Міністерство освіти та науки України

Національний технічний університет України

«Київський Політехнічний Інститут»

Кафедра обчислювальної техніки

**Курсова робота**

з дисципліни

«Інженерія програмного забезпечення»

Виконала:

Студентка ФІОТ

Групи ІО-01

Наумова Х.С.

зал.книжка № 101

Київ 2012 г.

ЗМІСТ

[ВСТУП 3](#_Toc319596709)

[1 Технічне завдання 4](#_Toc319596710)

[1.1 Загальне завдання 4](#_Toc319596711)

[1.2 Функціональність 4](#_Toc319596712)

[1.3 Вимоги до реалізації 4](#_Toc319596713)

[1.4 Варіант завдання 5](#_Toc319596714)

[2 Проектування інтерфейсу користувача 6](#_Toc319596715)

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

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

2.3[Розробка орієнтирів 8](#_Toc319596717)

2.4[Діаграма граничних класів 9](#_Toc319596718)

[3 Розробка програмного забезпечення 10](#_Toc319596722)

3.1[Скріншоти програми 11](#_Toc319596723)

3.2[Специфікація класів 13](#_Toc319596724)

[ВИСНОВКИ 32](#_Toc319596725)

[СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 33](#_Toc319596726)

[ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЕКТУ 35](#_Toc319596727)

[ДОДАТОК Б. СТРУКТУРА ПРОЕКТУ 78](#_Toc319596728)

ВСТУП

У курсовій роботі розробляється програма для побудови «стовпчастих» діаграм на основі пакета SWING. Необхідно реалізувати інтерфейс для зручного користування додатком і зручним наданням графічної інформації.

Так само необхідною частиною програми є інструкція користування у вигляді скріншотів з поясненням для користувача, і JavaDoc для розробника.

У роботі розглядаються основні принципи побудови графічного інтерфейсу користувача, основні елементи інтерфейсу, організація обробки подій, система відлову помилок і організація багатопоточності.

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

# Технічне завдання

## Загальне завдання

Необхідно побудувати програму візуалізації табличних даних у вигляді діаграми. В якості вхідних даних виступає таблиця, що знаходиться у CSV- файлі. Кожен з рядків таблиці є описом елементу діаграми – сектора з можливими додатковими атрибутами.

## 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 – її інтерпретацію у вигляді діаграми згідно варіанту;
* можливість редагування даних (модифікація елемента, вставка/видалення елемента) в одній області з синхронною зміною іншої області;
* можливість збереження результату графічної інтерпретації - діаграми у вигляді файлу формату JPEG.

## 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.

## Варіант завдання

Назва діаграми – стовпчата.

Вхідні дані (елемент таблиці) — дані з CSV-файлу, або вводимо у таблицю

Вид редагування – редагування графічної області з синхронною модифікацією табличної області.

# Проектування інтерфейсу користувача

## 2.1UML діаграма класів



Рисунок 2.1.Діаграма класів

**2.2 Графічний інтерфейс користувача**

**Призначення:**

Основне вікно програми використовується для створення і редагування стовпчастих діаграм.

1. Прецедент починається, коли користувач запускає програму

2. Користувач може виконати одну з дій:

1. Відкрити CSV файл

2. Вийти з програми.

3. Побудувати діаграму.

4. Побудувати діаграму в новому вікні

5. Зберегти дані як CSV файл

6. Зберегти діграму у форматі PNG

7. Видалити вибрані елементи натиснувши клавішу enter

8. Мітка редагування розміру головного вікна

Прецедент закінчується, при закрітті Програми.

**2.3 Розробка орієнтирів**

1. Прецедент починається коли користувач запускає програму

2. Користувач може виконати одну з дій:

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

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

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

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

5. Змінити дані на графіку [Автоматично змінюється таблична область] (Виконується в 75% випадків)

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

**2.4 Діаграма граничних класів:**

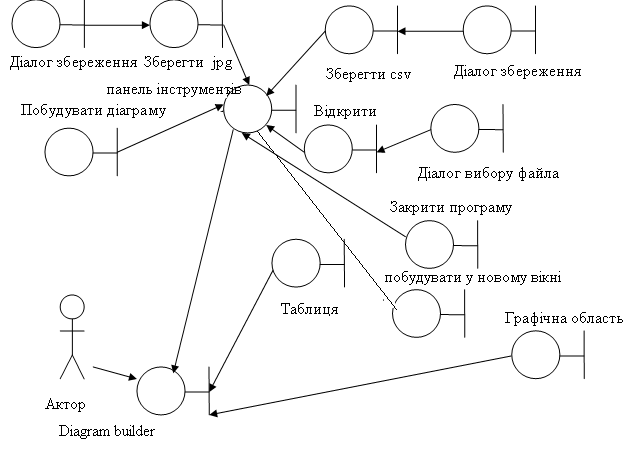
****

Рисунок 2.2. Діаграма граничних класів

# Розробка програмного забезпечення

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

Таблиця 3.1. Відповідності елементів

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

**3.2 Скріншоти програми**

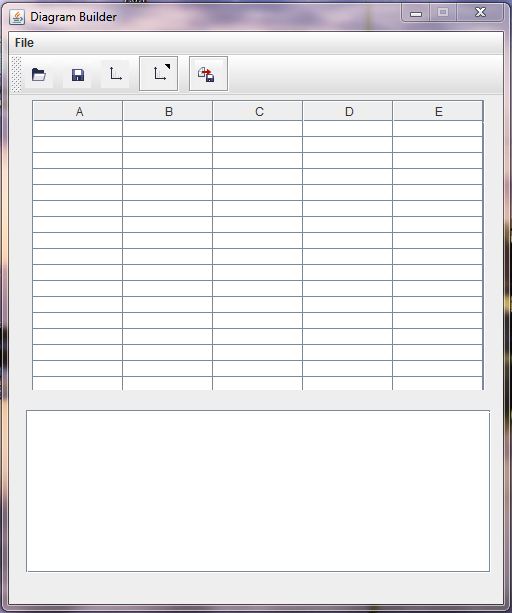
****

Рисунок 3.1. Створення нової діаграми

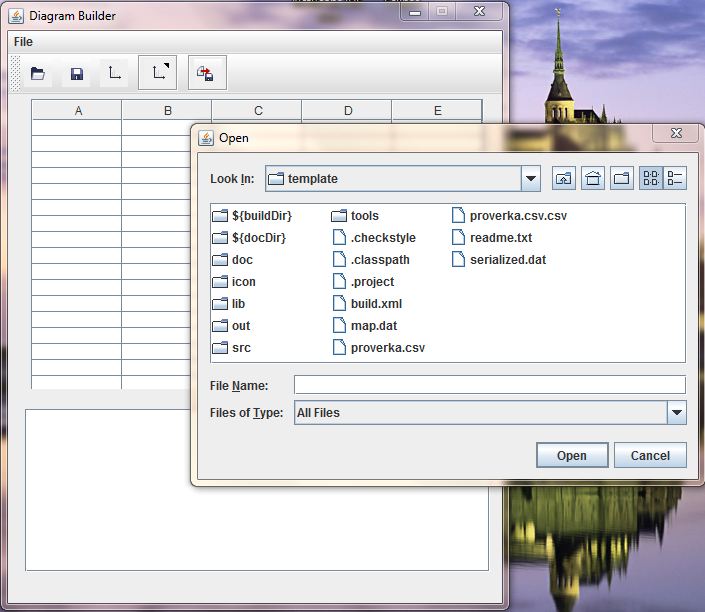
****

Рисунок 3.2. Відкриття файлу

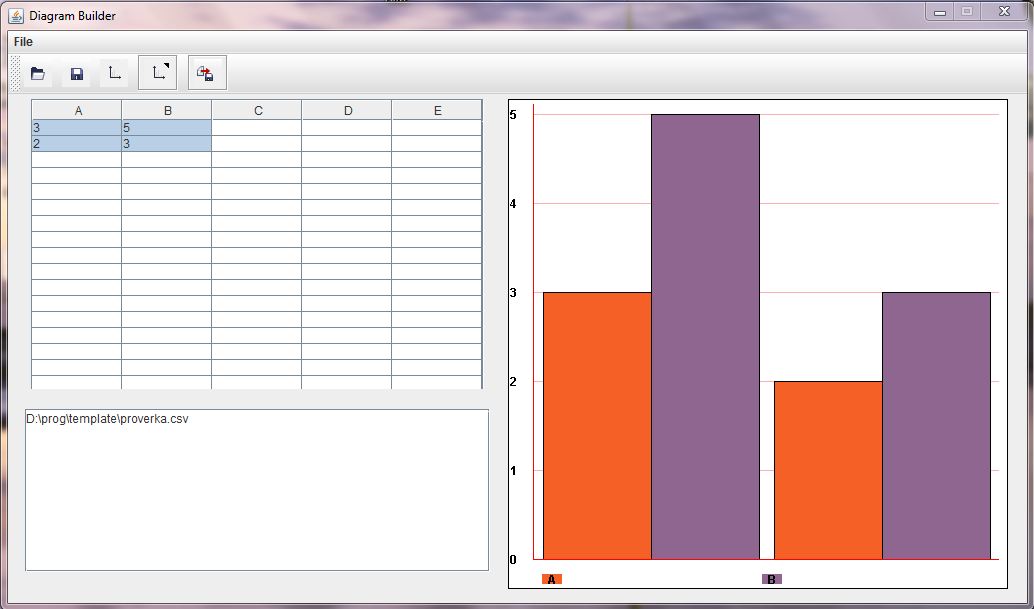


Рисунок 3.3. Побудована діаграма

Специфікація класів

**diagramdrower   
Class TestIO**

java.lang.Object

extended by **diagramdrower.TestIO**

public class **TestIO**

extends java.lang.Object

**Author:**

kris

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**TestIO**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\TestIO.html#TestIO())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| static void | [**main**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\TestIO.html#main(java.lang.String[]))(java.lang.String[] args)            main class which call methods for processing csv file |

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

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

**TestIO**

public **TestIO**()

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

**main**

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

main class which call methods for processing csv file

**Parameters:**

args -

**Throws:**

CSVParseExceptin - for catching errors during working with csv files

IOException

java.lang.ClassNotFoundException

**diagramdrower   
Class CalcRtects**

java.lang.Object

extended by **diagramdrower.CalcRtects**

public class **CalcRtects**

extends java.lang.Object

class for calculate parameters of rectangles

**Author:**

kris

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**CalcRtects**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CalcRtects.html#CalcRtects(int[][], int, int, java.awt.Dimension))(int[][] array, int nrows, int ncolum, java.awt.Dimension size) |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**calcRect**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CalcRtects.html#calcRect())()            method for calculetion sizes of rectangles |
| int | [**getMax**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CalcRtects.html#getMax())()            get max element of array |
| java.util.ArrayList<java.awt.geom.Rectangle2D> | [**getRects**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CalcRtects.html#getRects())() |
| double | [**getScale**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CalcRtects.html#getScale())()            scale for build rect. |

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

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

**CalcRtects**

public **CalcRtects**(int[][] array,

int nrows,

int ncolum,

java.awt.Dimension size)

**Parameters:**

array - of data

nrows - number of rows

ncolum - number of columns

size - of window

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

**calcRect**

public void **calcRect**()

method for calculetion sizes of rectangles

**getRects**

public java.util.ArrayList<java.awt.geom.Rectangle2D> **getRects**()

**Returns:**

array of rectangles

**getScale**

public double **getScale**()

scale for build rect.

**Returns:**

scale

**getMax**

public int **getMax**()

get max element of array

**Returns:**

max element

**diagramdrower   
Class CSVParseException**

java.lang.Object

extended by java.lang.Throwable

extended by java.lang.Exception

extended by java.io.IOException

extended by **diagramdrower.CSVParseException**

**All Implemented Interfaces:**

java.io.Serializable

public class **CSVParseException**

extends java.io.IOException

user exception for working with csv files and table data

**Author:**

kris

**See Also:**

[Serialized Form](file:///D:\java_spase\DiagramDrower\doc\serialized-form.html#diagramdrower.CSVParseException)

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**CSVParseException**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVParseException.html#CSVParseException())()            Constructs CSVParseException with null As the details of error messages |  |
| [**CSVParseException**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVParseException.html#CSVParseException(java.lang.String))(java.lang.String msg)            Constructs with CSVParseException a specific message about the details error. |  |

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

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

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

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

**CSVParseException**

public **CSVParseException**()

Constructs CSVParseException with null As the details of error messages

**CSVParseException**

public **CSVParseException**(java.lang.String msg)

Constructs with CSVParseException a specific message about the details error. @ Param msg error details

**diagramdrower   
Class CSVProcessor**

java.lang.Object

extended by **diagramdrower.CSVProcessor**

public class **CSVProcessor**

extends java.lang.Object

class for processing CSV file

**Author:**

kris

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**CSVProcessor**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#CSVProcessor())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| int | [**getColum**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#getColum())() |
| java.util.ArrayList<java.lang.String> | [**getMyLst**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#getMyLst())() |
| int | [**getRows**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#getRows())() |
| java.lang.String[][] | [**parse**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#parse(java.util.ArrayList))(java.util.ArrayList<java.lang.String> lst)            method for parse data |
| java.util.ArrayList<java.lang.String> | [**readF**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#readF(java.lang.String))(java.lang.String fname)            method for read rows from file data recorded into an array |
| void | [**writeIn**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html#writeIn(java.lang.String, java.util.ArrayList))(java.lang.String fname, java.util.ArrayList<java.lang.String> lst)            method for record elements of array to file |

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

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

**CSVProcessor**

public **CSVProcessor**()

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

**getMyLst**

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

**Returns:**

array of data

**readF**

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

throws java.io.IOException,

java.lang.ClassNotFoundException

method for read rows from file data recorded into an array

**Parameters:**

fname - name of the file for reading

**Returns:**

array list where store data after reading file

**Throws:**

java.io.IOException

java.lang.ClassNotFoundException

**writeIn**

public void **writeIn**(java.lang.String fname,

java.util.ArrayList<java.lang.String> lst)

throws java.io.IOException

method for record elements of array to file

**Parameters:**

fname - name of the file for writing

lst - array list where store data

**Throws:**

java.io.IOException

**parse**

public java.lang.String[][] **parse**(java.util.ArrayList<java.lang.String> lst)

throws java.io.IOException,

java.lang.ClassNotFoundException

method for parse data

**Parameters:**

lst - Array list, where store data

**Returns:**

two dimensional array of data

**Throws:**

java.lang.ClassNotFoundException

java.io.IOException

**getRows**

public int **getRows**()

**Returns:**

number of rows

**getColum**

public int **getColum**()

**Returns:**

number of columns

**diagramdrower   
Class CSVThread**

java.lang.Object

extended by java.lang.Thread

extended by **diagramdrower.CSVThread**

**All Implemented Interfaces:**

java.lang.Runnable

public class **CSVThread**

extends java.lang.Thread

class for parse date in other thread

**Author:**

kris

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

|  |
| --- |
| **Nested classes/interfaces inherited from class java.lang.Thread** |
| java.lang.Thread.State, java.lang.Thread.UncaughtExceptionHandler |

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class java.lang.Thread** |
| MAX\_PRIORITY, MIN\_PRIORITY, NORM\_PRIORITY |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**CSVThread**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVThread.html#CSVThread(diagramdrower.CSVProcessor))([CSVProcessor](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html) prcs) |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| int | [**getJj**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVThread.html#getJj())() |
| int | [**getKk**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVThread.html#getKk())() |
| int[][] | [**getMas**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVThread.html#getMas())() |
| void | [**run**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVThread.html#run())() |

|  |
| --- |
| **Methods inherited from class java.lang.Thread** |
| activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getAllStackTraces, getContextClassLoader, getDefaultUncaughtExceptionHandler, getId, getName, getPriority, getStackTrace, getState, getThreadGroup, getUncaughtExceptionHandler, holdsLock, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setDefaultUncaughtExceptionHandler, setName, setPriority, setUncaughtExceptionHandler, sleep, sleep, start, stop, stop, suspend, toString, yield |

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

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

**CSVThread**

public **CSVThread**([CSVProcessor](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\CSVProcessor.html) prcs)

**Parameters:**

csvp - variable of CSVProcessor class link into CSVProcessor

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

**getKk**

public int **getKk**()

**Returns:**

number of rows

**getJj**

public int **getJj**()

**Returns:**

number of columns

**run**

public void **run**()

**Specified by:**

run in interface java.lang.Runnable

**Overrides:**

run in class java.lang.Thread

**getMas**

public int[][] **getMas**()

**Returns:**

array

**diagramdrower   
Class DiagramDrawer**

java.lang.Object

extended by java.awt.Component

extended by java.awt.Container

extended by java.awt.Window

extended by java.awt.Frame

extended by javax.swing.JFrame

extended by **diagramdrower.DiagramDrawer**

**All Implemented Interfaces:**

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

public class **DiagramDrawer**

extends javax.swing.JFrame

class for creating frame

**Author:**

kris

**See Also:**

[Serialized Form](file:///D:\java_spase\DiagramDrower\doc\serialized-form.html#diagramdrower.DiagramDrawer)

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

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.BaselineResizeBehavior |

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class javax.swing.JFrame** |
| EXIT\_ON\_CLOSE |

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

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

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

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

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**DiagramDrawer**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#DiagramDrawer())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**build**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#build(boolean))(boolean newFrm)            method for build diagram |
| void | [**delete**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#delete())()            method for delete table |
| java.awt.Color[] | [**generateColors**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#generateColors())()            method for generate colors of diagram |
| int | [**getSelCol**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#getSelCol())() |
| int | [**getSelRow**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#getSelRow())() |
| void | [**openCSV**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#openCSV())()            method for open csv file |
| void | [**readFile**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#readFile())()            method for read csv file |
| void | [**STbl**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramDrawer.html#STbl())()            method for saving table |

|  |
| --- |
| **Methods inherited from class javax.swing.JFrame** |
| getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar, setLayeredPane, setLayout, setTransferHandler, update |

|  |
| --- |
| **Methods inherited from class java.awt.Frame** |
| addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated |

|  |
| --- |
| **Methods inherited from class java.awt.Window** |
| addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOwnedWindows, getOwner, getOwnerlessWindows, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, paint, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImages, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setVisible, show, toBack, toFront |

|  |
| --- |
| **Methods inherited from class java.awt.Container** |
| add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate |

|  |
| --- |
| **Methods inherited from class java.awt.Component** |
| action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocusInWindow, resize, resize, setBackground, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusUpCycle |

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

|  |
| --- |
| **Methods inherited from interface java.awt.MenuContainer** |
| getFont, postEvent |

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

**DiagramDrawer**

public **DiagramDrawer**()

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

**STbl**

public void **STbl**()

method for saving table

**generateColors**

public java.awt.Color[] **generateColors**()

method for generate colors of diagram

**getSelRow**

public int **getSelRow**()

**getSelCol**

public int **getSelCol**()

**openCSV**

public void **openCSV**()

method for open csv file

**readFile**

public void **readFile**()

method for read csv file

**delete**

public void **delete**()

method for delete table

**build**

public void **build**(boolean newFrm)

method for build diagram

**Parameters:**

newFrm -

**diagramdrower   
Class DiagramPanel**

java.lang.Object

extended by java.awt.Component

extended by java.awt.Container

extended by javax.swing.JComponent

extended by javax.swing.JPanel

extended by **diagramdrower.DiagramPanel**

**All Implemented Interfaces:**

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

public class **DiagramPanel**

extends javax.swing.JPanel

**See Also:**

[Serialized Form](file:///D:\java_spase\DiagramDrower\doc\serialized-form.html#diagramdrower.DiagramPanel)

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

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

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.BaselineResizeBehavior |

|  |
| --- |
| **Field Summary** |

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

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

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

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**DiagramPanel**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramPanel.html#DiagramPanel(int[][], int, int, java.lang.String[], java.awt.Color[]))(int[][] array, int nrows, int ncolum, java.lang.String[] colNam, java.awt.Color[] col) |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**paintComponent**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\DiagramPanel.html#paintComponent(java.awt.Graphics))(java.awt.Graphics g)            method for creating graphics |

|  |
| --- |
| **Methods inherited from class javax.swing.JPanel** |
| getAccessibleContext, getUI, getUIClassID, setUI, updateUI |

|  |
| --- |
| **Methods inherited from class javax.swing.JComponent** |
| addAncestorListener, addNotify, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintImmediately, paintImmediately, print, printAll, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update |

|  |
| --- |
| **Methods inherited from class java.awt.Container** |
| add, add, add, add, add, addContainerListener, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, getMousePosition, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, remove, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout, transferFocusBackward, transferFocusDownCycle, validate |

|  |
| --- |
| **Methods inherited from class java.awt.Component** |
| action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, createImage, createImage, createVolatileImage, createVolatileImage, dispatchEvent, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle |

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

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

**DiagramPanel**

public **DiagramPanel**(int[][] array,

int nrows,

int ncolum,

java.lang.String[] colNam,

java.awt.Color[] col)

**Parameters:**

array - of data

nrows - number of rows

ncolum - number of columns

colNam - name of columns

col - color of rectangles

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

**paintComponent**

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

method for creating graphics

**Overrides:**

paintComponent in class javax.swing.JComponent

**diagramdrower   
Class FrameDrower**

java.lang.Object

extended by java.awt.Component

extended by java.awt.Container

extended by java.awt.Window

extended by java.awt.Frame

extended by javax.swing.JFrame

extended by **diagramdrower.FrameDrower**

**All Implemented Interfaces:**

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

public class **FrameDrower**

extends javax.swing.JFrame

class for creating window

**Author:**

kris

**See Also:**

[Serialized Form](file:///D:\java_spase\DiagramDrower\doc\serialized-form.html#diagramdrower.FrameDrower)

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

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.BaselineResizeBehavior |

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class javax.swing.JFrame** |
| EXIT\_ON\_CLOSE |

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

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

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

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

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**FrameDrower**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\FrameDrower.html#FrameDrower())() |  |
| [**FrameDrower**](file:///D:\java_spase\DiagramDrower\doc\diagramdrower\FrameDrower.html#FrameDrower(int, int))(int width, int height) |  |

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

|  |
| --- |
| **Methods inherited from class javax.swing.JFrame** |
| getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar, setLayeredPane, setLayout, setTransferHandler, update |

|  |
| --- |
| **Methods inherited from class java.awt.Frame** |
| addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated |

|  |
| --- |
| **Methods inherited from class java.awt.Window** |
| addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOwnedWindows, getOwner, getOwnerlessWindows, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, paint, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImages, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setVisible, show, toBack, toFront |

|  |
| --- |
| **Methods inherited from class java.awt.Container** |
| add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate |

|  |
| --- |
| **Methods inherited from class java.awt.Component** |
| action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocusInWindow, resize, resize, setBackground, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusUpCycle |

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

|  |
| --- |
| **Methods inherited from interface java.awt.MenuContainer** |
| getFont, postEvent |

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

**FrameDrower**

public **FrameDrower**()

**FrameDrower**

public **FrameDrower**(int width,

int height)

**Parameters:**

width - of window

height - of window

ВИСНОВКИ

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

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

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

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

Вивчила організацію потоків, серіалізаціі і десеріалізаціі даних.

В результаті розробки всі пункти технічного завдання було виконано.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Герберт Шилдт SWING: руководство для начинающих – М.: “Вильямс”, 2007. – С. 704. – ISBN 0-07-226314-8.
2. Эккель Б. Философия Java / Эккель Брюс; Пер.с англ. Е.Матвеев.– 4-е изд.–СПб.: Питер, 2010. – 640с.: ил. – (Библиотека программиста). – Алф.указ.:с.631. – ISBN 978-5-388-00003-3.
3. Приемы объектно-ориентированого проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. – СПб.: Питер, 2010 – 368 с.: ил. –ISBN 978-5-469-01136-1.
4. Хорстманн Кей С. Java 2. Том 1. Основы / Кей Хорстманн, Гари Корнелл; Пер с англ. – Изд. 8-е. – М.: ООО “И.ДВильямс”, 2011. – 816 c.: ил. – Парал. тит. англ. – (Библиотека профессионала). –ISBN 978-5-8459-1378-4 (рус.).
5. <http://www.uic.rsu.ru/doc/programming/java/TIJ2e.ru/Chapter11.html>

**ДОДАТКИ**

ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЕКТУ

package diagr22;

import java.io.\*;

import java.util.ArrayList;

import java.util.StringTokenizer;

/\*\*

\* клас для обробки CSV file

\*/

public class CSVProcessor {

// array of data where store data from file

private ArrayList<String> myLst;

/\*\*

\* рядок

\*/

int trows = 0;

/\*\*

\* стовпчик

\*/

int tcolum = 0;

/\*\*

\*

\* @повертає масив даних

\*/

public ArrayList<String> getMyLst() {

return myLst;

}

/\*\*

\* метод для зчитування рядків із файлу даних записаного у масив

\*

\* @param fname

\* назва файлу для считування

\* @повертає масив де зберігаються дані із файлу після считування

\* @throws IOException

\* @throws ClassNotFoundException

\*/

public ArrayList<String> readF(String fname) throws IOException,

ClassNotFoundException {

String line = null;

if (fname.endsWith("csv/")) {

File file = new File(fname);

BufferedReader br = new BufferedReader(new InputStreamReader(

new FileInputStream(file), "Windows-1251"));

myLst = new ArrayList<String>();

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

myLst.add(line);

}

br.close();

}

else {

String err = "Error! file is not CSV";

throw new CSVParseException(err);

}

return myLst;

}

/\*\*

\* методя для запису єлементів масиву у файл

\*

\* @param fname

\* назва файлу для запису

\* @param lst

\* масив де зберігаються дані

\* @throws IOException

\*/

public void writeIn(String fname, ArrayList<String> lst) throws IOException {

PrintWriter out1 = new PrintWriter(new BufferedWriter(new FileWriter(

fname)));

for (String elem : lst) {

out1.println(elem);

}

out1.close();

}

/\*\*

\* методя для обробки даних

\*

\* @param lst

\* масив де зберігаються дані

\* @повертає два масиви даних

\* @throws ClassNotFoundException

\* @throws IOException

\*/

public String[][] parse(ArrayList<String> lst) throws IOException,

ClassNotFoundException {

String[][] mass = new String[50][50];

int i = 0;

int j = 0;

String data = null;

for (String elem : lst) {

StringTokenizer a = new StringTokenizer(elem, ";");

j = 0;

while (a.hasMoreTokens()) {

data = a.nextToken();

mass[i][j] = data;

j++;

}

i++;

}

while (mass[0][tcolum] != null) {

tcolum++;

}

while (mass[trows][0] != null) {

trows++;

}

return mass;

}

/\*\*

\*

\* @повертає кількість рядків

\*/

public int getRows() {

return trows;

}

/\*\*

\*

\* @повертає кількість стовпчиків

\*/

public int getColum() {

return tcolum;

}

}

**package** diagr22;

**import** java.util.ArrayList;

/\*\*

\* клас для обробки даних у іншому потоці

\*

\* **@author** kris

\*/

**public** **class** CSVThread **extends** Thread {

//value of class of CSVProcessor

CSVProcessor csvp = **new** CSVProcessor();

//array of data where store data from file

ArrayList<String> myList;

/\*\*

\* **@param** csvp

\* змінна класу CSVProcessor яка підключена до CSVProcessor

\*/

**public** CSVThread(CSVProcessor prcs) {

**this**.csvp = prcs;

**this**.myList = prcs.getMyLst();

}

//масив де зберігаються дані із файлу після обробки String[][] mas = **new** String[50][50];

// масив де зберігаються дані після переводу із String формату

**int**[][] masInt = **new** **int**[50][50];

//рядки масиву

**int** kk = 0;

/\*\*

\*

\* **@повертає кількість рядків**

\*/

**public** **int** getKk() {

**return** kk;

}

/\*\*

\*

\* **@повертає кількість стовпчиків**

\*/

**public** **int** getJj() {

**return** jj;

}

**int** jj = 0;

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

**try** {

mas = csvp.parse(myList);

**while** (mas[0][kk] != **null**) {

kk++;

}

**while** (mas[jj][0] != **null**) {

jj++;

}

**for** (**int** i = 0; i < jj; i++) {

**for** (**int** j = 0; j < kk; j++) {

masInt[i][j] = Integer.*parseInt*(mas[i][j]);

}

}

} **catch** (Exception e) {

e.printStackTrace();

System.*err*.println(e + " ");

}

}

/\*\*

\* **@повертає масив**

\*/

**public** **int**[][] getMas() {

**return** masInt;

}

}

**package** diagr22;

**import** java.io.IOException;

/\*\*

\*користувацьке виключення для роботи з csv файлом і таблицею даних

\*

\* **@author** kris

\*/

**public** **class** CSVParseException **extends** IOException {

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

/\*\*

\* створює CSVParseException з null як деталлю повідомлення про помилку

\*/

**public** CSVParseException() {

**super**();

}

/\*\*

\* створює з CSVParseException спеціальне повідомлення про деталі помилки.

\*/

**public** CSVParseException(String msg) {

**super**(msg);

}

}

**package** diagr22;

**import** javax.swing.\*;

/\*\*

\*

\* **@author** kris

\*

\*/

**public** **class** TestIO

{

/\*\*

\* головний клас який викликає методи для обробки csv файлу

\* **@throws** CSVParseExceptin для виявлення помилки протягом роботи з csv файлом

\* **@param** args

\* **@throws** IOException

\* **@throws** ClassNotFoundException

\*/

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

{

SwingUtilities.*invokeLater*(**new** Runnable()

{

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

{

**new** DiagramDrawer();

}

});

}

}

**package** diagr22;

**import** javax.swing.JFrame;

/\*\*

\* клас для створення вікна

\*

\* **@author** kris

\*

\*/

**public** **class** FrameDrower **extends** JFrame {

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

**public** FrameDrower() {

setSize(500, 550);

setVisible(**true**);

setResizable(**true**);

setTitle("Diagram");

}

/\*\*

\*

\* **@param** width

\* ширина вікна

\* **@param** height

\* висота вікна

\*/

**public** FrameDrower(**int** width, **int** height) {

setSize(width, height);

setVisible(**true**);

setResizable(**true**);

setTitle("Diagram");

}

}

package diagr22;

import java.awt.Dimension;

import java.awt.Rectangle;

import java.awt.geom.Rectangle2D;

import java.util.ArrayList;

/\*\*

\* клас для рахування параметрів прямокутників

\*

\* @author kris

\*

\*/

public class CalcRtects {

/\*\*

\* розміри панелі

\*/

Dimension size;

/\*\*

\* масив де зберігаються дані із таблиці

\*/

int[][] arr;

/\*\*

\* кількість рядків

\*/

int nrows;

/\*\*

\* кількість стовпчиків

\*/

int ncolum;

/\*\*

\* шкала для будування діаграми

\*/

double scale;

/\*\*

\* максимальний елемент масиву

\*/

int scl;

/\*\*

\*масив де зберігаються прямокутники для будування діаграми

\*/

ArrayList<Rectangle2D> rects = new ArrayList<Rectangle2D>();

/\*\*

\*

\* @param array

\* масив даних

\* @param nrows

\* кількість рядків

\* @param ncolum

\* кількість стовпчиків

\* @param size

\* розмір вікна

\*/

public CalcRtects(int[][] array, int nrows, int ncolum, Dimension size) {

this.size = size;

this.nrows = nrows;

this.ncolum = ncolum;

this.arr = array;

}

/\*\*

\* метод для обчислення розміру прямокутника

\*/

public void calcRect() {

scl = arr[0][0];

for (int i = 0; i < nrows; i++) {

for (int j = 0; j < ncolum; j++) {

if (arr[i][j] > scl) {

scl = arr[i][j];

}

}

}

if (scl == 0) {

scl = 1;

}

if (scl > size.height - 45) {

scale = (size.height - 45) \* 1.0 / scl;

} else {

scale = (size.height - 45) / scl;

}

int wRt = (size.width - 35 - 15 \* nrows) / (nrows \* ncolum);

int wd = 35;

for (int j = 0; j < nrows; j++) {

for (int i = 0; i < ncolum; i++) {

int znach = (int) (arr[j][i] \* scale);

Rectangle2D rect = new Rectangle(wd, size.height - 30 - znach,

wRt, znach);

rects.add(rect);

wd += wRt;

}

wd += 15;

}

}

/\*\*

\*

\* @повертає масив прямокутників

\*/

public ArrayList<Rectangle2D> getRects() {

calcRect();

return rects;

}

/\*\*

\* шкала для будування прямокутників

\*

\* @повертає шкалу

\*/

public double getScale() {

return scale;

}

/\*\*

\* отримання макс елементу масиву

\*

\* @повертає макс елемент

\*/

public int getMax() {

int max = scl;

return max;

}

}

package diagr22;

import java.awt.Color;

import java.awt.Cursor;

import java.awt.Dimension;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Random;

import javax.imageio.ImageIO;

import javax.swing.AbstractAction;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JToolBar;

/\*\*

\* class for creating frame

\*

\* @author kris

\*/

public class DiagramDrawer extends JFrame {

private static final long serialVersionUID = 1L;

/\*\*

\* width of frame

\*/

private int scrWidth;

/\*\*

\* height of frame

\*/

private int scrHeight;

/\*\*

\* dimension of frame

\*/

private Dimension fSize;

/\*\*

\* array of colors

\*/

private Color[] color;

/\*\*

\* table

\*/

private JTable table;

/\*\*

\* текстова область

\*/

private JTextArea textArea;

/\*\*

\* флаг збереження картинки

\*/

private int fl = 0;

/\*\*

\* копіювання CSVProcessor

\*/

private CSVProcessor csvp = new CSVProcessor();

/\*\*

\* задає назву стовпчикам

\*/

private String[] colName;

/\*\*

\* пусть до цсв файлу

\*/

private String way = null;

/\*\*

\* масив даних який считався із цсв файлу

\*/

private int[][] arr = new int[50][50];

/\*\*

\* беферізація зображення

\*/

protected static final BufferedImage BufferedImage = null;

private DiagramPanel dp;

private FrameDrower frame;

/\*\*

\* масив вибраних рядків

\*/

private int[] rows;

/\*\*

\* масив вибраних стовпчиків

\*/

private int[] colum;

private JFileChooser file;

public DiagramDrawer() {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screen = toolkit.getScreenSize();

scrWidth = screen.width;

scrHeight = screen.height;

setTitle("Diagram Builder");

fSize = new Dimension(500, 600);

setSize(fSize.width, fSize.height);

setLocation(scrWidth / 2 - fSize.width / 2, scrHeight / 2

- fSize.height / 2);

setResizable(false);

Image image = toolkit.getImage("icon/icon.jpg ");

setIconImage(image);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

setLayout(null);

JPanel buttonPanel = new JPanel();

buttonPanel.setBounds(0, 40, 500, 295);

buttonPanel.setFocusable(true);

add(buttonPanel);

JToolBar toolbar = new JToolBar();

toolbar.setBounds(0, 0, 1050, 40);

add(toolbar);

JButton openButton = new JButton(new ImageIcon("icon/folder.png"));

openButton.setBorder(null);

openButton.setSize(32, 32);

openButton.setToolTipText("Open new File");

toolbar.add(openButton);

toolbar.addSeparator();

JButton saveButton = new JButton(new ImageIcon("icon/save.png"));

saveButton.setBorder(null);

saveButton.setSize(32, 32);

saveButton.setToolTipText("Save as");

toolbar.add(saveButton);

toolbar.addSeparator();

JButton buildButton = new JButton(new ImageIcon("icon/build.png"));

buildButton.setBorder(null);

buildButton.setSize(32, 32);

buildButton.setToolTipText("Build");

toolbar.add(buildButton);

toolbar.addSeparator();

JButton builNewWinButton = new JButton(

new ImageIcon("icon/newwind.png"));

builNewWinButton.setSize(32, 32);

builNewWinButton.setToolTipText("Build in new Window");

toolbar.add(builNewWinButton);

toolbar.addSeparator();

JButton saveImgButton = new JButton(new ImageIcon("icon/saveimage.png"));

saveImgButton.setSize(32, 32);

saveImgButton.setToolTipText("Save as image");

toolbar.add(saveImgButton);

table = new JTable(17, 5);

Cursor cursor = new Cursor(2);

table.setCursor(cursor);

table.setRowSelectionAllowed(false);

table.setCellSelectionEnabled(true);

JScrollPane scrlPn = new JScrollPane(table);

scrlPn.setSize(50, 50);

buttonPanel.add(scrlPn);

JPanel areaPanel = new JPanel();

areaPanel.setBounds(0, 350, 500, 300);

areaPanel.setFocusable(true);

add(areaPanel);

textArea = new JTextArea(10, 42);

textArea.setBounds(0, 200, 500, 200);

textArea.setWrapStyleWord(true);

JScrollPane scrollPane = new JScrollPane(textArea);

areaPanel.add(scrollPane);

color = generateColors();

/\*\*

\* дія для збереження зображення

\*/

ActionListener savImg = new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

if (fl > 0) {

BufferedImage image = (BufferedImage);

if (fl == 1) {

image = (BufferedImage) dp.createImage(dp.getWidth(),

dp.getHeight());

Graphics2D g2 = image.createGraphics();

dp.paint(g2);

g2.dispose();

JFileChooser fileI = new JFileChooser();

fileI.setCurrentDirectory(new File("."));

String wayI = null;

if (fileI.showDialog(null, "Save img") == JFileChooser.APPROVE\_OPTION) {

wayI = fileI.getSelectedFile().getPath();

if (!wayI.endsWith("png")) {

wayI += ".png\\";

}

textArea.append("image saved into:" + wayI + "\n");

}

if (wayI != null) {

try {

ImageIO.write(image, "png", new File(wayI));

} catch (IOException io) {

io.printStackTrace();

}

}

}

else {

image = (BufferedImage) frame.createImage(frame

.getWidth(), frame.getHeight());

Graphics2D g2 = image.createGraphics();

frame.paint(g2);

g2.dispose();

JFileChooser fileI = new JFileChooser();

fileI.setCurrentDirectory(new File("."));

String wayI = null;

if (fileI.showDialog(null, "Save img") == JFileChooser.APPROVE\_OPTION) {

wayI = fileI.getSelectedFile().getPath();

if (!wayI.endsWith("png")) {

wayI += ".png\\";

}

textArea.append("image saved into:" + wayI + "\n");

}

if (wayI != null) {

try {

ImageIO.write(image, "png", new File(wayI));

} catch (IOException io) {

io.printStackTrace();

}

}

}

} else {

textArea.append("diagram was not created!" + "\n");

}

}

};

/\*\*

\* дія для відкриття файлу

\*/

ActionListener open = new ActionListener() {

public void actionPerformed(ActionEvent event) {

openCSV();

}

};

/\*\*

\* дія для будування діаграми у новому вікні

\*/

builNewWinButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

build(true);

}

});

/\*\*

\* дія для будування діаграми

\*/

buildButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {

build(false);

}

});

/\*\*

\*дія для збереження таблиці

\*/

ActionListener savT = new ActionListener() {

public void actionPerformed(ActionEvent event) {

STbl();

}

};

saveImgButton.addActionListener(savImg);

saveButton.addActionListener(savT);

openButton.addActionListener(open);

JMenuBar menu = new JMenuBar();

setJMenuBar(menu);

JMenu Fmenu = new JMenu("File");

menu.add(Fmenu);

JMenuItem openW = Fmenu.add("open");

openW.addActionListener(open);

Fmenu.addSeparator();

JMenuItem savImage = Fmenu.add("save img");

savImage.addActionListener(savImg);

JMenuItem saveT = Fmenu.add("save table");

saveT.addActionListener(savT);

Fmenu.addSeparator();

/\*\*

\* дія для виходу з програми

\*/

JMenuItem exitW = Fmenu.add(new AbstractAction("Exit") {

public void actionPerformed(ActionEvent even) {

System.exit(0);

}

});

/\*\*

\* створення методу для обробки таблиці

\*/

table.addKeyListener(new KeyAdapter() {

public void keyReleased(KeyEvent key) {

if (key.getKeyChar() == 10) {

if (key.isControlDown()) {

build(true);

} else {

build(false);

}

}

if (key.getKeyCode() == KeyEvent.VK\_S && key.isControlDown()) {

STbl();

}

}

public void keyPressed(KeyEvent key) {

if (key.getKeyChar() == 127) {

if (table.getSelectedColumn() != -1) {

int[] col = table.getSelectedColumns();

int[] row = table.getSelectedRows();

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

for (int j = 0; j < col.length; j++) {

table.setValueAt(null, row[i], col[j]);

}

}

}

}

}

});

}

/\*\*

\* метод для збереження таблиці

\*/

public void STbl() {

if (table.getValueAt(0, 0) != null) {

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

String value;

String valT;

int col = 0;

int row = 0;

while (table.getValueAt(0, col) != null) {

col++;

}

while (table.getValueAt(row, 0) != null) {

row++;

}

for (int i = 0; i < row; i++) {

value = "";

for (int j = 0; j < col; j++) {

valT = table.getValueAt(i, j).toString();

value += valT;

if (j < col - 1) {

value += ";";

}

}

lst.add(i, value);

}

JFileChooser fileT = new JFileChooser();

fileT.setCurrentDirectory(new File("."));

String wayT = null;

if (fileT.showDialog(null, "Save table") == JFileChooser.APPROVE\_OPTION) {

wayT = fileT.getSelectedFile().getPath() + ".csv" + "\\";

textArea.append("table saved into:" + wayT + "\n");

}

if (wayT != null) {

try {

csvp.writeIn(wayT, lst);

} catch (IOException e) {

e.printStackTrace();

}

}

} else {

textArea.append("table is empty or not enough data in the table"

+ "\n");

}

}

/\*\*

\* метод для генераціі кольорів діаграми

\*/

public Color[] generateColors() {

Random rand = new Random();

int rd = 0;

int gr = 0;

int bl = 0;

Color[] col = new Color[10];

for (int i = 0; i < 10; i++) {

rd = rand.nextInt(255);

gr = rand.nextInt(255);

bl = rand.nextInt(255);

Color colZ = new Color(rd, gr, bl);

col[i] = colZ;

}

return col;

}

public int getSelRow() {

return table.getSelectedRow();

}

public int getSelCol() {

return table.getSelectedColumn();

}

/\*\*

\* метод для відкриття csv file

\*/

public void openCSV() {

delete();

if (file == null) {

file = new JFileChooser();

file.setCurrentDirectory(new File("."));

}

if (file.showDialog(null, "Open") == JFileChooser.APPROVE\_OPTION) {

way = file.getSelectedFile().getPath() + "/";

textArea.append(file.getSelectedFile().getPath() + "\n");

}

readFile();

}

/\*\*

\* метод для зчитування з csv file

\*/

public void readFile() {

if (way != null && way.endsWith("csv/")) {

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

int cnt = 0;

while (cnt < 3) {

try {

String Fname = way;

File newcsv = new File(Fname);

if (newcsv.exists()) {

try {

mylst = csvp.readF(Fname);

cnt = 3;

}

catch (CSVParseException e) {

System.err.println(e + " ");

}

}

else {

System.out.println("this file does not exists");

cnt++;

}

} catch (IOException e) {

cnt++;

} catch (ClassNotFoundException e) {

cnt++;

}

}

CSVThread ustrm = new CSVThread(csvp);

Thread strm = new Thread(ustrm);

strm.start();

try {

strm.join();

} catch (Exception e) {

e.printStackTrace();

}

arr = ustrm.getMas();

int col = 0;

int row = 0;

while (table.getValueAt(0, col) != null) {

col++;

}

while (table.getValueAt(row, 0) != null) {

row++;

}

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

table.setValueAt(null, i, j);

}

}

for (int i = 0; i < ustrm.getJj(); i++) {

for (int j = 0; j < ustrm.getKk(); j++) {

table.setValueAt(arr[i][j], i, j);

}

}

} else {

textArea.append("Please, choose file '.CSV'" + "\n");

}

}

/\*\*

\* метод для видалення таблиці

\*/

public void delete() {

int col = 0;

int row = 0;

while (table.getValueAt(0, col) != null) {

col++;

}

while (table.getValueAt(row, 0) != null) {

row++;

}

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

table.setValueAt(null, i, j);

}

}

}

/\*\*

\* метод для будування діаграми

\*

\* @param newFrm

\*/

public void build(boolean newFrm) {

rows = table.getSelectedRows();

colum = table.getSelectedColumns();

int err = 0;

try {

int[][] arrDia = new int[rows.length][colum.length];

colName = new String[colum.length];

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

colName[i] = table.getColumnName(colum[i]);

}

if (table.getSelectedRow() != -1) {

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

for (int j = 0; j < colum.length; j++) {

if ((table.getValueAt(rows[i], colum[j]) != null)) {

arrDia[i][j] = Integer.parseInt(table.getValueAt(

rows[i], colum[j]).toString());

} else {

textArea

.append("selected area is empty or not enough data in the table"

+ "\n");

err = -1;

}

}

}

if (err != -1) {

if(dp != null){

dp.removeMouseListener(dp.getMouseListeners()[0]);

dp.removeMouseWheelListener(dp.getMouseWheelListeners()[0]);

}

dp = new DiagramPanel(arrDia, rows, colum,

colName, color, table);

dp.setBounds(500, 45, 500, 490);

if (newFrm || (rows.length \* colum.length > 30)) {

frame = new FrameDrower();

frame.add(dp);

} else {

add(dp);

fSize = new Dimension(1025, 600);

setSize(fSize.width, fSize.height);

setLocation(scrWidth / 2 - fSize.width / 2, scrHeight

/ 2 - fSize.height / 2);

dp.repaint();

}

fl = 1;

}

} else {

textArea.append("select the area" + "\n");

}

} catch (Exception e) {

textArea.append("not enough data in the table or incorrect data!"

+ "\n");

}

}

}

package diagr22;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Point;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseWheelEvent;

import java.awt.event.MouseWheelListener;

import java.awt.geom.Rectangle2D;

import java.util.ArrayList;

import java.util.Arrays;

import javax.swing.JPanel;

import javax.swing.JTable;

import javax.swing.border.LineBorder;

public class DiagramPanel extends JPanel {

private static final long serialVersionUID = 1L;

/\*\*

\* масив де зберігаються дані для будування діаграми

\*/

private int[][] arr = new int[50][50];

/\*\*

\* кількість рядків

\*/

private int[] nrows;

/\*\*

\* кількість стовпчиків

\*/

private int[] ncolum;

/\*\*

\* graphics2d

\*/

private Graphics2D g2;

/\*\*

\* масив де зберігається інформація про прямокутники

\*/

private ArrayList<Rectangle2D> squars;

/\*\*

\* масив де зберігаються назви стовпчиків

\*/

private String[] colName;

/\*\*

\* масив кольорів

\*/

private Color[] color;

/\*\*

\* значення CalcRtects

\*/

private CalcRtects rctg;

private int selCol = -1;

private JTable table;

/\*\*

\*

\* @param array

\* масив даних

\* @param nrows

\* кількість рядків

\* @param ncolum

\* кількість стовпчиків

\* @param colNam

\* назва стовпчику

\* @param col

\* колір прямокутнику

\*/

public DiagramPanel(int[][] array, int[] nrows, int[] ncolum, String[] colNam,

Color[] col, JTable table) {

this.colName = colNam;

this.arr = array;

this.nrows = nrows;

this.ncolum = ncolum;

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

this.color = col;

this.table = table;

this.addMouseListener(mouseSelect);

this.addMouseWheelListener(mouseResizer);

}

MouseListener mouseSelect = new MouseListener() {

@Override

public void mouseReleased(MouseEvent event) {

selCol = -1;

repaint();

}

@Override

public void mousePressed(MouseEvent event) {

Point p = event.getPoint();

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

if(p.getX()>=squars.get(i).getX() && p.getX()<=squars.get(i).getMaxX() && p.getY()>=squars.get(i).getY() && p.getY()<=squars.get(i).getMaxY()){

selCol = i;

repaint();

}

}

}

@Override

public void mouseExited(MouseEvent event) {}

@Override

public void mouseEntered(MouseEvent event) {}

@Override

public void mouseClicked(MouseEvent event) {}

};

MouseWheelListener mouseResizer = new MouseWheelListener() {

@Override

public void mouseWheelMoved(MouseWheelEvent event) {

if(selCol !=-1){

int x = selCol/ncolum.length;

int y = selCol%ncolum.length;

int next = arr[x][y] - event.getWheelRotation();

if(next>=0){

arr[x][y] = next;

repaint();

table.setValueAt(next, nrows[x], ncolum[y]);

}

}

}

};

/\*\*

\* метод для створення графіки

\*/

public void paintComponent(Graphics g) {

g2 = (Graphics2D) g;

g2.setColor(Color.white);

g2.fillRect(0, 0, getSize().width, getSize().height);

rctg = new CalcRtects(arr, this.nrows.length, this.ncolum.length, getSize());

squars = rctg.getRects();

int wRt = (getSize().width - 35 - 15 \* nrows.length) / (nrows.length \* ncolum.length);

int wd = 35;

int coo = rctg.getMax();

while (coo % 10 != 0) {

coo++;

}

int vrbl = (int) (coo \* rctg.getScale() / 10);

int coord = getSize().height - 30;

int lbl = 0;

Font fnt = new Font("Arial", 1, 12);

g2.setFont(fnt);

for (int i = 0; i < 11; i++) {

g2.setColor(Color.pink);

g2.drawLine(25, coord, getSize().width - 10, coord);

g2.setColor(Color.BLACK);

if ((coord + 5 - 12) > 0) {

g2.drawString("" + lbl, 2, coord + 5);

lbl += coo / 10;

}

coord -= vrbl;

}

int fontSize = wRt - 4;

if (rctg.getMax() > 100) {

fontSize /= 2;

}

int numb = 0;

for (int j = 0; j < nrows.length; j++) {

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

g2.setColor(color[i]);

Rectangle2D rct = squars.get(numb);

if(numb == selCol){

g2.draw(rct);

}else{

g2.fill(rct);

}

g2.setColor(Color.black);

g2.drawLine((int) rct.getX(), (int) rct.getY(), (int) rct

.getX(), (int) rct.getY() + (int) rct.getHeight());

g2.drawLine((int) rct.getX(), (int) rct.getY(), (int) rct

.getX()

+ (int) rct.getWidth(), (int) rct.getY());

g2.drawLine((int) rct.getX() + (int) rct.getWidth(), (int) rct

.getY(), (int) rct.getX() + (int) rct.getWidth(),

(int) rct.getY() + (int) rct.getHeight());

if (fontSize <= 30 && fontSize >= 8) {

Font font = new Font("Arial Bold", 3, fontSize);

g2.setFont(font);

int znach = (int) (rct.getY() - 1);

if ((znach) > wRt - 4) {

g2.drawString("" + arr[j][i], wd, znach - 2);

}

}

wd += wRt;

numb++;

}

wd += 15;

}

/\*\*

\* лінія навколо прямокутника

\*/

g2.setColor(Color.red);

g2.drawLine(25, 5, 25, getSize().height - 30);

g2.drawLine(25, getSize().height - 30, getSize().width - 10,

getSize().height - 30);

int shft = (int) (1.0 \* getSize().width / ncolum.length - 30);

int indent = 25;

Font ff = new Font("Arial", 1, 12);

g2.setFont(ff);

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

g2.setColor(color[i]);

g2.fillRect(indent + 9, getSize().height - 15, 20, 10);

g2.setColor(Color.black);

g2.drawString(colName[i], indent + 15, getSize().height - 5);

indent += shft;

}

}

}

ДОДАТОК Б. СТРУКТУРА ПРОЕКТУ

