Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА

АКАДЕМІЯ»

ПРАКТИКА НАВЧАЛЬНА

Звіт з практичного завдання «Графік функції»

Підготував студент

1-го року навчання

Факультету інформатики

Спеціальність «Інженерія програмного забезпечення»

3 група

Чорнокозинський Кирило

Сергійович

Викладач

Олена Миколаївна

Пєчкурова

Contents

[1. Постановка задачі 3](#_Toc37791)

[2. Структура програми, опис методів та класів 3](#_Toc37792)

[3. Інструкція для користувача 7](#_Toc37793)

[4. Скріншоти 16](#_Toc37794)

[5. Труднощі у виконанні завдання 19](#_Toc37795)

[6. Висновки 19](#_Toc37796)

[7. Код 20](#_Toc37797)

# Постановка задачі

Створити (мінімум 15 дій)

* інженерний калькулятор,
* або графічний редактор,
* або спрощену версію текстового редактора.

Обов'язкові функції:

1. Меню
2. Відкриття файлу
3. Збереження файлу
4. Скрол тексту, зображення 5) Збільшення та зменшення шрифту

Додатково:

1. Форматування тексту
2. Вставка фото, тексту, …

# Структура програми, опис методів та класів

Для створення графічного інтерфейсу була використана бібліотека javafx.

Програма складається з трьох публічних класів:

1.AppController

2.Util

3.AppLaunch

* Поля класу AppController:

|  |  |
| --- | --- |
|  | public static final String DEFAULT\_ROOT =  "C:\\Users\\CHIRNY\\Desktop\\";  @FXML  public TextArea infoText; |
| @FXML  private ToggleButton italicButton;  @FXML  private TextField lookingForField; private Stage fontStage;  @FXML  private Spinner<Double> fontSize;  @FXML  private ComboBox<String> fontChooser;  @FXML  private TextArea fontExample;  @FXML  private CheckMenuItem statusBar;  @FXML  private TextArea textArea;  @FXML  private AnchorPane anchorPane;  @FXML  private CheckMenuItem wordWrap;  @FXML  private Label cursorInfo; private int savedFileCounter; private String currentFileName = ""; private boolean textIsUpdated; private Stage stage; private Stage checkStage; private Stage replaceStage; private Scene scene; private boolean openingMode; private boolean creationMode;  @FXML  private TextField from;  @FXML  private TextField to;  @FXML  private ToggleButton boldButton; private boolean isBold; private boolean isItalic; private boolean i; | |

* Методи класу AppController:

|  |  |
| --- | --- |
|  | start () |

Створює головне вікно на основі файла sample.fxml.

public static void createEditor(double step, double xMin, double xMax, double yMin, double yMax)

Створює вікно

startCheckDialog()

Відкриває вікно, в якому пропунується хберегти файл.

startAboutDialog()

Відкриває вікно, в якому ви можете дізнатися інформацію.

startFontDialog()

Відкриває вікно для вибору шрифту.

public static void main(String[] args)

Запускає програму.

Інші методи цього класу менш важливі, а саме: сортування точок по іксам та ігрикам та перевірка стрічки на число.

* Для кнопки “Draw” - зчитуються дані з текстових полів та викликаться метод draw(), з якого розпочинаються подальші дії щодо створення графіку та виводу його на екран.
* Для кнопки “Save” – викликається метод save()(створюється файл формату .png, зі збереженою картинкою графіку).
* Поля класу Controller:

|  |  |
| --- | --- |
|  | @FXML  private Label sliderLabel;  @FXML  private Slider slider;  @FXML  private TextField xMin;  @FXML  private TextField xMax;  @FXML  private TextField yMin;  @FXML  private TextField yMax;  @FXML  private Label labelException1;  @FXML  private Label labelException2;  @FXML  private Label labelSaveException;    private static int counter = 0; private static int saveCounter = 0; |

* Методи класу Controller:

|  |  |
| --- | --- |
|  | public void draw(ActionEvent event) |

Запускає .

Private void drawGraphic()

Зчитує те, що було введено в комірки діапазону та кроку.

public void changeLabel()

Пов’язує лейбл зі слайдером.

public void save(ActionEvent event)

Зберігає скріншот графіка.

# Інструкція для користувача

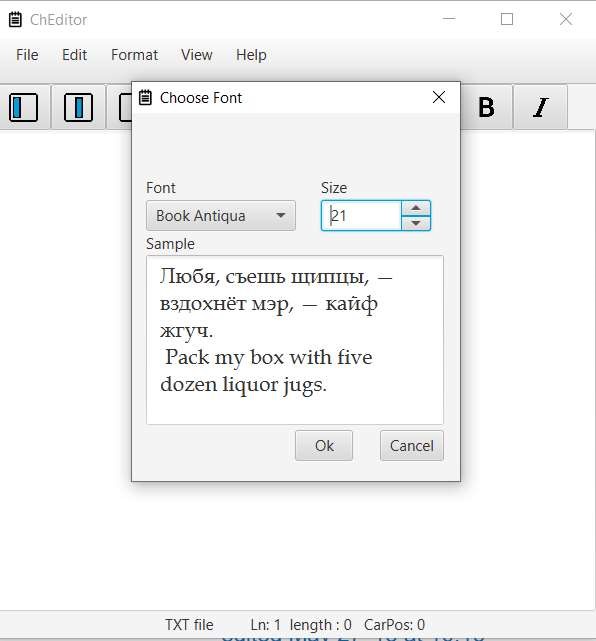
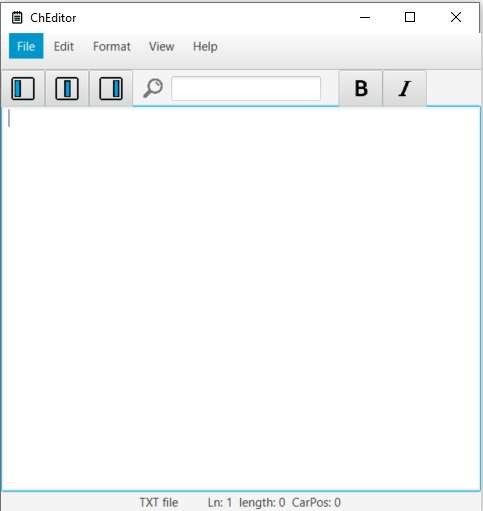
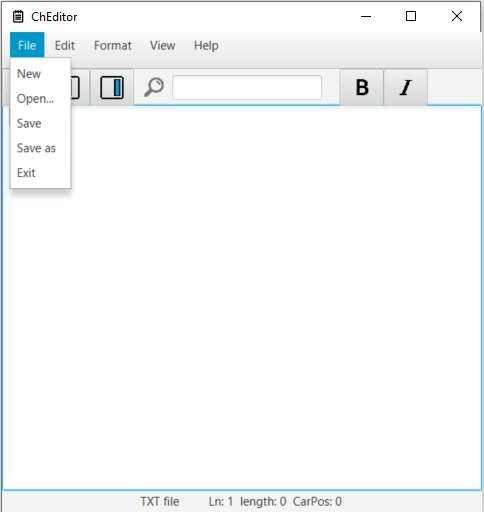


Рис. 1 - початкове вікно програми, яке зустрічає користувача



Це вікно зустрічає вас при запуску програми.



Відкрити файл

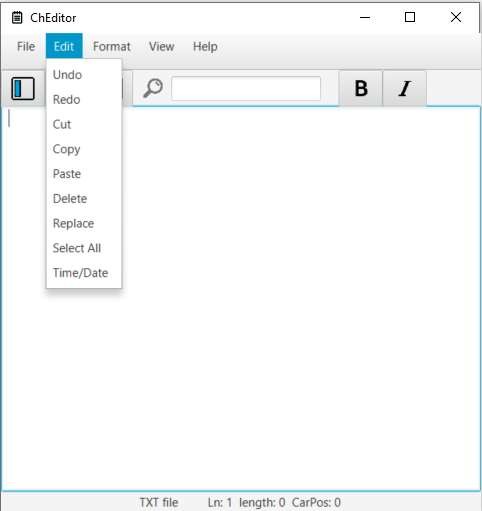
Засейвити файл в

дефолтній директорії

Засейвити файл за допомогою

туди куди хоче користувач

Закрити програму



Зробити крок назад

Зробити крок вперед

Вирізати текст

Скопіювати текст

Вставити текст

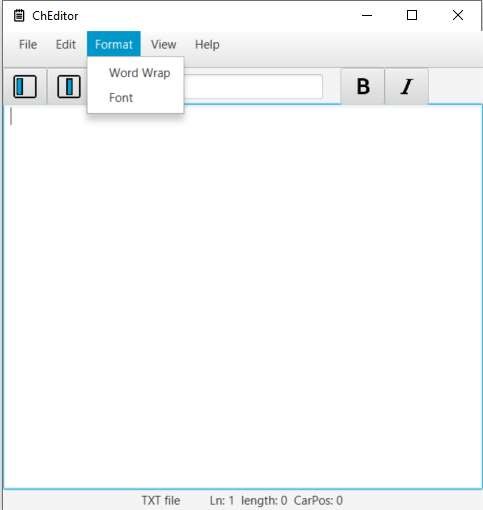
Видалити текст

Замінити стріч

ку в файлі на іншу підстрічку

Виділити весь текст в файлі

Вставити поточну дату і час у кінець файлу

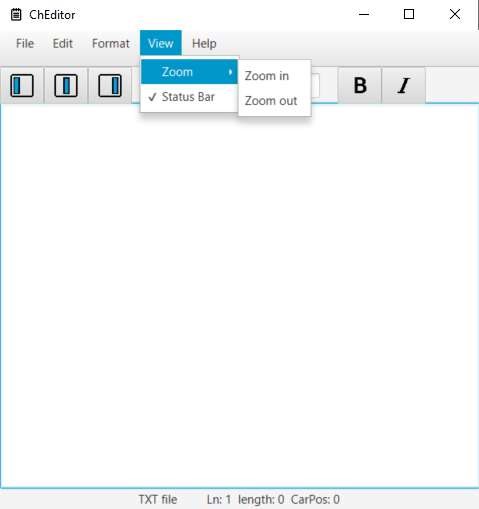


П

еренос

тексту

Зміна Шрифту



Status Bar

П

оказувати

\

не показ

у

вати

статус бар

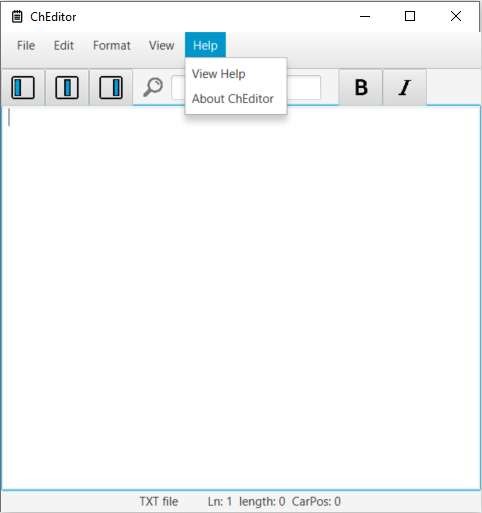
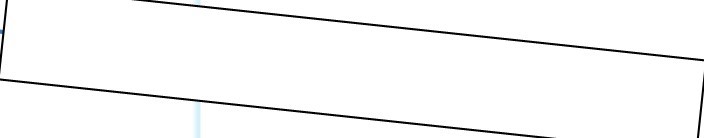
З

б

ільш

ити

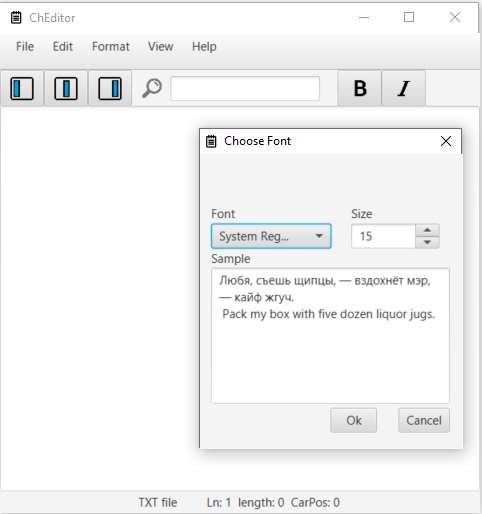
шрифт на 5 одиниць



Про програму

Та про його розробника

Силка на звіт



Вікно що

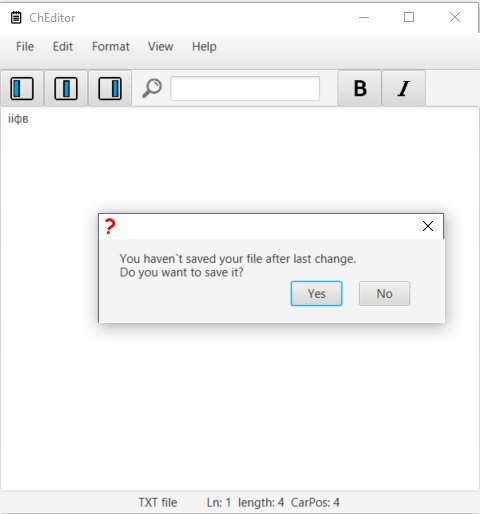
відкривається при

вибиранні шрифту

Тут можна обрати шрифт

Тут можна обрати розмір шрифта

Приклад шрифта



Вікно перевірки, щодо

збереження файлу

Зберегти і вийти

Не зберігати і вийти

Вирівнювання тексту

(

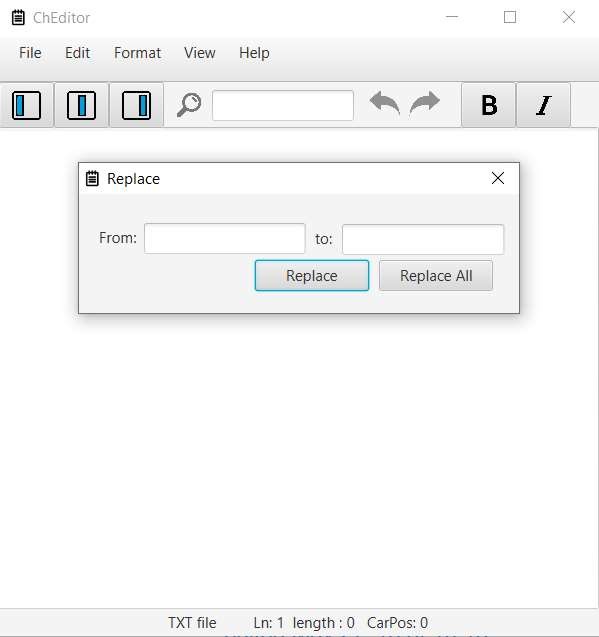
ліво, центр, право)

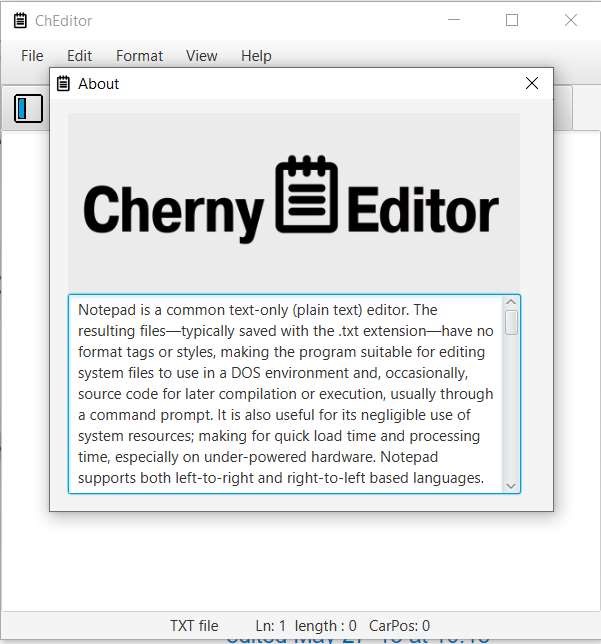
Зробити шрифт італіком

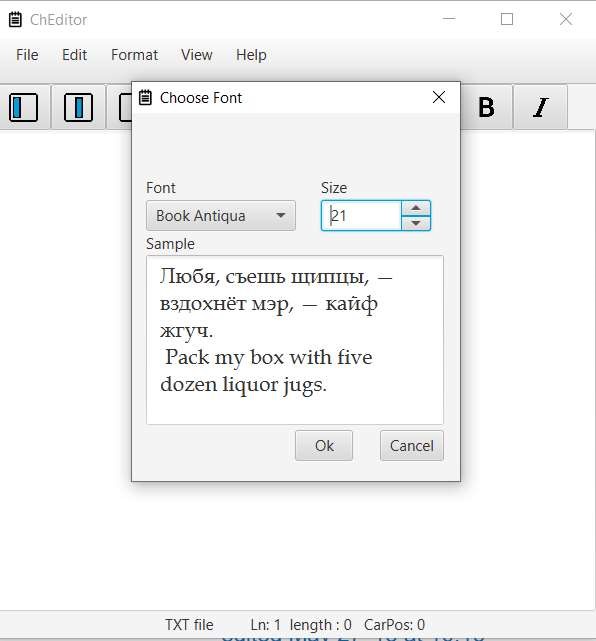
Зробити шрифт жирним

Пошук файлі підстроки

# Скріншоти







# Труднощі у виконанні завдання

Виконуючи завдання в мене були проблеми з слухачами, тому що в деяких компонентах є певні лістенери а в інших нема.

+ багато методів для текстереа приходилося писати деякі методи самому, тому що їч не було, або було не зручно іх заюзати.

# Висновки

Виконання цього завдання дало мені змогу поглибити свої знання Java, а саме використання графічного інтерфейсу (javafx), збереження в файли, тренувало мої здібності логічного мислення.

# Код

|  |
| --- |
| 8. package cheditor; import javafx.beans.value.ChangeListener; import javafx.beans.value.ObservableValue; import javafx.css.PseudoClass; import javafx.event.ActionEvent; import javafx.fxml.FXML; import javafx.fxml.FXMLLoader; import javafx.scene.Parent; import javafx.scene.Scene; import javafx.scene.control.Label; import javafx.scene.control.TextArea; import javafx.scene.control.TextField; import javafx.scene.control.\*; import javafx.scene.image.Image; import javafx.scene.layout.AnchorPane; import javafx.scene.text.Font; import javafx.scene.text.FontPosture; import javafx.scene.text.FontWeight; import javafx.scene.web.HTMLEditor; import javafx.stage.Window; import javafx.stage.\*;  import java.awt.\*; import java.io.File; import java.io.IOException; import java.io.PrintWriter; import java.text.DateFormat; import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date; import java.util.regex.Matcher; import java.util.regex.Pattern;  import java.util.regex.PatternSyntaxException;    public class AppController {  public static final String DEFAULT\_ROOT =  "C:\\Users\\CHIRNY\\Desktop\\";  @FXML  public TextArea infoText;  @FXML  private ToggleButton italicButton;  @FXML  private TextField lookingForField; private Stage fontStage;  @FXML  private Spinner<Double> fontSize;  @FXML  private ComboBox<String> fontChooser;  @FXML  private TextArea fontExample; @FXML |

private CheckMenuItem statusBar;

@FXML private TextArea textArea;

@FXML

|  |
| --- |
| private AnchorPane anchorPane;  @FXML  private CheckMenuItem wordWrap;  @FXML  private Label cursorInfo; private int savedFileCounter; private String currentFileName = ""; private boolean textIsUpdated; private Stage stage; private Stage checkStage; private Stage replaceStage; private Scene scene; private boolean openingMode; private boolean creationMode;  @FXML  private TextField from;  @FXML  private TextField to;  @FXML  private ToggleButton boldButton; private boolean isBold; private boolean isItalic; private boolean aBoolean; private HTMLEditor result; private int findNumber;    public AppController() {  }  public AnchorPane getPane() { return anchorPane;  }  public TextArea getTextArea() { return textArea;  }  public void setScene(Scene scene) { this.scene = scene;  } public void setCheckStage(Stage stage) { checkStage = stage;  }    @FXML  public void onNew() { creationMode = true; if (textIsUpdated) startCheckDialog(); else { clear();  creationMode = false;  }  } private void clear() { textArea.clear(); currentFileName = ""; |

}

@FXML

public void onOpen() throws IOException {

|  |
| --- |
| openingMode = true; if (textIsUpdated) { startCheckDialog();  } else {  FileChooser fileChooser = new FileChooser(); fileChooser.setTitle("chooser");  FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("TXT files (\*.txt)", "\*.txt"); fileChooser.getExtensionFilters().add(extFilter);  Stage fileChooserStage = new Stage();  //Show save file dialog File selectedFile =  fileChooser.showOpenDialog(fileChooserStage); if (selectedFile != null) {  textArea.setText(Util.readFile(selectedFile.getPath()));  System.out.println(Util.readFile(selectedFile.getPath())); currentFileName = selectedFile.getPath(); onSave();  }  }  }    @FXML  public void onExit() {    System.out.println(textIsUpdated); if (textIsUpdated) startCheckDialog();  else if (!creationMode && !openingMode)  System.exit(0);      }    public void setBinding() {  textArea.setPrefRowCount(1); textArea.setPrefRowCount(1);  lookingForField.textProperty().addListener(new  ChangeListener<String>() { @Override  public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) { find();  }  });    textArea.textProperty().addListener(new ChangeListener<String>() {  @Override  public void changed(final ObservableValue<? extends String> observable, final String oldValue, final String newValue) { textIsUpdated = true; find();  } });  textArea.prefWidthProperty().bind(anchorPane.widthProperty()); |

stage.getScene().getWindow().addEventFilter(WindowEvent.WINDOW\_CLOSE\_REQ

|  |
| --- |
| UEST, this::closeWindowEvent);  Window window = stage // Get the primary stage from your  Application class  .getScene()  .getWindow();    window.fireEvent(new WindowEvent(window,  WindowEvent.WINDOW\_CLOSE\_REQUEST));    } private void closeWindowEvent(WindowEvent event) { System.out.println("Window close request ..."); if (aBoolean) { if (textIsUpdated) { startCheckDialog(); event.consume(); } else System.exit(0);    }  }      @FXML  public void showLocation() { aBoolean = true;  //TXT file Ln: 1 length : 0 CarPos: 0  cursorInfo.setText("TXT file Ln: " + getNumberLines() +  " length: " + textArea.getLength() + " CarPos: " + textArea.getCaretPosition());  }    @FXML  public void onSaveAs() {  FileChooser fileChooser = new FileChooser();  //Set extension filter for text files  FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("TXT files (\*.txt)", "\*.txt"); fileChooser.getExtensionFilters().add(extFilter);  Stage fileChooserStage = new Stage(); fileChooser.setTitle("chooser");  //Show save file dialog  File file = fileChooser.showSaveDialog(fileChooserStage); if (file != null) {  if (Util.saveTextToFile(textArea.getText(), file)) textIsUpdated = false;  }  }    @FXML  public void onSave() throws IOException { if (currentFileName.isEmpty()) { savedFileCounter++;  PrintWriter out = new PrintWriter(DEFAULT\_ROOT + "New ChDocument (" + savedFileCounter + ").txt"); out.println(textArea.getText()); out.close();  } else {  textIsUpdated = false;  File file = new File(currentFileName); |

Util.saveTextToFile(textArea.getText(), file);

|  |
| --- |
| }      }      @FXML  public void onWordWrap() { if (wordWrap.isSelected()) { textArea.setWrapText(true); } else textArea.setWrapText(false);  }  public void setStage(Stage stage) { this.stage = stage;  }  public void onYes() throws IOException { if (currentFileName.equals("")) onSaveAs(); else onSave();  if (!creationMode && !openingMode) System.exit(0); checkStage.close(); if (creationMode) { onNew();  creationMode = false; } else if (openingMode) { onOpen();  openingMode = false;  } else System.exit(0);    }    public void onNo() throws IOException { if (creationMode) {  creationMode = false; textIsUpdated = false; clear();  checkStage.close(); } else if (openingMode) { openingMode = false; textIsUpdated = false; onOpen(); checkStage.close();  } else System.exit(0);  }    public void doCenterAlignment(ActionEvent event) {  PseudoClass center = PseudoClass.getPseudoClass("center");  PseudoClass left = PseudoClass.getPseudoClass("left"); PseudoClass right = PseudoClass.getPseudoClass("right"); textArea.pseudoClassStateChanged(right, false); textArea.pseudoClassStateChanged(left, false); textArea.pseudoClassStateChanged(center, true); }    public void doLeftAlignment(ActionEvent event) {  PseudoClass center = PseudoClass.getPseudoClass("center");  PseudoClass left = PseudoClass.getPseudoClass("left"); |

PseudoClass right = PseudoClass.getPseudoClass("right");

|  |
| --- |
| textArea.pseudoClassStateChanged(left, true); textArea.pseudoClassStateChanged(center, false); textArea.pseudoClassStateChanged(right, false);  }  public void doRightAlignment(ActionEvent event) {  PseudoClass center = PseudoClass.getPseudoClass("center"); textArea.pseudoClassStateChanged(center, false); PseudoClass left = PseudoClass.getPseudoClass("left"); PseudoClass right = PseudoClass.getPseudoClass("right"); textArea.pseudoClassStateChanged(left, false); textArea.pseudoClassStateChanged(right, true); }    @FXML  public void onStatusBar() { if (statusBar.isSelected()) { cursorInfo.setOpacity(100); } else cursorInfo.setOpacity(0);  }      @FXML  public void onZoomIn() {  Font font = textArea.getFont();  textArea.setFont(new Font(font.getStyle(), font.getSize() + 5));  }    @FXML  public void onZoomOut() {  Font font = textArea.getFont();  textArea.setFont(new Font(font.getStyle(), font.getSize() - 5)); }  public void startReplaceDialog() {  Parent parent = null;  try {  FXMLLoader fxmlLoader = new FXMLLoader(AppLaunch.class  .getResource("/ReplaceDialog.fxml")); fxmlLoader.setController(this); parent = fxmlLoader.load(); } catch (IOException e) {  e.printStackTrace(); }  replaceStage = new Stage();  replaceStage.initModality(Modality.WINDOW\_MODAL); replaceStage.initOwner(stage); Scene scene = new Scene(parent); replaceStage.getIcons().add(new  Image(AppLaunch.class.getResourceAsStream("/icon.png"))); replaceStage.setTitle("Replace"); replaceStage.setScene(scene); replaceStage.setResizable(false); setCheckStage(replaceStage); replaceStage.show();  }    public void startCheckDialog() { Parent parent = null; try { |

FXMLLoader fxmlLoader = new FXMLLoader(AppLaunch.class

|  |
| --- |
| .getResource("/СheckDialog.fxml")); fxmlLoader.setController(this); parent = fxmlLoader.load(); } catch (IOException e) {  e.printStackTrace();  }  Stage checkStage = new Stage();  checkStage.initModality(Modality.WINDOW\_MODAL);  checkStage.initOwner(stage); Scene scene = new Scene(parent); checkStage.getIcons().add(new  Image(AppLaunch.class.getResourceAsStream("/question-mark.png"))); checkStage.setTitle(""); checkStage.setScene(scene); checkStage.setResizable(false); setCheckStage(checkStage); checkStage.show();  } public void onCut() { textArea.cut();  } public void onPaste() { textArea.paste();  } public void onCopy() { textArea.copy();  } public void onSelectAll() { textArea.selectAll();  } public void onUndo() { textArea.undo();  } public void onRedo() { textArea.redo();  }  public void onDate() {  DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd  HH:mm:ss");  Date date = new Date();  String newContent = textArea.getText(); newContent += dateFormat.format(date); textArea.setText(newContent); }  public void onReplace() { startReplaceDialog();  }  public void onReplaceFirst() {  String content = textArea.getText();  content = content.replaceFirst(from.getText(), to.getText()); textArea.setText(content); replaceStage.close();  } |

public void onReplaceAll() { String content = textArea.getText();

content = content.replaceAll(from.getText(), to.getText());

|  |
| --- |
| textArea.setText(content); replaceStage.close();  }  public void onDelete() {  textArea.deleteText(textArea.getSelection()); }  public void startAboutDialog() { Parent parent = null;  try {  FXMLLoader fxmlLoader = new FXMLLoader(AppLaunch.class  .getResource("/AboutDialog.fxml")); fxmlLoader.setController(this); parent = fxmlLoader.load(); } catch (IOException e) {  e.printStackTrace();  }  Stage AboutStage = new Stage();  AboutStage.initModality(Modality.WINDOW\_MODAL);  AboutStage.initOwner(stage);  Scene scene = new Scene(parent);  AboutStage.getIcons().add(new  Image(AppLaunch.class.getResourceAsStream("/icon.png")));  AboutStage.setTitle("About");  AboutStage.setScene(scene); AboutStage.setResizable(false); setCheckStage(AboutStage); infoText.setWrapText(true);  AboutStage.show();  } public void startFontDialog() { Parent parent = null;  try {  FXMLLoader fxmlLoader = new FXMLLoader(AppLaunch.class  .getResource("/FontDialog.fxml")); fxmlLoader.setController(this); parent = fxmlLoader.load(); } catch (IOException e) {  e.printStackTrace();  }  Stage fontStage = new Stage(); this.fontStage = fontStage;  fontStage.initModality(Modality.WINDOW\_MODAL); fontStage.initOwner(stage); Scene scene = new Scene(parent); fontStage.getIcons().add(new  Image(AppLaunch.class.getResourceAsStream("/icon.png"))); fontStage.setTitle("Choose Font"); setComboBox(); setSpinner();  fontStage.setScene(scene); fontStage.setResizable(false); setCheckStage(fontStage); fontStage.show();  fontExample.setFont(textArea.getFont());  }  private void setComboBox() {  for (String fontName : Util.getFonts()) { |

fontChooser.getItems().add(fontName);

|  |
| --- |
| }  fontChooser.setPromptText(textArea.getFont().getName()); } private void setSpinner() { fontSize.setValueFactory(new  SpinnerValueFactory.DoubleSpinnerValueFactory(0.0, 50, textArea.getFont().getSize()));      }    @FXML  public void setFontExample() {  fontChooser.setPromptText(fontExample.getFont().getStyle()); fontExample.setFont(new Font(fontChooser.getValue(), fontSize.getValue()));  }    @FXML  public void setFont() {  textArea.setFont(fontExample.getFont()); fontStage.close();    }    @FXML  public void onClose() { fontStage.close();    }    @FXML  public void find() { findNumber = 0; try {  matchRegExp();  } catch (InterruptedException e) { e.printStackTrace();  }  }    @FXML  public void setBold() {  if (boldButton.isSelected()) { isBold = true; if (isItalic)    textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.BOLD, FontPosture.ITALIC, textArea.getFont().getSize())); else textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.BOLD, textArea.getFont().getSize()));  } else {  if (isItalic)    textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.NORMAL, FontPosture.ITALIC, textArea.getFont().getSize())); else  textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.NORMAL, textArea.getFont().getSize())); |

isBold = false;

}

}

|  |
| --- |
| @FXML  public void setItalic() {    if (italicButton.isSelected()) { isItalic = true; if (isBold)    textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.BOLD, FontPosture.ITALIC, textArea.getFont().getSize())); else  textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.NORMAL, FontPosture.ITALIC, textArea.getFont().getSize()));  } else { if (isBold)    textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.BOLD, FontPosture.REGULAR, textArea.getFont().getSize())); else  textArea.setFont(Font.font(textArea.getFont().getFamily(),  FontWeight.NORMAL, FontPosture.REGULAR, textArea.getFont().getSize())); isItalic = false;  }  }    @FXML  public void onHelp() {  if (Desktop.isDesktopSupported()) { try {  File myFile = new  File("C:\\Users\\CHIRNY\\Desktop\\Plan\\Vova\\VovanPlan.pdf");  Desktop.getDesktop().open(myFile);  } catch (IOException ex) {  // no application registered for PDFs  }  }  }  public void nextFindWord() { findNumber++; try {  matchRegExp();  } catch (InterruptedException e) { e.printStackTrace();  }  } public void previousFindWord() { findNumber--; try {  matchRegExp();  } catch (InterruptedException e) { e.printStackTrace();  }  }    public void matchRegExp() throws InterruptedException { |

result = new HTMLEditor(); Pattern pattern = null; try {

pattern = Pattern.compile(lookingForField.getText());

|  |
| --- |
| } catch (PatternSyntaxException psex) { result.setHtmlText(String.format(  "<html><body><p>" +  "<span style=\"color:Red;\">%s</span>" +  "</p></body></html>", psex.getMessage())); return;  }  Matcher matcher = pattern.matcher(textArea.getText());    boolean found = false;  StringBuilder sb = new StringBuilder();    class Tuple {  final int start, end;    Tuple(int start, int end) { this.start = start; this.end = end;  }    @Override  public String toString() {  return "Tuple{start=" + start + ", end=" + end + '}';  }  }      ArrayList<Tuple> positions = new ArrayList<>();    while (matcher.find()) {  positions.add(new Tuple(matcher.start(), matcher.end()));  }  if (!positions.isEmpty()) {  if (findNumber < 0) findNumber = 0;  if (findNumber >= positions.size()) findNumber = positions.size() - 1;    System.out.println(positions.get(findNumber)); textArea.selectRange(positions.get(findNumber).start, positions.get(findNumber).end);  } else textArea.deselect();  }      private int getNumberLines(){ String s = textArea.getText(); s = s.replaceAll("\n","\n" + " ");  System.out.println(s); return s.split("\n").length;  }    }  9. package cheditor;  import javafx.application.Application; |

import javafx.fxml.FXMLLoader; import javafx.scene.Scene; import javafx.scene.image.Image; import javafx.scene.layout.AnchorPane;

|  |
| --- |
| import javafx.stage.Stage;    public class AppLaunch extends Application {    @Override  public void start(Stage primaryStage) throws Exception {  primaryStage.setMinWidth(350);  FXMLLoader fxmlLoader = new  FXMLLoader(AppLaunch.class.getResource("/MainStage.fxml"));  AnchorPane root = fxmlLoader.load(); primaryStage.setTitle("ChEditor"); primaryStage.getIcons().add(new  Image(AppLaunch.class.getResourceAsStream("/icon.png")));  Scene scene = new Scene(root, 600, 600); primaryStage.setScene(scene);  scene.getStylesheets().add("app.css");  AppController controller = fxmlLoader.getController(); controller.setStage(primaryStage); controller.setBinding(); controller.setScene(scene); primaryStage.show();      }  public static void main(String[] args) {    launch(args);  }    }      10. package cheditor;  import javafx.scene.text.Font;    import java.io.\*; import java.util.HashMap; import java.util.Map;  public class Util {    public static boolean saveTextToFile(String content, File file) { try {  PrintWriter writer;  writer = new PrintWriter(file); writer.println(content); writer.close(); return true;  } catch (IOException ex) { return false;  }  }  public static String readFile(String path) throws IOException {  BufferedReader br = new BufferedReader(new FileReader(path));  String st = ""; |

|  |  |
| --- | --- |
| String k;  while ((k = br.readLine()) != null) st += k + "\n"; return st;  }    public static String[] getFonts() {  String[] fonts = new String[]{"Arial", "Arial Black", "Arial  Narrow", "Arial Unicode MS", "Book Antiqua", "Bookman Old Style",  "Calibri", "Cambria", "Candara", "Century", "Century Gothic", "Comic  Sans MS", "Consolas", "Constantia", "Corbel", "Courier New", "Franklin  Gothic Medium", "Garamond", "Georgia", "Impact", "Lucida Console",  "Lucida Sans Unicode", "Microsoft Sans Serif", "Mistral", "Monotype  Corsiva", "Palatino Linotype", "Segoe Print", "Segoe Script", "Segoe UI", "Sylfaen", "Tahoma", "Times New Roman", "Trebuchet MS", "Verdana"}; return fonts;  }  } | |
|  |  |