# 

# **Secure IDE (IDE למפתחי נוזקה):**

**תיק פרויקט**

**אדם ליבנה 209621325**

**גרסה 2.0**

**תיכון הראל, מבשרת ציון**

**הנדסת תוכנה 883589**

**חנן רוזנטל**

**14/10/2019**

תוכן עניינים

[**Secure IDE (IDE למפתחי נוזקה): 1**](#_Toc36663945)

[**תוכן עניינים 2**](#_Toc36663946)

[**מבוא 3**](#_Toc36663947)

[**מבנה / ארכיטקטורה של הפרויקט 4**](#_Toc36663948)

[**תרשים UML 5**](#_Toc36663949)

[**מדריך למשתמש 6**](#_Toc36663950)

[**הוראות התקנה 6**](#_Toc36663951)

[**בדיקות תקינות 7**](#_Toc36663952)

[**מדריך למפתח 8**](#_Toc36663953)

[**Use cases 8**](#_Toc36663954)

[**קבצי טקסט 9**](#_Toc36663955)

[**תיקיות 9**](#_Toc36663956)

[**קבצי הרצה 9**](#_Toc36663957)

[**StartSandboxed.exe 10**](#_Toc36663958)

[**SecureIDE.exe 11**](#_Toc36663959)

[**ProjectMain.java 11**](#_Toc36663960)

[**MenuAction.java 13**](#_Toc36663961)

[**Gui2.java 14**](#_Toc36663962)

[**Function.java 24**](#_Toc36663963)

[**ErrorMessagePanel.java 34**](#_Toc36663964)

[**ColorMode.java 36**](#_Toc36663965)

[**סיכום אישי / רפלקציה 37**](#_Toc36663966)

[**ביבליוגרפיה 37**](#_Toc36663967)

[**נספחים 39**](#_Toc36663968)

מבוא

בחלק יכלול את המידע הבא:

* תיאור תכולת הספ ר
* הרקע לפרויק ט
* תהליך המחקר, יתכן שהמחקר המקדים היה עבודת שט ח o סקירת המצב הקיי ם בשו ק o אילו חידושים יש בפרויק ט
* סקירת ספרו ת )במידה והמחקר המקדים כלל ספרות מקצועית( .
* אתגרים מרכזיים
  + הבעיה איתה התמודד התלמיד o הסיבות לבחירת הנושא o מוטיבציה לעבוד ה
  + על איזה צורך הפרויקט עונה? איזה פתרון הפרויקט הזה בא לתת?
* הצגת פתרונות לבעיה )הפתרונות שנבחנו במסגרת המחקר המקדים(
* חלק זה יהיה לפחות עמוד מלא.

ספר זה הוא תיק הפרויקט הסופי שלי במגמת סייבר. בספר זה מוצג המוצר הסופי אותו פיתחתי במהלך עבודתי. את הפרויקט התחלתי בתחילת כיתה י"ב. המחקר המקדים התמקד בעיקר בבדיקה ששל האפשרויות אשר כבר קיימות בשוק. בהמשך המחקר שלי בחנתי את הטכנולוגיות בהן עבדתי ובדקתי באילו דרכים ניתן ליצור את התוכנית שרציתי. החידוש בפרויקט זה הוא שילובן של שתי טכנולוגיות, IDE ו Sandbox, והקניית תמיכה ראשונית לשילוב זה. האתגרים המרכזיים בפרויקט היו יצירת ה IDE, הפעלה של הקומפיילר של ג'אווה, ושילוב של ה IDE ביחד עם ה Sandbox. בחרתי בנושא זה מכיוון שרציתי להתחיל לעסוק בתחום של אבטחת המידע, אך לשם כך ישנם כלים אשר צריך לרכוש (כגון IDE, מערכת של Sandbox, ומאגר של מידע). אמנם יכולתי להשתמש בכלים קיימים, אך בחרתי לבנות לעצמי כלים שכאלה שכן עצם זה שבניתי אותם בעצמי מאפשר לי לשנות ולהתאים אותם לצרכיי, וגם דבר זה נותן לי הבנה מעמיקה של הדרך בה אותם כלים עובדים. בנוסף, מכיוון שבכוונתי להשתמש באותם כלים אשר פיתחתי אז הייתה לי מוטיבציה רבה יותר לעבוד על הפרויקט. כאמור, התוכנית שלי באה לענות על צורכם של אלו המעוניינים להתחיל לעסוק בתחום אבטחת המידע, בעזרת מתן כלים מגוונים אשר כולם מרוכזים תחת תוכנית אחת. ריכוזם של כל הכלים במקום אחד מאפשר התחלה יותר קלה עבור המשתמש ומייעלת את תהליך הלמידה.

מבנה / ארכיטקטורה של הפרויקט

הפרק יכלול התייחסות לנושאים הבאים:

* הצגת הפתרון המוצע והסיבות לבחירת ו
* הארכיטקטורה של הפתרון המוצע בפורמט של Top-Down level Design
* חלק זה אמור להיות הסבר מהכלל את הפרט החל ממבט על כל יחידות) Component( / מחלקות הפרויק ט
* מה תפקידה של כל יחידה o מה הקלטים שלה o מה הפלטים שלה
* הבהרה קלט יכול להיות מהמשתמש / מהרשת / מבסיס הנתונים או מכל יחידה אחרת, כנ"ל גם פלט.
* מבנה נתונים בהם נעשה שימוש )רשימה, מחסנית, תור וכדומה(
* יש להראות את זרימ ת המידע בין היחידות השונו ת
* תיאור האלגוריתם הראשי, ותיאור אלגוריתמים עיקריים 2
* הקשרים בין היחידות השונות.
* הצגת מקרה) use case( עבור הפונקציות העיקריות בפרויקט.

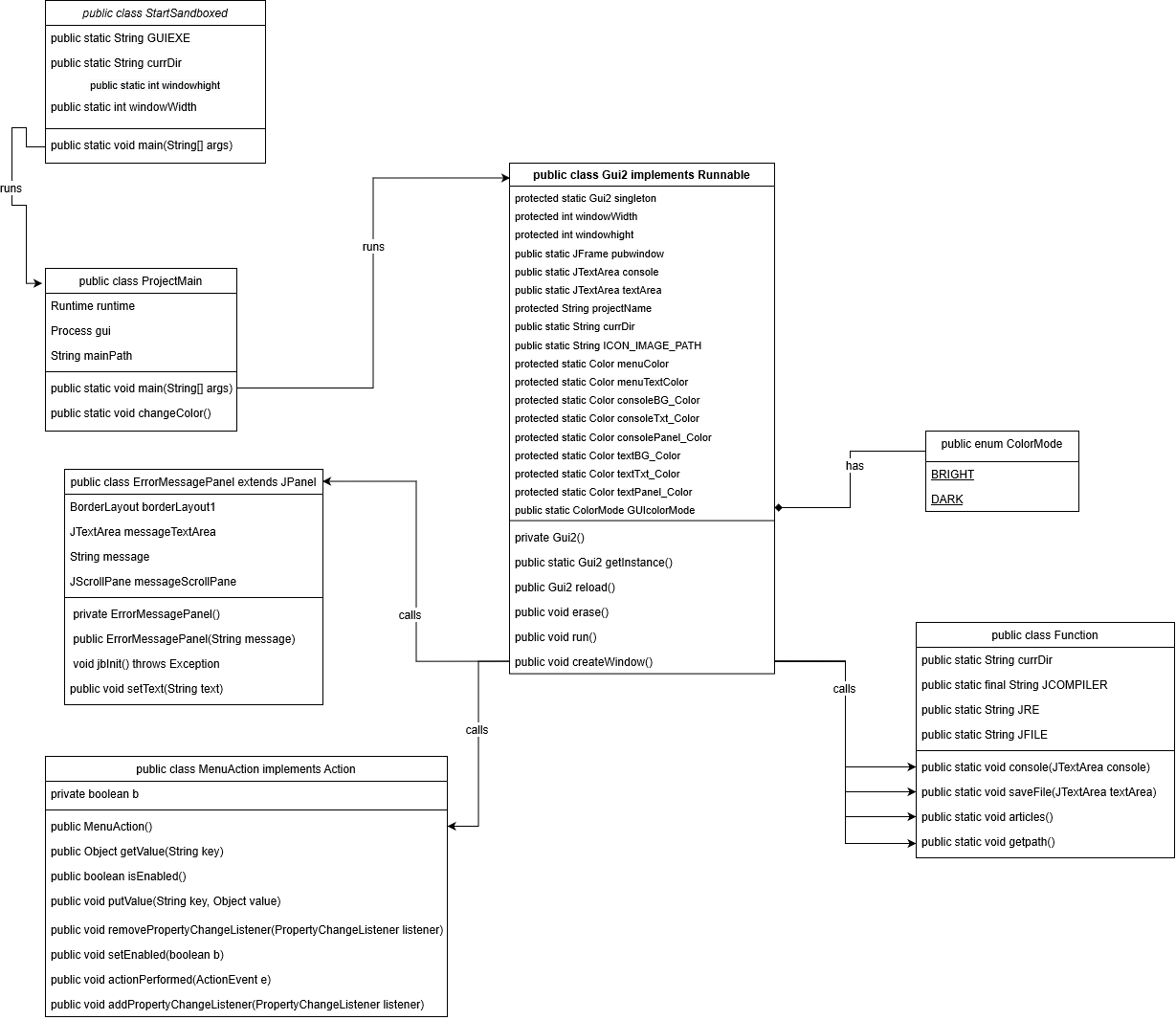
* עץ מודולי ם[[1]](#footnote-1)
* 3Use case diagram
* 3 use cases רשימת
* תרשים UML 3
* 3 Design classes diagram
* רכיבי ממש ק 3
* באילו **טכנולוגיות** [[2]](#footnote-2) נעשה שימוש בפרויקט.
* ממשקים למערכות אחרות) API( 3

הפתרון אשר בחרתי עבור הבעיה הוא יצירת GUI אשר דרכו ניתן יהיה לקמפל ולהריץ תוכנות אשר יכתוב המשתמש. התוכנה תרוץ בתוך Sandbox ע"מ להבטיח את ביטחונו של מחשב המשתמש בזמן ההרצה. בחרתי לעבוד וליצור את ה IDE בשפת java, משום שרציתי לתרגל שימוש בה ומכיוון שהיא כבר הייתה מותקנת על המחשב שלי והייתה זמינה ומוכרת עבורי.

ממשקים בהם נעשה שימוש: java class library (JCL), הספרייה הרשמית מבית אורקל של שפת ג'אווה. התוכנית נכתבה בגרסא 5 של ג'אווה.

טכנולוגיות נוספות בהן נעשה שימוש: java, eclipse, NetBeans, Launch4j, JDK, Draw.io, Adobe xd, Inno Setup Compiler, FileZilla,

תרשים UML



public class StartSandboxed

public class ProjectMain

public class MenuAction implements Action

public class Function

public enum ColorMode

public class Gui2 implements Runnable

public class ErrorMessagePanel extends JPanel

מדריך למשתמש

הוראות התקנה

1. התוכנה מיועדת למחשבים אשר פועלת בהם מערכת הפעלה מסוג windows מבית Microsoft
2. הורדת קובץ ההתקנה SecureIDE\_setup.exe למיקום ידוע במחשב המשתמש. ניתן להוריד מהאתר <http://adam.harelwebs.com/> את הגרסא הקלה, או להעתיק למקום בזיכרון דרך כונן זיכרון נייד (USB, SD) שעליו מותקן הקובץ את הגרסא הכבדה
3. במקרה של הורדה מהאתר יש להיכנס לעמוד הראשי וללחוץ על הכפתור "Download Older Versions", ואז על "Secure IDE". יש להיכנס לתיקייה בה התבצעה ההתקנה ולחלץ כל תיקיית .zip שיכולה להיות שם. כמו כן, צריך להתקין את ה JDK מהאתר (השני מלמטה, jdk-13.0.2\_windows-x64\_bin.exe). לאחר ההתקנה יש להעתיק את שתי התיקיות משם למיקומן בתוכנית.

<https://www.oracle.com/java/technologies/javase-jdk13-downloads.html>

1. לחיצה כפולה על קובץ ההתקנה תפתח אותו, ובשלב זה יש לעבור על תוכנית ההתקנה (installer) ולהתקין את הפרויקט. את התיקייה בה נשמרה ההתקנה יש לזכור לשלבים הבאים
2. פתיחה של סייר קבצים, כניסה לתיקייה של מיקום ההתקנה - > installations. במידה ואחד מהקבצים אינו נמצא שם, ניתן להורידו מהאתר של החברה האחראית עליו
3. לחיצה כפולה על SandboxieInstall-533-3.exe תתחיל התקנה של התוכנה Sandboxie, יש לעבור על תוכנית ההתקנה (installer) ולהתקין את התוכנה
4. לחיצה כפולה על jdk-13.0.2\_windows-x64\_bin.exe תתחיל התקנה של ה SDK של ג'אווה, יש לעבור על תוכנית ההתקנה (installer) ולהתקין את התוכנה, את התיקייה בה נשמרה ההתקנה יש לזכור לשלבים הבאים
5. לאחר סיום כל ההתקנות יש להיכנס למיקום בו נשמר הפרויקט, ותחת התיקייה java להעתיק שתי תיקיות מהתיקייה בה נשמרה ההתקנה של ג'אווה. שתי תיקיות אלו הן jdk-13.0.2 , וגם jre1.8.0\_241
6. לאחר סיום כל ההתקנות וגם סיום העתקת שתי התיקיות הנוספות, יש לבצע בדיקת תקינות מלאה לתוכנית על מנת לוודא כי כל רכיבי התוכנית פועלים כנדרש

בדיקות תקינות

1. לחיצה כפולה על התוכנית כדי לוודא שהיא אכן נפתחת
2. שינוי הצבע מבהיר לכהה ולהפך
3. פתיחה של הלינקים בחלק הלימודי והמשפטי
4. הרצה של פקודת טרמינל (cmd)
5. שגיאות העשויות להיווצר ממקרי קצה למיניהם
6. כתיבת קוד במקום הראוי לכך ושמירתו כקובץ ג'אווה (.java)
7. הרצה של הקוד ובדיקה של: קלט, פלט, שגיאת קוד, זריקת שגיאה, שגיאת קומפילציה, שגיאת זמן ריצה, שגיאה של ה JVM
8. שינוי המיקום של הקובץ ג'אווה וחזרה על סעיף (6)

פרק זה יכיל את ההסבר לדרישות ההתקנה בסביבת העבודה, **הוראות התקנה** ואילו **קבצים נדרשים**, באילו תיקיות וכדומה.

כאן יש שוני בין החלופות השונות

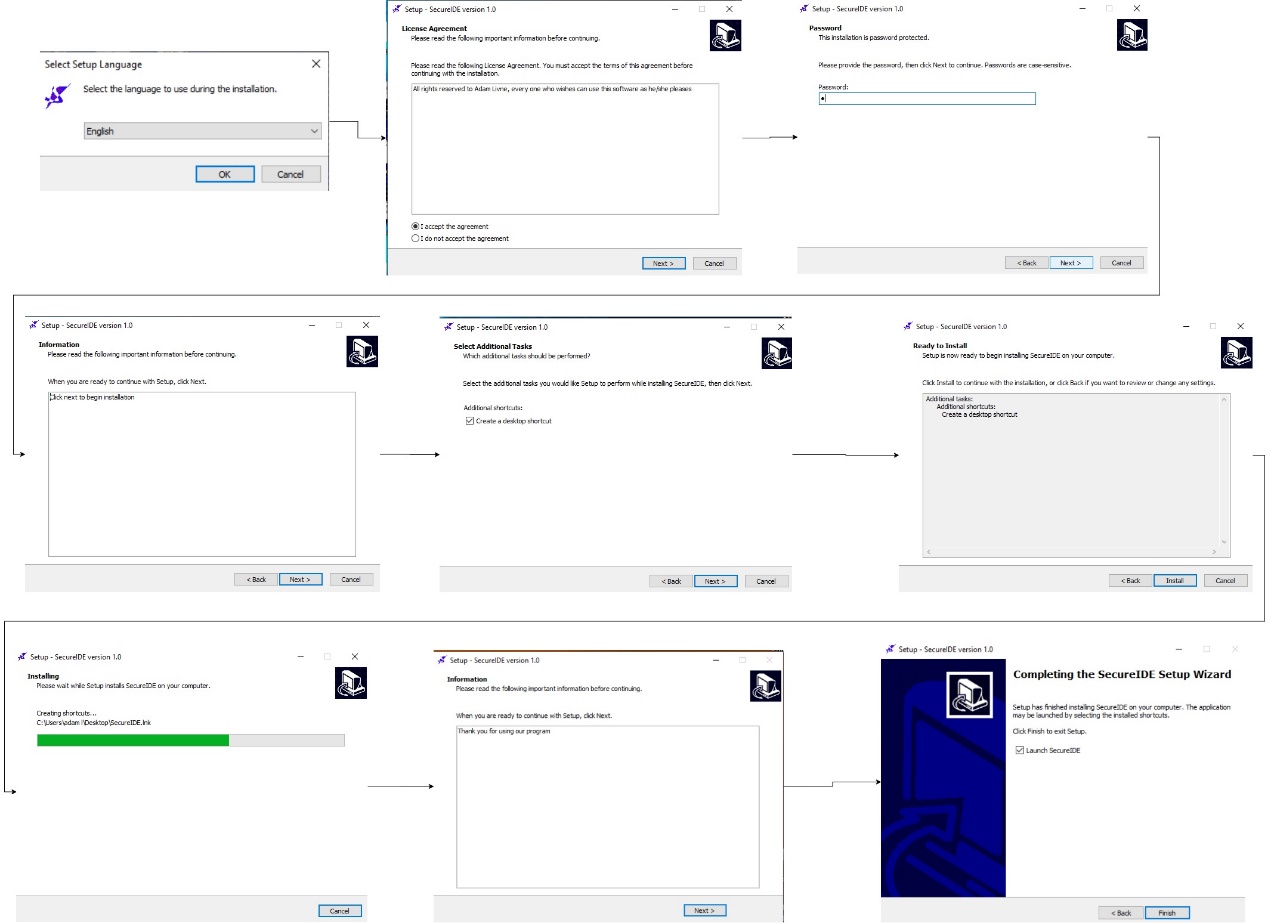
בחלופות - טלפונים ניידים / מערכות מנהליות / שירותי רשת / או כל פרויקט שיש בו מסכים שונים יש לתא ר

* תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם )Screen flow diagram (
* מה תפקידו של כל מסך / חלון עם **צילום מס ך** של החלון הרלוונטי.
* תיאור מסך הפתיחה – מה הוא מכיל והאם משמש נקודת ניווט
* כל מסכי האפליקציה / אתר / מערכת מנהלי ת, בליווי הסברים.
* עבור כל אלמנט תצוגה כדוג מת: כפתור, תיבת טקסט יש להסביר את תפקידם.
* הודעות למשתמש) alert למיניהם.(

בחלופות שאין בהן מסכים או/ו חלונות תצוגה, למשל בחלופת cyber יש לתאר את:

* תפקידו של כל חלק / יחידה בפרויקט
* אילו סימנים )אינדיקציות( המשתמש יכול לקבל כדי לוודא שמה שאמור לקרות קורה.
* מה הפלטים של כל יחידה )יכול להיות בעזרת קבצי לוג או בעזרת אלמנטים אחרים.(
* אילו חיווים המשתמש יכול / אמור לקבל על מנת להפעיל ולבקר את פעולתה התקינה של יחידה הרלוונטית.

בכל החלופות - ההסבר הוא עבור המשתמש. כמו כן יש לציין אם יש מגבלות או אילוצים באי אילו אלמנטים בחלון / מסך המדובר )מס תווים / חוזק סיסמה, וכיוצא בזה**(**



## מדריך למפתח

Use cases

* הדפסה
* קלט
* מספר קלטים
* שגיאת מערכת (של ה JVM)
* שגיאת קוד (throw, זמן ריצה)
* שגיאת קומפילציה
* האם הקוד עוצר כשצריך

חלק זה הינו במידה רבה הרחבה של המ דריך למשתמ ש

עבור **כל קוב ץ** שקיים בפרויקט )כולל קבצי תצורה / קונפיגורציה( יש להציג:

1. שם הקובץ / מחלקה ותפקידו / תפקיד המחלקה.
2. מיקומו )בספרית תיקיות הפרויקט(
3. תוכנו )הכוונה **לתדפיס של הקו ד** ולא צילום מסך שלו( הכולל הערות במקומות הרלוונטיים

)לרוב כאן נכנס תיעוד הקוד(

1. הסבר על כל משתנה בקובץ) אין הכוונה שתלמיד יסביר מה תפקידו של i בלולאה, אך מצופה לקבל הסבר עבור משתנה שמחזיק רשימה של אובייקטים למשל(
2. הסבר עבור הפונקציות בקובץ )אין הכוונה בפעולות get, set אלא הפונקציות שממשמות פעולה משמעותית בקוד(
3. אם רלוונטי גם UML

קבצי טקסט

AFTER\_INST.txt – מסמך טקסט אשר תוכנו מופיע לפני ההתקנה

BEFORE\_INST.txt - מסמך טקסט אשר תוכנו מופיע אחרי ההתקנה

LICENSE.txt - מסמך טקסט אשר תוכנו מכיל את התנאי השימוש

תיקיות

fotos\ – תיקייה המכילה קובצי תמונות ואייקונים

Sandboxie\ - תיקייה המכילה את קובצי תוכנית ה Sandbox

קבצי הרצה

SandboxieInstall-533-3.exe \ installations- קובץ ההתקנה של התוכנית Sandboxie

JavaSetup8u241.exe \ installations- קובץ ההתקנה של ה JVM וה JDK

java\jdk-13.0.2\bin\ javac.exe – קובץ ההרצה של הקומפיילר של גאווה

java\jdk-13.0.2\bin\ java.exe – קובץ ההרצה של ה JRE

java\jdk-13.0.2\bin\adam\_prog\ Test.java – קובץ ברירת המחדל של ההרצה (Run)

SecureIDE\_setup.exe – קובץ ההתקנה של הפרוייקט

StartSandboxed.exe – קובץ הרצה עבור הפרוייקט תחת הגנת ה Sandbox

SecureIDE.exe – קובץ הרצה עבור הפרוייקט, ללא הגנת ה Sandbox

StartSandboxed.exe

/\*\*

\*

\*/

**package** ver\_14;

**import** java.awt.Dimension;

**import** java.io.File;

**import** javax.swing.JFrame;

**import** javax.swing.JOptionPane;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTextArea;

/\*\*

\* The main executable of the program<br />

\* starts the <code>Gui2</code> of the project in the sandbox <br />

\* needs to be under <b>Sandboxie</b> directory

\* **@author** adam l

\* **@version** Mar 14, 2020

\* **@since** ver 8

\*/

**public** **class** StartSandboxed

{

**public** **static** String currDir = System.getProperty("user.dir"); // current directury of the program

**public** **static** String GUIEXE = currDir+"\\SecureIDE.exe"; // directory of executable

**public** **static** **int** windowWidth = 50;

**public** **static** **int** windowhight = 50;

/\*\*

\* **@param** args

\*/

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

{

System.out.println(GUIEXE);

Process pro;

**try**{

pro = Runtime.getRuntime().exec("Sandboxie\\Start.exe "+GUIEXE);

System.out.println("exit value: 0");

}**catch**(Exception e){

javax.swing.JOptionPane.showMessageDialog(**null**, "couldn't execute");

javax.swing.JOptionPane.showMessageDialog(**null**, "error: \n"+e);

System.err.println("couldn't execute: \n");

e.printStackTrace();

}

}

}

SecureIDE.exe

ProjectMain.java

**package** ver\_14;

/\*\*

\* the main thread/class of my cyber project

\*/

**import** java.awt.Color;

**import** java.lang.Thread;

**import** javax.swing.JTextArea;

/\*\*

\* **@author** adam l

\* **@version** Oct 16, 2019

\* **@since** ver\_1

\*/

**public** **class** ProjectMain

{

/\*\*

\* **@param** args

\*/

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

{

Gui2.GUIcolorMode = ColorMode.BRIGHT; // defauld color mode

Thread gui = **new** Thread(Gui2.getInstance(), "gui2");

gui.run();

}

/\*\*

\* changes the color mode of the gui from DARK to BRIGHT and vice versa

\*/

**public** **static** **void** changeColor(){

**if**(Gui2.GUIcolorMode == ColorMode.BRIGHT){

Gui2.GUIcolorMode = ColorMode.DARK;

Gui2 instance = Gui2.getInstance();

instance.reload();

Thread gui = **new** Thread(Gui2.getInstance(), "gui2");

gui.run();

}

**else** **if**(Gui2.GUIcolorMode == ColorMode.DARK){

Gui2.GUIcolorMode = ColorMode.BRIGHT;

Gui2 instance = Gui2.getInstance();

instance.reload();

Thread gui = **new** Thread(Gui2.getInstance(), "gui2");

gui.run();

}

System.out.println(Gui2.GUIcolorMode);

}

}

MenuAction.java

**package** ver\_14;

**import** java.awt.Dimension;

**import** java.awt.event.ActionEvent;

**import** java.beans.PropertyChangeListener;

**import** javax.swing.Action;

**import** javax.swing.JFrame;

**import** javax.swing.JOptionPane;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTextArea;

/\*\*

\* an implementation of the <code>javax.swing.Action</code> interface <br />

\* helps to clean out the code at <code>Gui2</code>

\* **@author** adam l

\* **@version** Mar 10, 2020

\* **@see** <code>Gui2</code>

\* **@since** ver\_3

\*/

**public** **class** MenuAction **implements** Action

{

**private** **boolean** b=**true**;

**public** MenuAction(){}

**public** Object getValue(String key){**return** **null**;}

**public** **boolean** isEnabled(){**return** **this**.b;}

**public** **void** putValue(String key, Object value){}

**public** **void** removePropertyChangeListener(PropertyChangeListener listener){}

**public** **void** setEnabled(**boolean** b){**this**.b = b;}

**public** **void** actionPerformed(ActionEvent e){}

**public** **void** addPropertyChangeListener(PropertyChangeListener listener){}

}

Gui2.java

**package** ver\_14;

/\*\*

\* the GUI for my cyber project

\*/

**import** java.awt.BorderLayout;

**import** java.awt.Canvas;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.GraphicsConfiguration;

**import** java.awt.GraphicsDevice;

**import** java.awt.HeadlessException;

**import** java.awt.MenuItem;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseListener;

**import** java.awt.image.BufferedImage;

**import** java.beans.PropertyChangeListener;

**import** java.io.BufferedInputStream;

**import** java.io.BufferedOutputStream;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.OutputStreamWriter;

**import** java.io.UnsupportedEncodingException;

**import** java.io.Writer;

**import** java.util.Scanner;

**import** javax.imageio.ImageIO;

**import** javax.swing.Action;

**import** javax.swing.BoxLayout;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuItem;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTextArea;

**import** javax.swing.JToolBar;

/\*\*

\* this class builds the projects' GUI

\* **@author** adam l

\* **@version** Mar 14, 2020

\* **@since** ver\_1

\*/

**public** **class** Gui2 **implements** Runnable

{

**protected** **static** Gui2 singleton = **null**;

**protected** **int** windowWidth = 1500;

**protected** **int** windowhight = 1000;

**public** **static** JFrame pubwindow; // programs' main window

**public** **static** JTextArea console; // io handling

**public** **static** JTextArea textArea; // where code is written

**protected** String projectName = "Secure IDE";

**public** **static** String currDir = System.getProperty("user.dir"); // current directury of the program

**public** **static** String ICON\_IMAGE\_PATH = currDir+"\\fotos\\SecureIDE.jpg"; // directury of programs' icon

**protected** **static** Color menuColor; // color of the menu

**protected** **static** Color menuTextColor; // color of the text at menu

**protected** **static** Color consoleBG\_Color; // color of the console background

**protected** **static** Color consoleTxt\_Color; // color of the console text

**protected** **static** Color consolePanel\_Color; // color of the console panel

**protected** **static** Color textBG\_Color; // color of the text area background

**protected** **static** Color textTxt\_Color; // color of the text area's code

**protected** **static** Color textPanel\_Color; // color of the text area's panel

**public** **static** ColorMode GUIcolorMode;

**private** Gui2(){

System.out.println("Gui2()");

}

/\*\*

\* returnes the only instance of the class <br />

\* if there is no instance, a new one is created

\* **@return** <code>Gui2</code>

\*/

**public** **static** Gui2 getInstance(){

**if**(Gui2.singleton==**null**)

Gui2.singleton = **new** Gui2();

**return** Gui2.singleton;

}

/\*\*

\* erases the old gui and returns a new one

\* **@return** <code>Gui2</code>

\*/

**public** Gui2 reload(){

Gui2.singleton = **new** Gui2();

**return** Gui2.singleton;

}

/\*\*

\* erases the current instance of the gui

\*/

**public** **void** erase(){ Gui2.singleton = **null**; }

/\*\*

\* creates the <code>JFrame</code> main window of the project

\*

\*/

**public** **void** createWindow(){

System.out.println("window created");

JFrame window = **new** JFrame(**this**.projectName);

// set colors

**if**(**this**.GUIcolorMode == ColorMode.DARK){

Gui2.menuTextColor = Color.WHITE;

**this**.menuColor = Color.DARK\_GRAY;

**this**.consoleBG\_Color = Color.BLACK;

**this**.consoleTxt\_Color = **new** Color(100,100,255); //Red?

**this**.textBG\_Color = **new** Color(0, 50, 0, 255); // RGBA 0-255

**this**.textTxt\_Color = Color.YELLOW;

**this**.consolePanel\_Color = **this**.menuColor;

**this**.textPanel\_Color = **this**.consolePanel\_Color;

}

**else** **if**(**this**.GUIcolorMode == ColorMode.BRIGHT){

**this**.menuTextColor = Color.BLACK;

**this**.menuColor = **null**;

**this**.consoleBG\_Color = Color.WHITE;

**this**.consoleTxt\_Color = Color.RED;

**this**.textBG\_Color = Color.WHITE;

**this**.textTxt\_Color = Color.BLACK;

**this**.consolePanel\_Color = **new** Color(0, 0, 255, 10); // RGBA 0-255

**this**.textPanel\_Color = **this**.consolePanel\_Color;

}

**try** {

BufferedImage projIcon = ImageIO.read(**new** File(ICON\_IMAGE\_PATH)); // not supporting .png and .ico

window.setIconImage(projIcon);

} **catch** (Exception e) {

System.err.println("failed to load icon image "+e);

}

Gui2.pubwindow = window;

// JFrame.DISPOSE\_ON\_CLOSE sets 'X' to close the current window , JFrame.EXIT\_ON\_CLOSE sets 'X' to close the app

window.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

window.setSize(windowWidth, windowhight); // should be changeable & default full screen

// window.pack(); // automatically changes the size of the frame according to the size of components in it

window.setExtendedState(JFrame.MAXIMIZED\_BOTH); // sets window at full screen

window.setLocation(0,0); // from top - left

// text area

JPanel p1 = **new** JPanel();

**final** JTextArea textArea = **new** JTextArea(29, 171);

JScrollPane scroll = **new** JScrollPane(textArea);

scroll.requestFocus(); //InWindow

textArea.setBackground(**this**.textBG\_Color);

textArea.setForeground(**this**.textTxt\_Color);

textArea.setFont(**new** Font("font", 5, 13));

textArea.setEditable(**true**);

p1.add(scroll);

p1.setBackground(**this**.textPanel\_Color);

p1.setEnabled(**true**);

window.getContentPane().add(BorderLayout.CENTER, p1);

// console

JPanel p = **new** JPanel();

window.getContentPane().add(BorderLayout.SOUTH, p);

**final** JTextArea console = **new** JTextArea(22, 145);

JScrollPane consoleScroll = **new** JScrollPane(console);

console.setForeground(**this**.consoleTxt\_Color);

console.setBackground(**this**.consoleBG\_Color);

console.setFont(**new** java.awt.Font("font", 30, 15));

consoleScroll.requestFocus(); //InWindow

p.add(consoleScroll);

p.setMinimumSize(**new** Dimension(22, 145));

p.setBackground(**this**.consolePanel\_Color);

// menu

javax.swing.JMenuBar menu = **new** javax.swing.JMenuBar();

JMenuItem item = **new** JMenuItem(**new** Action(){

**private** **boolean** b = **true**;

**public** Object getValue(String key){**return** **null**;}

**public** **boolean** isEnabled(){**return** **this**.b;}

**public** **void** putValue(String key, Object value){}

**public** **void** removePropertyChangeListener(PropertyChangeListener listener){}

**public** **void** setEnabled(**boolean** b){**this**.b = b;}

**public** **void** actionPerformed(ActionEvent e){textArea.append(**this**.NAME);}

**public** **void** addPropertyChangeListener(PropertyChangeListener listener){ }

});

item.getAccessibleContext().setAccessibleName("menu");

item.setText("menu");

// File

JMenu file = **new** JMenu("File");

file.setForeground(**this**.menuTextColor);

JMenuItem saveas = **new** JMenuItem("Save As...");

JMenuItem exitItem = **new** JMenuItem("Exit");

exitItem.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){textArea.append("Exit\n");java.lang.System.exit(0);}});

// Save As

saveas.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){Function.saveFile(textArea);}});

JMenuItem saveasfile = **new** JMenuItem("Save As File...");

// Save As File

saveasfile.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){Function.GUIsaveFile(textArea);}});

file.add(saveas);

file.add(saveasfile);

file.addSeparator();

file.add(exitItem);

// Window

JMenu windowMenu = **new** JMenu("Window");

windowMenu.setForeground(**this**.menuTextColor);

JMenu showMenu = **new** JMenu("Show");

JMenuItem changeC = **new** JMenuItem("Change color");

changeC.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){ProjectMain.changeColor();}});

showMenu.add(changeC);

JMenuItem showArticlesItem = **new** JMenuItem("Articles");

showArticlesItem.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){Function.articles();}});

showMenu.add(showArticlesItem);

windowMenu.add(showMenu);

// Edit

JMenu edit = **new** JMenu("Edit");

edit.setForeground(**this**.menuTextColor);

JMenuItem changepath = **new** JMenuItem("Change Path");

changepath.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){Function.getpath();}});

edit.add(changepath);

// Tools

JMenu tools = **new** JMenu("Tools");

tools.setForeground(**this**.menuTextColor);

JMenuItem runcmd = **new** JMenuItem("Run Command");

runcmd.addActionListener(**new** Action(){

**private** **boolean** b=**true**;

**public** Object getValue(String key){**return** **null**;}

**public** **boolean** isEnabled(){**return** **this**.b;}

**public** **void** putValue(String key, Object value){}

**public** **void** removePropertyChangeListener(PropertyChangeListener listener){}

**public** **void** setEnabled(**boolean** b){**this**.b = b;}

**public** **void** actionPerformed(ActionEvent e){

JTextArea textAreaCmd = **new** JTextArea();

textAreaCmd.setEditable(**true**);

JScrollPane scrollPane = **new** JScrollPane(textAreaCmd);

scrollPane.requestFocus();

textAreaCmd.requestFocusInWindow();

scrollPane.setPreferredSize(**new** java.awt.Dimension(800, 600));

JOptionPane.showMessageDialog(**null**, scrollPane, "Enter Command", JOptionPane.PLAIN\_MESSAGE);

String cmmnd = textAreaCmd.getText();

**try**{

java.util.Scanner s = **new** java.util.Scanner(Runtime.getRuntime().exec(cmmnd).getInputStream()).useDelimiter("\\A");

String cmdret="";

**while**(s.hasNext()) cmdret+=s.next();

javax.swing.JOptionPane.showMessageDialog(**null**, String.format("executing command: %s", cmmnd));

JFrame f = **new** JFrame();

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // sets 'X' to close the app

f.setExtendedState(JFrame.MAXIMIZED\_BOTH); // sets window at full screen

f.setLocation(0,0);

f.setMaximumSize(**new** Dimension(10, 10));

ErrorMessagePanel cmdreturn = **new** ErrorMessagePanel(cmdret);

cmdreturn.setMaximumSize(**new** Dimension(10, 10));

JOptionPane.showMessageDialog(f, cmdreturn);

}**catch**(Exception e1){javax.swing.JOptionPane.showMessageDialog(**null**, "couldn't execute");}

}

**public** **void** addPropertyChangeListener(PropertyChangeListener listener){

}});

tools.add(runcmd);

// Run

JMenu Run = **new** JMenu("Run");

Run.setForeground(**this**.menuTextColor);

JMenuItem Run2Item = **new** JMenuItem("Run");

Run2Item.addActionListener(**new** MenuAction(){

**public** **void** actionPerformed(ActionEvent e){Function.console(console);}});

Run2Item.setBackground(Color.ORANGE);

Run.add(Run2Item);

menu.setBackground(**this**.menuColor);

menu.add(file);

menu.add(windowMenu);

menu.add(edit);

menu.add(tools);

menu.add(Run);

window.getContentPane().add(BorderLayout.PAGE\_START, menu);

window.setVisible(**true**);

}

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

{

System.out.println("run() Gui2");

createWindow();

}

}

Function.java

**package** ver\_14;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Component;

**import** java.awt.Desktop;

**import** java.awt.Dimension;

**import** java.awt.FileDialog;

**import** java.awt.Point;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

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

**import** java.awt.image.BufferedImage;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.OutputStreamWriter;

**import** java.io.PrintWriter;

**import** java.io.Writer;

**import** java.net.URI;

**import** java.net.URL;

**import** java.util.Scanner;

**import** javax.imageio.ImageIO;

**import** javax.swing.JButton;

**import** javax.swing.JFileChooser;

**import** javax.swing.JFrame;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextArea;

/\*\*

\* implements some helpful functions that could be written in other place, to clean out the code <br />

\* all functions here must be <b>static</b>

\* **@author** adam l

\* **@version** Mar 10, 2020

\* **@see** <code>Gui2</code>

\* **@since** ver\_5

\*/

**public** **class** Function

{

**public** **static** String currDir = System.getProperty("user.dir"); // current directury of the program

**public** **static** **final** String JCOMPILER = System.getProperty("user.dir")+"\\java\\jdk-13.0.2\\bin\\javac.exe"; // the javac compilers' location

**public** **static** String JRE = currDir+"\\java\\jdk-13.0.2\\bin\\java.exe"; // the java JVM(JRE)s' location path on local device

**public** **static** String JFILE = currDir+"\\java\\jdk-13.0.2\\bin\\adam\_prog\\Test.java"; // the file to compile & execute

/\*\*

\* Compiles and runs the code that was written by the user in the textArea. <br />

\* In addition, handles any input/output/error that the runing program has <br />

\* **@param** console the <code>javax.swing.JTextArea</code> that acts as a <b>console</b> <br />

\* compiling file, executing, and printing to the console

\*/

**public** **static** **void** console(javax.swing.JTextArea console){

**int** exitV = 0;

**int** rowCount = 0;

**try**{

// running compiler

**if**(Function.JFILE == **null** || Function.JCOMPILER == **null** || Function.JRE == **null**) **throw** **new** Exception("empty path");

java.lang.Process proc = **new** ProcessBuilder(JCOMPILER, JFILE).start();

Scanner s2 = **new** Scanner(proc.getInputStream()).useDelimiter("\\A");

String cmdret2=**null**;

**while**(s2.hasNext()) cmdret2+=s2.next();

System.out.println(cmdret2);

proc.waitFor();

System.out.println(proc.exitValue());

// running .class file (executing)

Process prog = **new** ProcessBuilder(JRE, JFILE).start(); // working

// get errors from program

java.util.Scanner progErr = **new** Scanner(prog.getErrorStream()).useDelimiter("\\A");

java.lang.String progerr="";

**do**{

**try**{ // check if the program exited

exitV = prog.exitValue(); // throws exception if program hasn't exited yet

System.out.println(exitV);

**while**(progErr.hasNext()) progerr+=progErr.next();

**if**(exitV != 0){

javax.swing.JOptionPane.showMessageDialog(**null**, "couldn't execute exit code: "+exitV);

System.err.println(String.format("failed to execute exit code: %d", exitV));

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

console.append(progerr);

System.err.println(progerr);

}

}

**break**;

}**catch**(java.lang.IllegalThreadStateException e){

// get errors from program

java.util.Scanner errSc = **new** Scanner(prog.getErrorStream()).useDelimiter("\\A");

java.lang.String errOut="";

// set output scanner from program

java.util.Scanner progSc = **new** Scanner(prog.getInputStream()).useDelimiter("\\A");

java.lang.String progOut="";

// send input to the program

java.io.BufferedWriter progIn = **new** BufferedWriter(**new** OutputStreamWriter(prog.getOutputStream()));

progIn.append(console.getText());

progIn.newLine();

progIn.close();

// get output from program

**while**(progSc.hasNext()) progOut+=progSc.next();

System.out.println(progOut);

**if**(progOut != **null**) console.append(progOut);

// get error from program

**while**(errSc.hasNext()) errOut+=errSc.next();

System.out.println(errOut);

**if**(errOut != **null**) console.append(errOut);

}

}**while**(**true**);

}**catch**(Exception e1){

javax.swing.JOptionPane.showMessageDialog(**null**, String.format("couldn't execute: %s", e1.toString()));

System.err.println(String.format("failed to execute %s: ", "notship.exe")+e1);

System.err.println(exitV);

e1.printStackTrace();

}

}

/\*\*

\* Saves the text areas' content as a file at users' choise directory

\* **@param** textArea the <code>javax.swing.JTextArea</code> to get the users' code from

\*/

**public** **static** **void** saveFile(javax.swing.JTextArea textArea){

JTextArea pathWindow = **new** JTextArea();

pathWindow.setEditable(**true**);

JScrollPane scrollPane = **new** JScrollPane(pathWindow);

scrollPane.requestFocus();

pathWindow.requestFocusInWindow();

scrollPane.setPreferredSize(**new** Dimension(800, 600));

JOptionPane.showMessageDialog(**null**, scrollPane, "Paste path Info", JOptionPane.PLAIN\_MESSAGE);

String path = pathWindow.getText(); // also works with a single \

File file = **new** File(path);

**try**{

java.io.PrintWriter writer = **new** PrintWriter(file);

// writer.print("");

writer.close(); // erases files' content

Writer my\_file = **new** BufferedWriter(**new** OutputStreamWriter(**new** FileOutputStream(file, **true**), "UTF-8"));

my\_file.append(textArea.getText());

my\_file.close();

javax.swing.JOptionPane.showMessageDialog(**null**, "File saved");

Function.JFILE = path;

textArea.append("okkk\n");

}**catch**(Exception e1){javax.swing.JOptionPane.showMessageDialog(**null**, String.format("failed to save file: ")+e1);e1.printStackTrace();};

}

/\*\*

\* creates the articles window with the links

\*/

**public** **static** **void** articles(){

javax.swing.JFrame articles = **new** JFrame("articles");

java.awt.Dimension size = **new** Dimension(800, 600);

**try**{

**final** java.awt.Desktop desk = Desktop.getDesktop();

**try** {

BufferedImage projIcon = ImageIO.read(**new** File(Gui2.ICON\_IMAGE\_PATH)); // not supporting .ico

articles.setIconImage(projIcon);

} **catch** (Exception e) {

System.err.println("failed to load icon image "+e);

JOptionPane.showMessageDialog(**null**, "error: failed to load icon image: \n"+e);

}

JPanel panel = **new** JPanel();

**final** JButton ai = **new** JButton("Artificial Intelligence");

ai.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try**{

desk.browse(**new** java.net.URI("https://he.wikipedia.org/wiki/%D7%91%D7%99%D7%A0%D7%94\_%D7%9E%D7%9C%D7%90%D7%9B%D7%95%D7%AA%D7%99%D7%AA"));

}**catch**(Exception e1){

e1.printStackTrace();}}});

**final** JButton cs = **new** JButton("Computer Science");

cs.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try**{

desk.browse(**new** URI("https://en.wikipedia.org/wiki/Computer\_science#Computer\_graphics\_and\_visualization"));

}**catch**(Exception e1){

e1.printStackTrace();}}});

**final** JButton siw = **new** JButton("Secure IDE Website");

siw.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try**{

desk.browse(**new** URI("https://adam.harelwebs.com/"));

}**catch**(Exception e1){

e1.printStackTrace();}}});

panel.add(ai);

panel.add(cs);

panel.add(siw);

articles.getContentPane().add(panel);

articles.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); // sets 'X' to close the current window

// articles.setSize(size); // should be changeable & default full screen

articles.pack(); // automatically changes the size of the frame according to the size of components in it

// articles.setExtendedState(JFrame.MAXIMIZED\_BOTH); // sets window at full screen

articles.setLocation(**new** java.awt.Point(900, 500)); // from top - left

articles.setVisible(**true**);

}**catch**(Exception e){

javax.swing.JOptionPane.showMessageDialog(**null**, "error: can't find url: \n"+e);

System.err.println("can't find url: \n");

e.printStackTrace();

}

}

/\*\*

\* Saves the users' code in a file at entered directory

\*/

**public** **static** **void** getpath(){

javax.swing.JFileChooser fc = **new** JFileChooser();

**int** returnVal = fc.showOpenDialog(**new** JFrame());

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

File file = fc.getSelectedFile();

Function.JFILE = file.getPath();

javax.swing.JOptionPane.showMessageDialog(**null**, "JFILE set to location: "+file.getPath());

} **else** {

javax.swing.JOptionPane.showMessageDialog(**null**, "error: File Not Found");

System.err.println("error: File Not Found");

}

}

/\*\*

\* Saves the users' code in a file at entered directory <br />

\* this is a later version of <code>getpath()</code>

\* **@return** the saved files' location (path)

\* **@throws** FileNotFoundException

\*/

**public** **static** String returnpath() **throws** FileNotFoundException{

javax.swing.JFileChooser fc = **new** JFileChooser();

**int** returnVal = fc.showOpenDialog(**new** JFrame());

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

File file = fc.getSelectedFile();

javax.swing.JOptionPane.showMessageDialog(**null**, "JFILE set to location: "+file.getPath());

**return** file.getPath();

} **else** {

javax.swing.JOptionPane.showMessageDialog(**null**, "error: File Not Found");

System.err.println("error: File Not Found");

**throw** **new** java.io.FileNotFoundException();

}

}

/\*\*

\* Saves the text areas' content as a file at users' choise directory

\* this is a copy of saveFile() that uses <code>returnpath()</code>

\* **@param** textArea the <code>javax.swing.JTextArea</code> in which the user writes code

\*/

**public** **static** **void** GUIsaveFile(javax.swing.JTextArea textArea){

JOptionPane.showMessageDialog(**null**, "Select path Info");

**try**{

String path = Function.returnpath();

File file = **new** File(path);

**if**(file.isFile()){

java.io.PrintWriter writer = **new** PrintWriter(file);

writer.close(); // erases files' content

Writer my\_file = **new** BufferedWriter(**new** OutputStreamWriter(**new** FileOutputStream(file, **true**), "UTF-8"));

my\_file.append(textArea.getText());

my\_file.close();

javax.swing.JOptionPane.showMessageDialog(**null**, "File saved");

Function.JFILE = path;

}**else**{**throw** **new** FileNotFoundException();}

}**catch**(Exception e1){javax.swing.JOptionPane.showMessageDialog(**null**, String.format("failed to save file: ")+e1);e1.printStackTrace();};

}

}

ErrorMessagePanel.java

**package** ver\_14;

**import** java.awt.\*;

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

/\*\* Jenerates an error message <br />

\* This is non-production code, be careful.

\* **@author** adam l

\* **@version** Mar 10, 2020

\* **@since** ver\_1

\*/

**public** **class** ErrorMessagePanel **extends** JPanel

{

BorderLayout borderLayout1 = **new** BorderLayout();

JScrollPane messageScrollPane = **new** JScrollPane();

JTextArea messageTextArea = **new** JTextArea();

String message; // the displayed text

/\*\*

\* Creates an error message

\* **@param** message

\*/

**public** ErrorMessagePanel(String message)

{

**this**.message = message;

**try**

{

jbInit();

}

**catch**(Exception ex)

{

ex.printStackTrace();

}

}

**private** ErrorMessagePanel()

{

}

/\*\*

\* **@throws** Exception

\*/

**void** jbInit() **throws** Exception

{

**this**.setLayout(borderLayout1);

messageScrollPane.add(messageScrollPane.createHorizontalScrollBar());

messageTextArea.setEnabled(**true**);

messageTextArea.setEditable(**false**);

messageTextArea.setLineWrap(**true**);

messageTextArea.setText(message);

// messageTextArea.setSize(messageTextArea.getPreferredSize());

// messageTextArea.setPreferredSize(messageTextArea.getPreferredSize());

**this**.add(messageScrollPane, BorderLayout.CENTER);

messageScrollPane.getViewport().add(messageTextArea, **null**);

}

/\*\*

\* Changes the text that will be displayed

\* **@param** text

\*/

**public** **void** setText(String text)

{

**this**.message = text;

}

}

ColorMode.java

/\*\*

\*

\*/

**package** ver\_14;

/\*\*

\* represents in which color mode should the Gui2 work

\* **@author** adam l

\* **@version** Mar 21, 2020

\* **@since** ver\_10

\*/

**public** **enum** ColorMode

{

DARK,

BRIGHT

}

## סיכום אישי / רפלקציה

חלק זה הינו חשוב ביותר, ואין להקל בו ראש כלל. התלמיד ישקף:

* איך הייתה עבורו העבודה על הפרויק ט
* מה הוא קיב ל
* אילו כלים הוא לוקח אתו להמש ך
* מה הקשיים / אתגרים שעמדו בפני ו
* מה המסקנות של ו
* מה הוא היה עושה אחרת לו היה מתחיל היו ם
* מה אם היה קורה אחרת העבודה הייתה יעילה יותר עבורו
* ועוד שאלות חקר עצמי לשיקולכם.

מצופה מתלמיד שעבד כמעט שנה )ולעיתים אף יותר( על פרויקט שלפחות ימלא חצי עד עמוד שלם בחלק זה של הרפלקציה.

העבודה על הפרויקט עבורי הייתה מאד מעניינת ומאתגרת, ונהניתי ממנה מאד למרות כי התוצאה הסופית לא הייתה כפי שציפיתי שתהיה. גיליתי דברים מאד מפתיעים, תוספות שהייתי בטוח שיהיו מסובכות מדיי למימוש התגלו כחלקים היותר קלים של הפרויקט ודווקא דברים שהייתי בטוח בפשטותם התגלו כהרבה יותר עמוקים ומסובכים ממה שנראה בהתחלה. מהעבודה הרבה שהשקעתי קיבלתי הבנה עמוקה של הדרך בה עובדות הפלטפורמות בהן אני משתמש באופן יומיומי לתכנות, ולהמשך אני לוקח איתי את הידע הזה ואת המוצר שעשיתי. בתחילת העבודה היה לי קושי להחליט אילו דברים אני רוצה להוסיף לפרויקט מבלי שיגזלו ממני זמן רב מדיי, וגם קושי בהבנה של הצורה בה בנויה השפה בה כתבתי. מההתמודדות שלי עם קשיים אלה הגעתי למסקנה כי צריך לעבוד לפי שיטה מסודרת מראש. אם הייתי מתחיל את העבודה היום, ככל הנראה כי הייתי קודם חוקר לעומק את הנושא אליו אני נכנס ורק אז מתחיל לעבוד על הפרויקט. אני חושב כי אם הייתי פועל כך מההתחלה העבודה הייתה יותר יעילה עבורי.

## ביבליוגרפיה

מומלץ ואף רצוי לעודד תלמידים לרשום את מקורות המידע שהם עשו בהם שימוש, אמנם חלק זה אינו תמיד רלוונטי לכל הפרויקטים, אך כאשר התלמיד ביצע מחקר, חלק זה חיוני וחובה. מקורות יכולים להיות מאמרים, קישורים לאתרי מידע, ספרים, כתבי עת וכדומה.

יש להקפיד להציג את הרשימה לפי כללי ה – APA

במהלך עבודתי נעזרתי באתרים ובמקורות הבאים:

<http://launch4j.sourceforge.net/>

<https://coderanch.com/t/511249/java/handle-JFrame-minimize-restore>

<https://www.oracle.com/java/technologies/javase-jdk13-downloads.html>

<https://jrsoftware.org/isinfo.php>

<https://java.com/en/download/manual.jsp>

<https://www.online-convert.com/result/5b7a6519-2263-4a54-bc64-a9a1a946a6e1>

<https://www.w3schools.com/java/java_enums.asp>

<https://www.youtube.com/watch?v=WRwPVZ4jmNY&list=WL&index=11&t=10337s>

<https://www.tutorialspoint.com/>

<https://github.com/>

<https://docs.microsoft.com/>

<https://stackoverflow.com/>

<https://docs.oracle.com/>

<https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html>

<https://he.wikipedia.org/wiki/>

<https://askleo.com/whats-the-difference-between-a-sandbox-and-a-virtual-machine/>

<https://www.maketecheasier.com/best-sandbox-applications-windows10/>

https://youtu.be/eQkSArQxYJg

https://youtu.be/o95Or-Z\_Ybk

https://youtu.be/bjUoQbSJDJs

<https://tympanus.net/codrops/2011/09/05/slicebox-3d-image-slider/>

https://docs.oracle.com/javase/tutorial/deployment/jar/manifestindex.html

https://www.i-programmer.info/ebooks/134-modern-java/5041-building-a-java-gui-containers.html

<http://adam.harelwebs.com/>

https://www-geeksforgeeks-org.cdn.ampproject.org/v/s/www.geeksforgeeks.org/inter-process-communication-ipc/amp/?amp\_js\_v=a3&amp\_gsa=1&usqp=mq331AQCKAE%3D#aoh=15829205973892&referrer=https%3A%2F%2Fwww.google.com&amp\_tf=%D0%A1%20%D1%81%D0%B0%D0%B9%D1%82%D0%B0%20%251%24s&ampshare=https%3A%2F%2Fwww.geeksforgeeks.org%2Finter-process-communication-ipc%2F

https://www.researchgate.net/publication/321694778\_Interprocess\_Communication\_with\_Java\_in\_a\_Microsoft\_Windows\_Environment

https://mobile.developer.com/java/data/understanding-java-process-and-java-processbuilder.html

<https://www.maketecheasier.com/best-sandbox-applications-windows10/>

[https://askleo.com/whats-the-difference-between-a-sandbox-and-a-virtual-machine](https://askleo.com/whats-the-difference-between-a-sandbox-and-a-virtual-machine/)/

[https://www.sandboxie.com/DownloadSandboxie](https://www.sandboxie.com/DownloadSandboxie/)

https://youtu.be/SpKANTHXwmo

https://netbeans.org/kb/articles/javase-deploy.html

[https://stackoverflow.com/questions/1906445/what-is-the-difference-between-jdk-and-jre](https://stackoverflow.com/questions/1906445/what-is-the-difference-between-jdk-and-jre/)

[https://image.online-convert.com/convert-to-ico](https://image.online-convert.com/convert-to-ico/)

## נספחים

כאן התלמיד יכול להוסיף הסברים על הטכנולוגיות שנעשה בהם שימוש, או כל מידע שיכול להועיל לקורא העבודה

1. אם יש [↑](#footnote-ref-1)
2. הכוונה כאן ליצור אבחנה בין הטכנולוגיה שנעשה בה שימוש לבין הפרויקט שממש אלגוריתם ומשתמש בטכנולוגיה [↑](#footnote-ref-2)