAction getOpenInNewTabAction() {

return openInNewTabAction;

}

/\*\*

\* Повертає дію, що відкриває файл у поточній вкладці.

\*

\* @return дію, що відкриває файл у поточній вкладці

\*/

public final OpenAction getOpenInCurrentTabAction() {

return openInCurrentTabAction;

}

/\*\*

\* Повертає дію, що закриває файл.

\*

\* @return дію, що закриває файл

\*/

public final CloseAction getCloseAction() {

return closeAction;

}

/\*\*

\* Повертає дію, що закриває усі файли.

\*

\* @return дію, що закриває усі файли

\*/

public final CloseAction getCloseAllAction() {

return closeAllAction;

}

/\*\*

\* Повертає дію, що виконує закриття програми.

\*

\* @return дію, що виконує закриття програми

\*/

public final ExitAction getExitAction() {

return exitAction;

}

/\*\*

\* Повертає дію, що виконує додавання нового рядка перед поточним.

\*

\* @return дію, що виконує додавання нового рядка перед поточним

\*/

public final AddNewRowAction getAddNewRowBeforeAction() {

return addNewRowBeforeAction;

}

/\*\*

\* Повертає дію, що виконує додавання нового рядка після поточного.

\*

\* @return дію, що виконує додавання нового рядка після поточного.

\*/

public final AddNewRowAction getAddNewRowAfterAction() {

return addNewRowAfterAction;

}

/\*\*

\* Повертає дію, що виконує видалення рядка.

\*

\* @return дію, що виконує видалення рядка

\*/

public final RemoveRowAction getRemoveRowAction() {

return removeRowAction;

}

/\*\*

\* Повертає дію, що виконує зберігання у цьому ж файлі.

\*

\* @return дію, що виконує зберігання у цьому ж файлі

\*/

public final SaveAction getSaveAction() {

return saveAction;

}

/\*\*

\* Повертає дію, що виконує зберігання в іншому файлі.

\*

\* @return дію, що виконує зберігання в іншому файлі

\*/

public final SaveAction getSaveAsAction() {

return saveAsAction;

}

/\*\*

\* Повертає дію, що виконує створення нового файлу.

\*

\* @return дію, що виконує створення нового файлу

\*/

public final NewFileAction getNewFileAction() {

return newFileAction;

}

/\*\*

\* Повертає дію, що виконує зберігання діагрми у файлі формату JPEG.

\*

\* @return дію, що виконує зберігання діагрми у файлі формату JPEG

\*/

public final SaveDiagramAsImageAction getSaveDiagramAsImageAction() {

return saveDiagramAsImageAction;

}

/\*\*

\* Повертає дію, що виконує зміну стилю вікна.

\*

\* @return дію, що виконує зміну стилю вікна

\*/

public final ChangeLookAndFeelAction getChangeLookAndFeelAction() {

return changeLookAndFeelAction;

}

/\*\*

\* Повертає дію, що виконує зміну відокремлювача CSV файлу.

\*

\* @return дію, що виконує зміну відокремлювача CSV файлу

\*/

public final ChangeDelimiterAction getChangeDelimiterAction() {

return changeDelimiterAction;

}

}

**NewFileAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.ImageIcon;

import javax.swing.JScrollPane;

import javax.swing.JSplitPane;

import com.main.Main;

/\*\*

\* Клас, що виконує створення нового файлу.

\*

\* @author Куцовол Віктор

\*/

public class NewFileAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -1919116897881656952L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Номер нового файлу.

\*/

private int num;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\*/

public NewFileAction(final MainFrame fr, final String name) {

frame = fr;

putValue(Action.NAME, name);

setEnabled(true);

putValue(Action.SHORT\_DESCRIPTION, name);

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/filenew.png")));

num = 0;

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

DiagramModel model = new DiagramModel("noname" + num + ".csv");

model.addObserver(frame.getDiagramPanel());

model.addObserver(frame.getTablePanel());

JSplitPane jsp = new JSplitPane(JSplitPane.HORIZONTAL\_SPLIT, true,

new JScrollPane(frame.getTablePanel()),

frame.getDiagramPanel());

frame.getModels().add(model);

frame.getJtp().addTab("noname" + num + ".csv", jsp);

frame.getJtp().setSelectedIndex(frame.getJtp().getTabCount() - 1);

frame.getJtp().setTabComponentAt(frame.getJtp().getTabCount() - 1,

new ButtonTabComponent(frame));

num++;

frame.getOpenInNewTabAction().setEnabled(true);

frame.getCloseAction().setEnabled(true);

frame.getCloseAllAction().setEnabled(true);

frame.getAddNewRowAfterAction().setEnabled(true);

frame.getAddNewRowBeforeAction().setEnabled(true);

frame.getSaveAction().setEnabled(true);

frame.getSaveAsAction().setEnabled(true);

frame.getSaveDiagramAsImageAction().setEnabled(true);

frame.getRemoveRowAction().setEnabled(false);

}

}

**OpenAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import javax.swing.JScrollPane;

import javax.swing.JSplitPane;

import com.main.Main;

/\*\*

\* Клас, що виконує відкриття файлу.

\*

\* @author Куцовол Віктор

\*/

public class OpenAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 484153554180610766L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Ознака відкриття у новій вкладці.

\*/

private boolean newTab;

/\*\*

\* Діалог вибору файлу.

\*/

private JFileChooser jfc;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\* @param inNewTab

\* ознака відкриття у новій вкладці

\*/

public OpenAction(final MainFrame fr, final String name,

final boolean inNewTab) {

frame = fr;

putValue(Action.NAME, name);

putValue(Action.SHORT\_DESCRIPTION, name);

newTab = inNewTab;

if (inNewTab) {

setEnabled(false);

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/tab\_new.png")));

} else {

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/fileopen.png")));

}

jfc = new JFileChooser();

jfc.setFileFilter(new CSVFileFilter());

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

int result = jfc.showOpenDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

DiagramModel model = new DiagramModel(jfc.getSelectedFile());

model.addObserver(frame.getDiagramPanel());

model.addObserver(frame.getTablePanel());

JSplitPane jsp = new JSplitPane(JSplitPane.HORIZONTAL\_SPLIT, true,

new JScrollPane(frame.getTablePanel()), frame

.getDiagramPanel());

if (newTab || frame.getJtp().getTabCount() == 0) {

frame.getModels().add(model);

frame.getJtp().addTab(jfc.getSelectedFile().getName(), jsp);

frame.getJtp().setTabComponentAt(

frame.getJtp().getTabCount() - 1,

new ButtonTabComponent(frame));

frame.getJtp().setSelectedIndex(

frame.getJtp().getTabCount() - 1);

} else if (frame.getJtp().getTabCount() != 0) {

frame.getModels().set(frame.getJtp().getSelectedIndex(), model);

frame.setCurrentModel(frame.getModels().get(

frame.getJtp().getSelectedIndex()));

frame.getDiagramPanel().setModel(frame.getCurrentModel());

frame.getTablePanel().setModel(frame.getCurrentModel());

frame.getJtp().setComponentAt(

frame.getJtp().getSelectedIndex(), jsp);

frame.getJtp().setTitleAt(frame.getJtp().getSelectedIndex(),

jfc.getSelectedFile().getName());

if (frame.getCurrentModel().getTable()[0].length == 1) {

frame.getRemoveRowAction().setEnabled(false);

} else {

frame.getRemoveRowAction().setEnabled(true);

}

}

frame.getOpenInNewTabAction().setEnabled(true);

frame.getCloseAction().setEnabled(true);

frame.getCloseAllAction().setEnabled(true);

frame.getAddNewRowAfterAction().setEnabled(true);

frame.getAddNewRowBeforeAction().setEnabled(true);

frame.getSaveAction().setEnabled(true);

frame.getSaveAsAction().setEnabled(true);

frame.getSaveDiagramAsImageAction().setEnabled(true);

}

}

}

**RemoveRowAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

/\*\*

\* Клас, що виконує видалення рядка.

\*

\* @author Куцовол Віктор

\*

\*/

public class RemoveRowAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 7211041617730253836L;

/\*\*

\* /\*\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\*/

public RemoveRowAction(final MainFrame fr, final String name) {

frame = fr;

putValue(Action.NAME, name);

setEnabled(false);

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

DiagramModel model = frame.getCurrentModel();

int index = frame.getTablePanel().getTable().getSelectedRow();

if (index == -1) {

index = model.getDiagramDrawer().getCurrentSector();

}

if (model != null && index != -1) {

model.removeRow(index);

model.doChange("remove");

if (model.getTable()[0].length == 1) {

frame.getRemoveRowAction().setEnabled(false);

} else {

frame.getRemoveRowAction().setEnabled(true);

}

}

}

}

**SaveAction**

package com.gui;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import com.main.Main;

/\*\*

\* Клас, що виконує зберігання файлу.

\*

\* @author Куцовол Віктор

\*/

public class SaveAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = -7405214080191726687L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Флаг збереження файлу під іншим іменем.

\*/

private boolean saveAsFlag;

/\*\*

\* Діалог збереження файлу.

\*/

private JFileChooser jfc;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\* @param saveAs

\* флаг збереження файлу під іншим іменем.

\*/

public SaveAction(final MainFrame fr, final String name,

final boolean saveAs) {

frame = fr;

putValue(Action.NAME, name);

putValue(Action.SHORT\_DESCRIPTION, name);

if (saveAs) {

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/filesaveas.png")));

} else {

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/filesave.png")));

}

setEnabled(false);

saveAsFlag = saveAs;

jfc = new JFileChooser();

jfc.setFileFilter(new CSVFileFilter());

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

DiagramModel model = frame.getCurrentModel();

if (model != null) {

if (saveAsFlag) {

int result = jfc.showSaveDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

model.save(jfc.getSelectedFile());

frame.getJtp().setTitleAt(

frame.getJtp().getSelectedIndex(),

jfc.getSelectedFile().getName());

}

} else {

model.save(model.getFile());

}

}

}

}

**SaveDiagramAsImageAction**

package com.gui;

import java.awt.event.ActionEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import com.main.Main;

/\*\*

\* Клас, що виконує зберігання діагрми у файлі формату JPEG.

\*

\* @author Куцовол Віктор

\*/

public class SaveDiagramAsImageAction extends AbstractAction {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 6061904337345719827L;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Діалог збереження файлу.

\*/

private JFileChooser jfc;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\* @param name

\* назва дії

\*/

public SaveDiagramAsImageAction(final MainFrame fr, final String name) {

frame = fr;

putValue(Action.NAME, name);

putValue(Action.SHORT\_DESCRIPTION, name);

putValue(SMALL\_ICON, new ImageIcon(Main.class.

getResource("img/thumbnail.png")));

setEnabled(false);

jfc = new JFileChooser();

jfc.setFileFilter(new JPEGFileFilter());

}

/\*\*

\* Виконується при виникненні події.

\*

\* @param e

\* подія

\*/

public final void actionPerformed(final ActionEvent e) {

int result = jfc.showSaveDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

BufferedImage img = new BufferedImage(frame.getDiagramPanel()

.getWidth(), frame.getDiagramPanel().getHeight(),

BufferedImage.TYPE\_INT\_RGB);

frame.getDiagramPanel().paint(img.getGraphics());

File f = jfc.getSelectedFile();

try {

ImageIO.write(img, "jpg", f);

} catch (IOException e1) {

System.out.println("Помилка збереження діаграми");

JOptionPane.showOptionDialog(frame,

"Помилка збереження діаграми", "Помилка",

JOptionPane.CLOSED\_OPTION, JOptionPane.ERROR\_MESSAGE,

null, null, null);

}

}

}

}

**SectorDiagramDrawer**

package com.gui;

import java.awt.Color;

import java.awt.Component;

import java.awt.Cursor;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Point;

import java.awt.Polygon;

import java.awt.event.MouseEvent;

import java.util.ArrayList;

/\*\*

\* Клас відображувача кільцевої діаграми.

\*

\* @author Куцовол Віктор

\*/

public class SectorDiagramDrawer implements DiagramDrawer {

/\*\*

\* Список кольорів секторів.

\*/

private ArrayList<Color> colors = new ArrayList<Color>();

/\*\*

\* Радіус діаграми.

\*/

private double r;

/\*\*

\* Величина, на яку збільшується радіус сектора при його виділенні.

\*/

private double h;

/\*\*

\* Таблиця значень.

\*/

private Object[][] table;

/\*\*

\* Частини секторів.

\*/

private Polygon[][] polygons;

/\*\*

\* Поточний сектор.

\*/

private int currentSector;

/\*\*

\* Крок малювання.

\*/

private double step;

/\*\*

\* Координата x центра діаграми.

\*/

private double cX;

/\*\*

\* Координата y центра діаграми.

\*/

private double cY;

/\*\*

\* Коефіцієнт а еліпса.

\*/

private double a;

/\*\*

\* Коефіцієнт b еліпса.

\*/

private double b;

/\*\*

\* Старе значення координати Y.

\*/

private int oldY;

/\*\*

\* Повертає таблицю значень.

\*

\* @return таблиця значень.

\*/

public final Object[][] getTable() {

return table;

}

/\*\*

\* Встановлює таблицю значень.

\*

\* @param t

\* таблиця значень

\*/

public final void setTable(final Object[][] t) {

table = t;

colors.clear();

createColors();

}

/\*\*

\* Встановлює таблицю значень та список кольорів секторів.

\*

\* @param t

\* таблиця значень

\* @param newColors

\* список кольорів секторів

\*/

public final void setTable(final Object[][] t,

final ArrayList<Color> newColors) {

table = t;

colors = newColors;

}

/\*\*

\* Повертає крок малювання.

\*

\* @return крок малювання

\*/

public final double getStep() {

return step;

}

/\*\*

\* Встановлює крок малювання.

\*

\* @param s

\* крок малювання

\*/

public final void setStep(final double s) {

step = s;

}

/\*\*

\* Повертає коефіцієнт а еліпса.

\*

\* @return коефіцієнт а еліпса

\*/

public final double getA() {

return a;

}

/\*\*

\* Встановлює коефіцієнт а еліпса.

\*

\* @param newA

\* коефіцієнт а еліпса

\*/

public final void setA(final double newA) {

a = newA;

}

/\*\*

\* Повертає коефіцієнт b еліпса.

\*

\* @return коефіцієнт b еліпса

\*/

public final double getB() {

return b;

}

/\*\*

\* Встановлює коефіцієнт b еліпса.

\*

\* @param newB

\* коефіцієнт b еліпса

\*/

public final void setB(final double newB) {

b = newB;

}

/\*\*

\* Встановлює колір сектору за його індексом.

\*

\* @param index

\* індекс сектору

\* @param color

\* колір сектору

\*/

public final void setColor(final int index, final Color color) {

colors.set(index, color);

}

/\*\*

\* Повертає колір сектору за його індексом.

\*

\* @param index

\* індекс сектору

\* @return колір сектору

\*/

public final Color getColor(final int index) {

return colors.get(index);

}

/\*\*

\* Додає колір сектору.

\*

\* @param index

\* індекс сектру

\*/

public final void addColor(final int index) {

Color[] colorsName = {Color.BLUE, Color.MAGENTA, Color.ORANGE,

Color.PINK, Color.YELLOW, Color.GREEN, Color.RED, Color.WHITE };

colors.add(index, colorsName[index % (colorsName.length - 1)]);

}

/\*\*

\* Видаляє колір сектору.

\*

\* @param index

\* індекс сектру

\*/

public final void removeColor(final int index) {

colors.remove(index);

}

/\*\*

\* Повертає полігон сектора.

\*

\* @param begin

\* початок сектора

\* @param end

\* кінець сектора

\* @param disp

\* зміщення

\* @return полігон частини сектора

\*/

private Polygon getSector(final double begin, final double end,

final double disp) {

Polygon sector = new Polygon();

for (int i = (int) Math.round((end - begin) / step); i >= 0; i--) {

sector.addPoint((int) Math.round(cX + Math.cos(begin + i \* step)

\* a \* (r + disp)), (int) Math.round(cY

+ Math.sin(begin + i \* step) \* b \* (r + disp)));

}

for (int i = 0; i <= Math.round((end - begin) / step); i++) {

sector.addPoint((int) Math.round(cX), (int) Math.round(cY));

}

return sector;

}

/\*\*

\* Конструктор.

\*

\* @param t

\* таблиця значень

\*/

public SectorDiagramDrawer(final Object[][] t) {

final double fH = 10;

final double fStep = 0.005;

final double fA = 1;

final double fB = 1;

h = fH;

step = fStep;

a = fA;

b = fB;

table = t;

currentSector = -1;

createColors();

}

/\*\*

\* Генерує список кольорів секторів.

\*/

private void createColors() {

Color[] colorsName = {Color.BLUE, Color.MAGENTA, Color.ORANGE,

Color.PINK, Color.YELLOW, Color.GREEN, Color.RED, Color.WHITE };

for (int i = colors.size(); i < table[0].length; i++) {

colors.add(colorsName[i + 5 % (colorsName.length - 1)]);

}

}

/\*\*

\* Малює діаграму використовуючи даний графічний контекст.

\*

\* @param g

\* графічний контекст

\*/

public final void draw(final Graphics g) {

final double size = 0.9;

final int shiftX = 5;

final int shiftY = 3;

final int percent = 100;

final int polygonCount = 1;

cX = g.getClipBounds().getCenterX();

cY = g.getClipBounds().getCenterY();

if (cX < cY) {

r = size \* cX;

} else {

r = size \* cY;

}

int x, y;

FontMetrics fm = g.getFontMetrics();

double s = 0;

double disp;

String label;

double angle = 2 \* Math.PI;

double offset;

Color color;

polygons = new Polygon[polygonCount][table[1].length];

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

s += Double.valueOf((String) table[1][i]);

}

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

if (s == 0) {

offset = 2 \* Math.PI / table[1].length;

} else {

offset = 2 \* Math.PI \* Double.valueOf((String) table[1][i]) / s;

}

if (i == currentSector) {

disp = h;

} else {

disp = 0;

}

polygons[0][i] = getSector(angle - offset, angle, disp);

angle -= offset;

}

/\*\*

\* Малювання секторів

\*/

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

color = colors.get(i);

g.setColor(color);

g.fillPolygon(polygons[polygonCount - 1][i]);

if (currentSector == i) {

g.setColor(Color.BLACK);

} else {

g.setColor(color.darker().darker());

}

g.drawPolygon(polygons[polygonCount - 1][i]);

}

/\*\*

\* Підпис секторів

\*/

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

label = table[0][i]

+ " "

+ Math.round(Double.valueOf((String) table[1][i]) / s

\* percent) + "%";

x = (int) (polygons[polygonCount - 1][i].getBounds().getCenterX()

- fm.stringWidth(label) / 2);

y = (int) polygons[polygonCount - 1][i].getBounds().getCenterY();

g.setColor(Color.WHITE);

g.fillRect(x - shiftX, y - fm.getHeight() + shiftY, fm

.stringWidth(label)

+ shiftX, fm.getHeight() + shiftY);

g.setColor(Color.GRAY);

g.drawRect(x - shiftX, y - fm.getHeight() + shiftY, fm

.stringWidth(label)

+ shiftX, fm.getHeight() + shiftY);

g.setColor(Color.BLACK);

g.drawString(label, x, y);

}

}

/\*\*

\* Повертає номер сектору, в який входить дана точка.

\*

\* @param p

\* дана точка

\* @return номер сектору

\*/

private int find(final Point p) {

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

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

if (polygons[i][j].contains(p)) {

return j;

}

}

}

return -1;

}

/\*\*

\* Виконується при русі миші.

\*

\* @param event

\* подія миші

\*/

public final void mouseMoved(final MouseEvent event) {

Component comp = (Component) event.getSource();

if (find(event.getPoint()) != -1) {

comp.setCursor(new Cursor(Cursor.HAND\_CURSOR));

} else {

comp.setCursor(Cursor.getDefaultCursor());

}

currentSector = find(event.getPoint());

oldY = event.getY();

}

/\*\*

\* Виконується при одночасному натисканні та русі миші.

\*

\* @param event

\* подія миші

\*/

public final void mouseDragged(final MouseEvent event) {

final int percent = 200;

double s = 0;

Component comp = (Component) event.getSource();

if (currentSector != -1) {

comp.setCursor(new Cursor(Cursor.N\_RESIZE\_CURSOR));

if (event.getY() - oldY != 0) {

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

s += Double.valueOf((String) table[1][i]);

}

}

if (s == 0 && event.getY() - oldY != 0) {

s = 1;

} else {

s = s / percent;

}

if (event.getY() - oldY < 0) {

s = -s;

}

if (Double.valueOf((String) table[1][currentSector]) + s >= 0) {

table[1][currentSector] = String.valueOf(s

+ Double.valueOf((String) table[1][currentSector]));

}

oldY = event.getY();

}

}

/\*\*

\* Повертає поточний сектор.

\*

\* @return the currentSector поточний сектор

\*/

public final int getCurrentSector() {

return currentSector;

}

}

**TablePanel**

package com.gui;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.Point;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.util.Observable;

import java.util.Observer;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPopupMenu;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.ListSelectionModel;

import javax.swing.event.TableModelEvent;

import javax.swing.event.TableModelListener;

import javax.swing.table.AbstractTableModel;

/\*\*

\* @author Куцовол Віктор

\*

\*/

public class TablePanel extends JPanel implements Observer {

/\*\*

\* serialVersionUID.

\*/

private static final long serialVersionUID = 3064626846380266571L;

/\*\*

\* Таблиця.

\*/

private JTable table;

/\*\*

\* Модель діаграми.

\*/

private DiagramModel model;

/\*\*

\* Заголовок таблиці.

\*/

private String[] headings;

/\*\*

\* Контекстне меню. Викликається при натисненні правою кнопки миші.

\*/

private JPopupMenu jpu;

/\*\*

\* Посилання на фрейм.

\*/

private MainFrame frame;

/\*\*

\* Конструктор.

\*

\* @param fr

\* посилання на фрейм

\*/

public TablePanel(final MainFrame fr) {

frame = fr;

setModel(frame.getCurrentModel());

}

/\*\*

\* Встановлює модель діаграми.

\*

\* @param m

\* модель діаграми

\*/

@SuppressWarnings("serial")

public final void setModel(final DiagramModel m) {

model = m;

setLayout(new BorderLayout());

removeAll();

if (model != null) {

headings = new String[] {"Назва", "Значення"};

AbstractTableModel tm = new AbstractTableModel() {

public Object getValueAt(final int rowIndex,

final int columnIndex) {

return model.getTable()[columnIndex][rowIndex];

}

public void setValueAt(final Object aValue, final int rowIndex,

final int columnIndex) {

try {

if (columnIndex > 0) {

Double.valueOf((String) (aValue));

}

model.getTable()[columnIndex][rowIndex] = aValue;

fireTableDataChanged();

} catch (NumberFormatException e) {

System.out.println("Помилкові дані.");

JOptionPane.showOptionDialog(frame, "Помилкові дані",

"Помилка", JOptionPane.CLOSED\_OPTION,

JOptionPane.ERROR\_MESSAGE, null, null, null);

}

}

public int getRowCount() {

return model.getTable()[0].length;

}

public int getColumnCount() {

return model.getTable().length;

}

public String getColumnName(final int column) {

return headings[column];

}

public boolean isCellEditable(final int rowIndex,

final int columnIndex) {

return true;

}

};

tm.addTableModelListener(new TableModelListener() {

public void tableChanged(final TableModelEvent e) {

model.doChange("");

}

});

jpu = new JPopupMenu();

jpu.add(new JMenuItem(frame.getAddNewRowBeforeAction()));

jpu.add(new JMenuItem(frame.getAddNewRowAfterAction()));

jpu.add(new JMenuItem(frame.getRemoveRowAction()));

table = new JTable(tm);

table.addMouseListener(new MouseAdapter() {

public void mousePressed(final MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON3) {

Point point = e.getPoint();

int column = table.columnAtPoint(point);

int row = table.rowAtPoint(point);

if (column != -1 && row != -1) {

table.setColumnSelectionInterval(column, column);

table.setRowSelectionInterval(row, row);

}

}

if (e.isPopupTrigger()) {

jpu.show(e.getComponent(), e.getX(), e.getY());

}

}

public void mouseReleased(final MouseEvent e) {

if (e.isPopupTrigger()) {

jpu.show(e.getComponent(), e.getX(), e.getY());

}

}

});

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

table.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

JScrollPane jsp = new JScrollPane(table);

add(jsp, BorderLayout.WEST);

setPreferredSize(new Dimension(table.getPreferredSize().width,

getPreferredSize().height));

}

repaint();

}

/\*\*

\* Виконується, коли змінюється об'єкт, за яким ведеться нагляд.

\*

\* @param o

\* об'єкт, за яким ведеться нагляд

\* @param arg

\* значення, яке пересилається від об'єкта, за яким ведеться

\* нагляд

\*/

public final void update(final Observable o, final Object arg) {

if (((String) arg) != "") {

table.updateUI();

} else {

repaint();

}

}

/\*\*

\* Повертає таблицю.

\*

\* @return таблицю

\*/

public final JTable getTable() {

return table;

}

}

**AppSettings**

package com.main;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Properties;

import javax.swing.JOptionPane;

/\*\*

\* Клас налаштувань. Налаштування зберігаються у файлі "settings.txt".

\*

\* @author Куцовол Віктор

\*/

public final class AppSettings {

/\*\*

\* Екземпляр класу AppSettings.

\*/

private static AppSettings instance = new AppSettings();

/\*\*

\* Екземпляр класу Properties.

\*/

private Properties properties;

/\*\*

\* Конструктор.

\*/

private AppSettings() {

FileReader fr = null;

properties = new Properties();

try {

fr = new FileReader(Main.class.getResource("settings.txt").

getPath());

} catch (FileNotFoundException e) {

System.out.println("Відсутній файл налаштувань.");

JOptionPane.showOptionDialog(null, Main.class.

getResource("settings.txt").getPath(),

"Помилка", JOptionPane.CLOSED\_OPTION,

JOptionPane.ERROR\_MESSAGE, null, null, null);

}

try {

if (fr != null) {

properties.load(fr);

fr.close();

}

} catch (IOException e) {

System.out.println("Помилка завантаження налаштувань.");

JOptionPane.showOptionDialog(null,

"Помилка завантаження налаштувань", "Помилка",

JOptionPane.CLOSED\_OPTION, JOptionPane.ERROR\_MESSAGE, null,

null, null);

}

}

/\*\*

\* Повертає екземпляр AppSettings.

\*

\* @return екземпляр AppSettings

\*/

public static AppSettings getInstance() {

return instance;

}

/\*\*

\* Зберігає налаштування у файлі "settings.txt".

\*

\* @throws IOException

\* виникає внаслідок помилки вводу/виводу

\*/

public void save() throws IOException {

FileWriter fw = new FileWriter(Main.class.

getResource("settings.txt").getPath());

properties.store(fw, "Diagram viewer");

}

/\*\*

\* Додає до хештейбла значення та зв'язує з його ключем.

\*

\* @param key

\* ключ

\* @param value

\* значення

\*/

public void put(final Object key, final Object value) {

properties.put(key, value);

}

/\*\*

\* Повертає значення за ключем.

\*

\* @param key

\* ключ

\* @return значення за ключем

\*/

public Object get(final Object key) {

return properties.get(key);

}

}

**Main**

package com.main;

import javax.swing.JFrame;

import javax.swing.SwingUtilities;

import com.gui.MainFrame;

/\*\*

\* Головний клас.

\*

\* @author Куцовол Віктор

\*

\*/

public final class Main {

/\*\*

\* Конструктор.

\*/

private Main() {

super();

}

/\*\*

\* Виконується при запуску програми.

\*

\* @param args

\* параметри з командного рядка

\*/

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

SwingUtilities.invokeLater(new Runnable() {

public void run() {

MainFrame frame = new MainFrame();

frame.setTitle("Diagram Drawer");

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

frame.setVisible(true);

}

});

}

}

#### **Висновки.**

Результатом проробленої роботи є повністю функціональний програмний продукт. Основна цільова аудиторія користувачів програми є бізнесова аудиторія, а саме: економісти, бухгалтери, аналітики. Продукт є зручним додатком до бізнес – плану або звіту динаміки будь-яких даних.

#### **Список літератури.**

* 1. <http://ru.wikipedia.org/wiki/>
  2. Эккель. Философия Java.
  3. <http://java.sun.com/javase/6/docs/api/>