בס"ד

מטלה במונחה עצמים.

מגישים:

**שלום וינברגר**

**אלכסי טיטוב**

תיאור המערכת :

מערכת זו מקבלת קובץ CSV עם נתונים של רשתות וואיפיי , נתונים אלה הם זמן, מיקום, מזהה מכשיר , רשימת רשתות הוואיפיי כולל שם הרשת , MAC שלה, תדר ועוצמת הרשת. לאחר קבלת הקובץ התוכנה יוצרת קובץ CSV חדש אשר מסודר בצורה הבאה :

Time, ID, Lat, Lon, Alt, #WiFi networks (up to 10), SSID1, MAC1, Frequncy1, Signal1, SSID2, MAC2, Frequncy2, Signal2, ...

ישנם לכול היותר 10 נקודות וואיפי לכול שורה , במקרה ויש יותר מעשר צריך לשמור את העשר עם הSIGNAL הכי חזק. לאחר מכן אנחנו בונים תוכנת פילטרים אשר מקבלת את הקובץ שיצרנו מקודם ויכולה לסנן את הקובץ לפי מיקום , זמן, או רשימת משתמשים ,ואז ליצור מיזה קובץ KML .

המערכת שלנו מבוססת על WIGLEWIFI ולכן היא מקבלת רק קבצים המתאימים לקבצים המגיעים מתוכנה זו. בסעיף שלוש אנו מקבלים רק קבצים שיכולים לעבור את סעיף שתיים כלומר קבצי CSV בלבד. השתמשנו באפליקציה wigleWIFI בשביל לסרוק נתוני WIFI וGPS ותוכניות ג'אווה שכתבנו באקליפס (עם שימוש בספריות).

בחלק החדש (EX2) הוספנו שני אלגוריתמים ואת מחלקות העזר שלהם. על מנת לשערך מיקום במידה ואין לנו בסריקה קורדינטות.

המערכת מחולקת לשלושה קבצים עיקריים (EX1):

**סעיף 2 :**

קבלת תקייה,מעבר עליה וקריאת קבצי ה CSV שבה,עובר על כל קובץ CSV ומכניס את המידע של כולם למערך דינאמי אחד. ומשם יוצר קובץ CSV חדש של התוצאות.

בתוכו קיים, אובייקט של DATAWIFI שבתוכו יש אובייקטים של זמן,מיקום. נתונים של נקודת WIFI ושם המכשיר שקלט את הWIFI.

כל אובייקט מכיל אפשרות לקבלת מצב ולשינוי מצב.

**סעיף 3:**

קבלת קובץ CSV שנוצר מסעיף 2 . מעבר על הקובץ עם סינון לפי בקשה. ויצירת קובץ KML מהנתונים העומדים בדרישות.

התוכנית הראשית מכילה רק בנאי שקורה לכל התהליך ולכן התוכנית בנויה ממחלקה פנימית

SimpleAction שבה ממשק הבחירה ומשם התוכנית מתחילה.

ופונק' :

setMAX יוצרת רשימה של הסינגלים החזקים

Search מחפשת לפי MAC את הסינגלים החזקים ומחזירה אמת אם הסינגל היה חזק יותר ממה ששלחו או שקר אם הם שווים.

Removeworsesignal מוחקת מהרשימה סינגלים חלשים מהרשימה.

WriteKML כותבת את הקובץ ע"פ פורמט KML.

ReadCSV קוראת את הCSV המתקבל ומכניסה את הנתונים למערך דינאמי שעליו יעשו הבדיקות הבאות

Program פותחת את החלון של פתיחת הקובץ ומשם קוראת לפונקציות readcsv ו writeKML

Actionpreformed בונה הגדרה לinterface ComperDataWIFI וקוראת לפונק' program

**Interface ComperDataWIFI**

אינטרפייס להשוואות. יצרנו בסיסים משותפים להשוואות עם החזרים.

compareמקבל שני ארגומנטים ומחזיר מספר שלם:

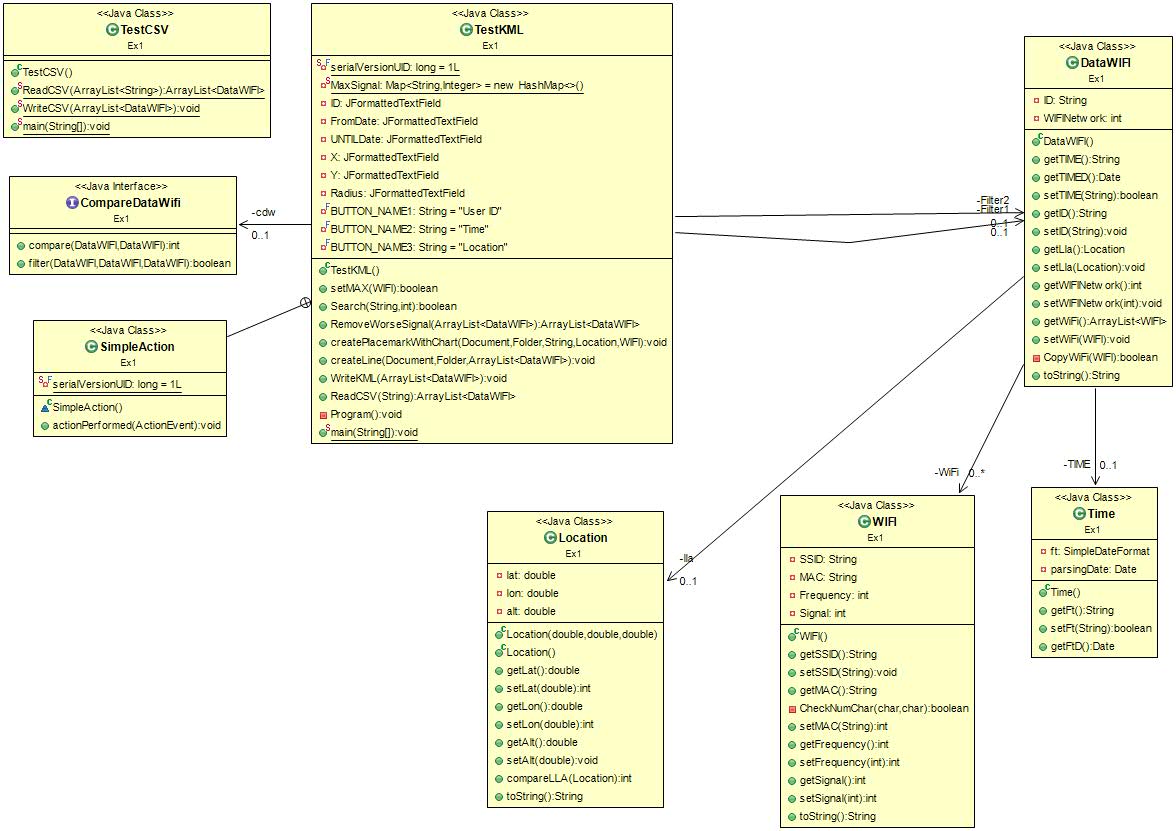
0 – באשר שני הארגומנטים זהיים, מינוס 1 כאשר הארגומנט הראשון גדול, 1 כאשר הארגומנט השני גדול יותר.

filterמקבל שלושה ארגומנטים ומחזיר משתנה לוגי:

-trueאם ארגומנט ראשון נמצאה בין ארגומנטים 2 ו.3במקרה של ID אםארומט ראשון שווה לשני.

-falseאם ארגומנט לא נמצאה בתחום ו לא שווה לארגומנט שני.

פירוט הספריות במחלקות והממשק:



**Interface ComperDataWIFI**

-גישה לספריית java util comparator

פעולות :

compare

מקבלת 2 פרמטרים ומשווה.

filter

מקבלת 3 פרמטרים ומשווה.

**DataWIFI**

גישה לספריית java util.\*

משתנים:

ID,WIFINetwork,lla(location),TIME(time),WiFi(WIFI)

פונק:

בנאי ריק

Getterand setter: ID,WIFINetwork,lla(location),TIME(time),WiFi(WIFI)

CopyWiFi

toString

**Location**

אין בספריות

משתנים:

lat,lon,alt

פונק:

בנאים location

אחד אחד מאופס ואחד עם קבלת נתונים

Getter and setter:lat,lon,alt

ToString

compareLLA - משוואה בין שני מיקומים מי הכי קרוב לכיוון דרום מזרח ומי הכי גבוה.

מחזירה 0, 1 ומינוס 1 לפי הגדרת השוואה כללית שרשומה למעלה

**Time**

גישה לספריות:

Java util.\*

Java text.\*

ליצירת פורמט זמן

משתנים:

Ft,parsingDate

פונק:

בנאי Time

Getter and setter for ft, parsingDate

**WIFI**

**אין ספריות**

מכילה משתנים:

SSID,MAC,Frequncy,Signal

פונק:

בנאי WIFI

Getter and setter: SSID,MAC,Frequncy,Signal

toString

cheakNumChar – מקבלת 2 משתנים מסוג CHAR ומחזירה אמת אם שניהם בין אפס ל9 או בין A לF (בודק שהנתון כתוב באקסדצינאמי)

**TestCSV**

גישה לספריות

java.io.\*

java.util.\*

javax.swing.\*

javax.swing.filechooser.FileNameExtensionFilter

com.csvreader.CsvReader

com.csvreader.CsvWriter

פונק:

Main

ReadCSV

WriteCSV

**TestKML**

גישה לספריות:

java.awt.\*

java.awt.event.\*

java.io.\*

java.text.DateFormat

java.text.DecimalFormat

java.text.NumberFormat

java.text.ParseException

java.text.SimpleDateFormat

java.util.\*

java.util.List

javax.swing.\*

javax.swing.filechooser.FileNameExtensionFilter

javax.swing.text.DateFormatter

javax.swing.text.NumberFormatter

com.csvreader.CsvReader

de.micromata.opengis.kml.v\_2\_2\_0.AltitudeMode

de.micromata.opengis.kml.v\_2\_2\_0.Coordinate

de.micromata.opengis.kml.v\_2\_2\_0.Document

de.micromata.opengis.kml.v\_2\_2\_0.Folder

de.micromata.opengis.kml.v\_2\_2\_0.Kml

de.micromata.opengis.kml.v\_2\_2\_0.LineStyle

de.micromata.opengis.kml.v\_2\_2\_0.Placemark

de.micromata.opengis.kml.v\_2\_2\_0.Style

de.micromata.opengis.kml.v\_2\_2\_0.Icon

משתנים:

***serialVersionUID***

*MaxSignal*

cdw

ID

FromDate UNTILDate

X1 Y1

X2 Y2

Filter1 Filter2

BUTTON\_NAME1 BUTTON\_NAME2 BUTTON\_NAME3

פונק:

Main

קלאס פנימי:

SimpleAction

משתנים:

SerialVersionUID

פונק:

בנאי SimpleAction

שאר הפונק' רשומות למעלה בתיאור הקבצים

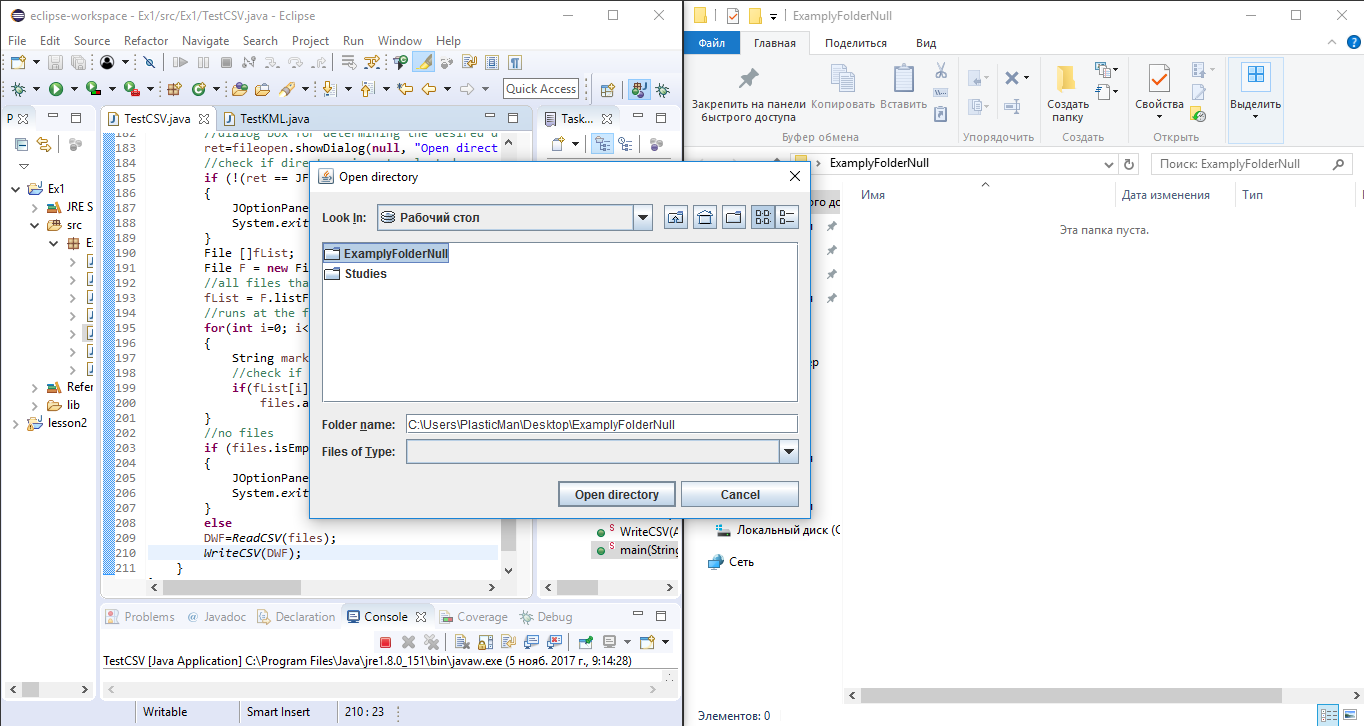
**כמו כן נכתבו** Unit Testing**עבור כל** class :

CompareDataWifiTest, DataWIFITest, LocationTest, TestCSVTest, TestKMLTest, TimeTest and WIFITest.

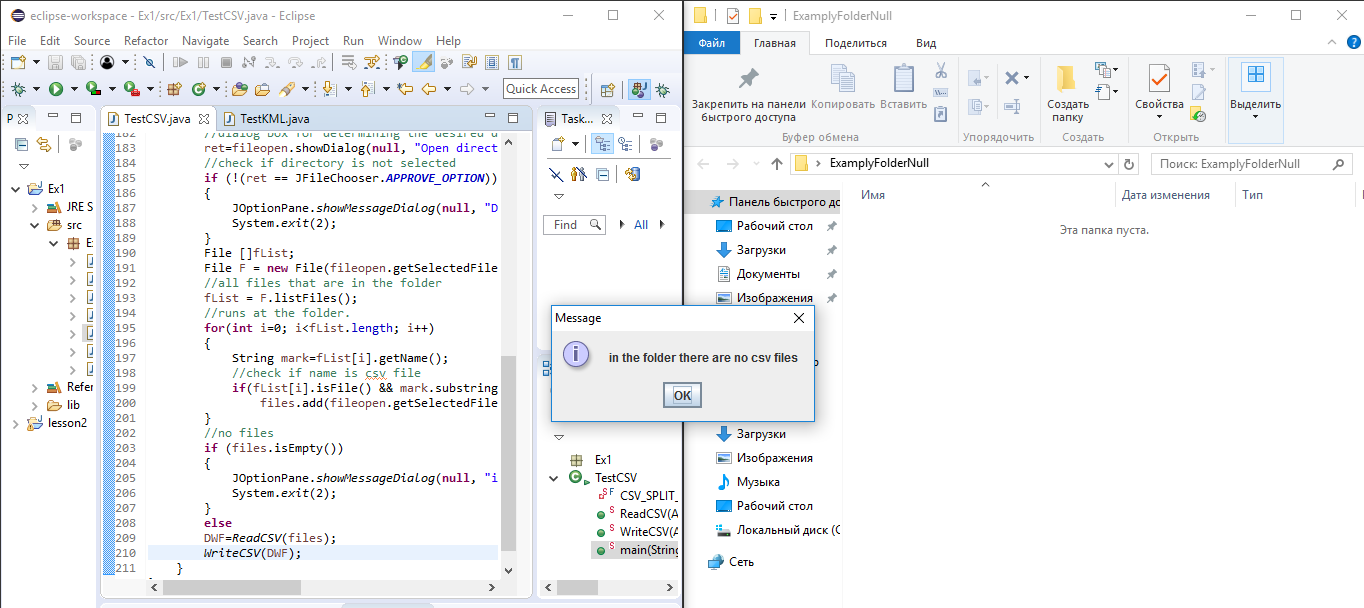
דוגמאות לבדיקת הקוד, והניסוי שעשינו:

**TestCSV**

בחירת תקייה ריקה:

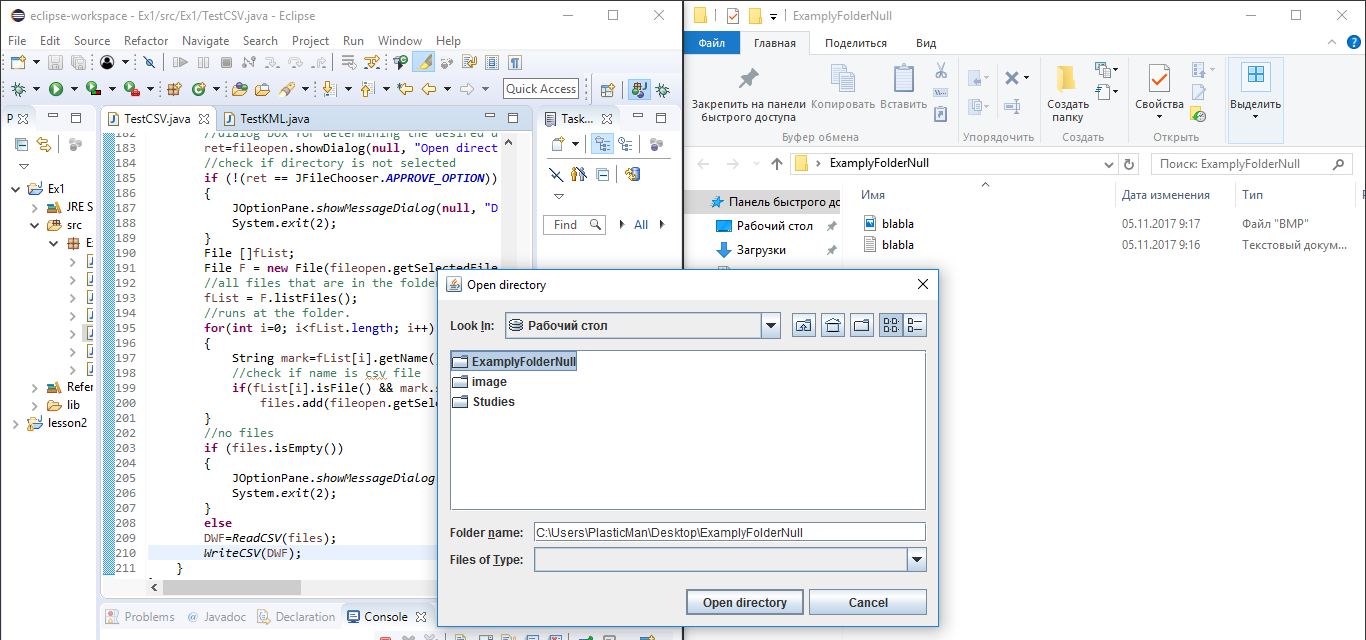


תוצאה:

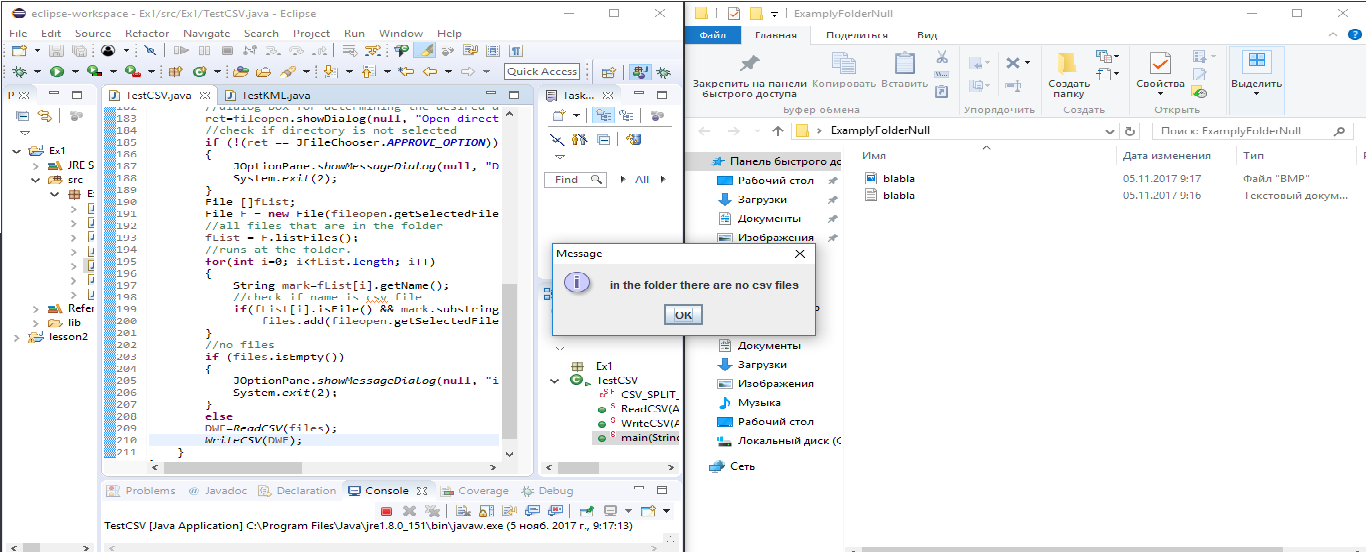


הרצאת התוכנית מסתיימת

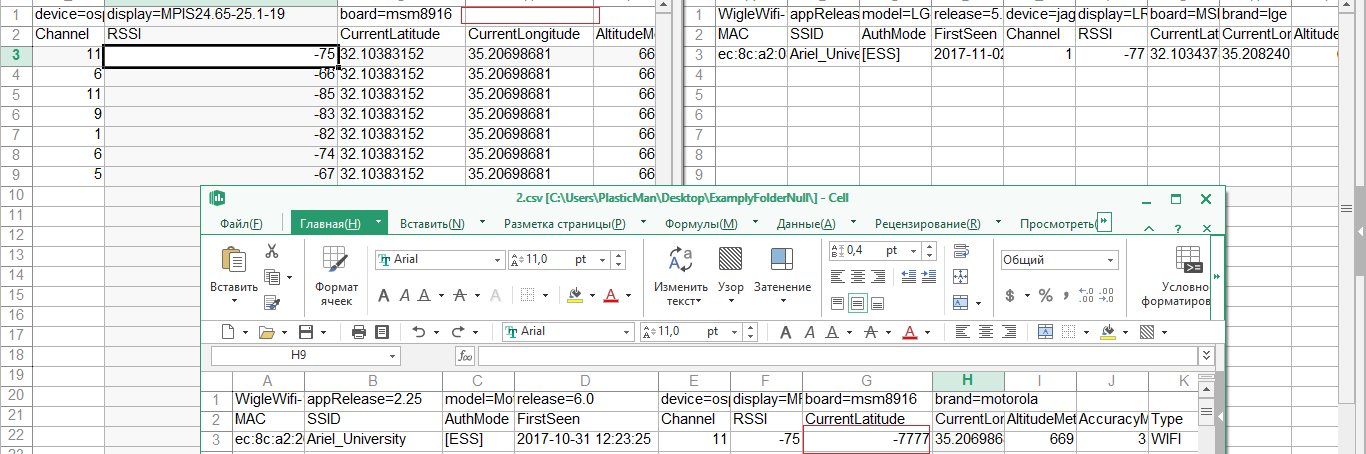
בדיקת תקיה ללא קבצי CSV

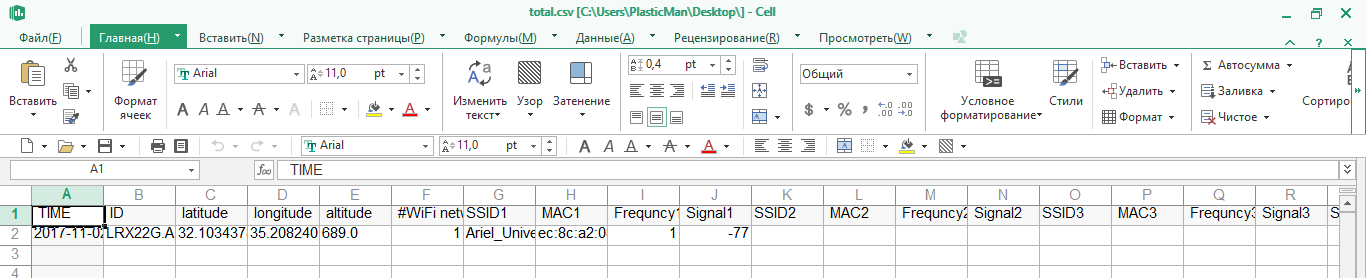


תוצאה:

הרצאת התוכנית מסתיימת.

בדיקת תקינות קבצי CSV ותוכנן:

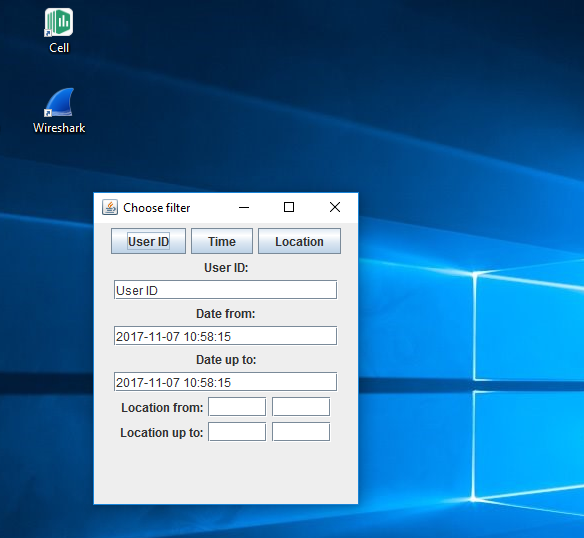


תוצאה:

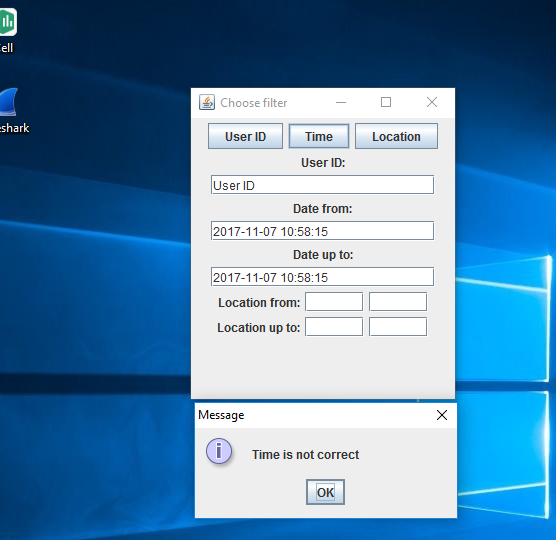
רק הקובץ התקין נכנס לקובץ החדש (בתמונה הראשונה מסומנים האדום הבעיות (מיקום לא תקין וחוסר בשם עמודה)

**TestKML**

בחירת סינון:

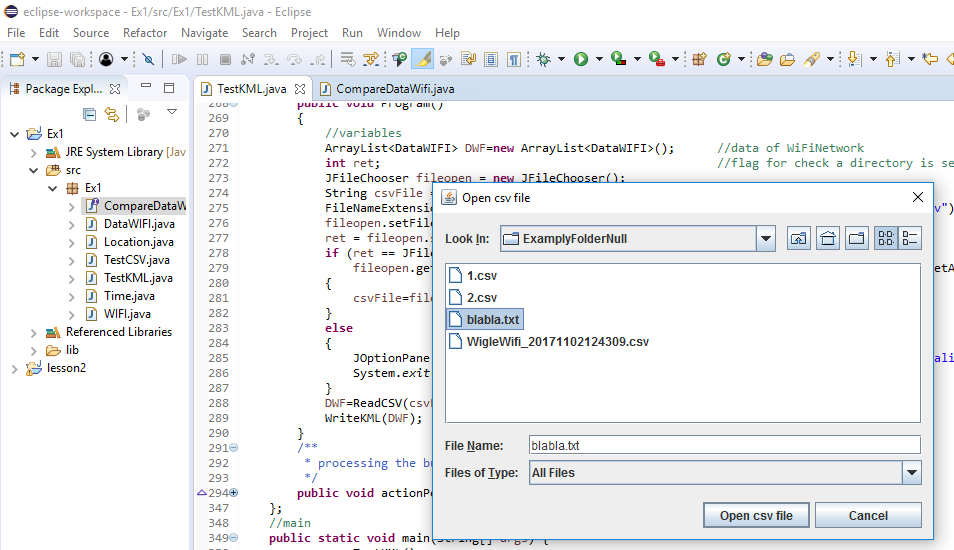


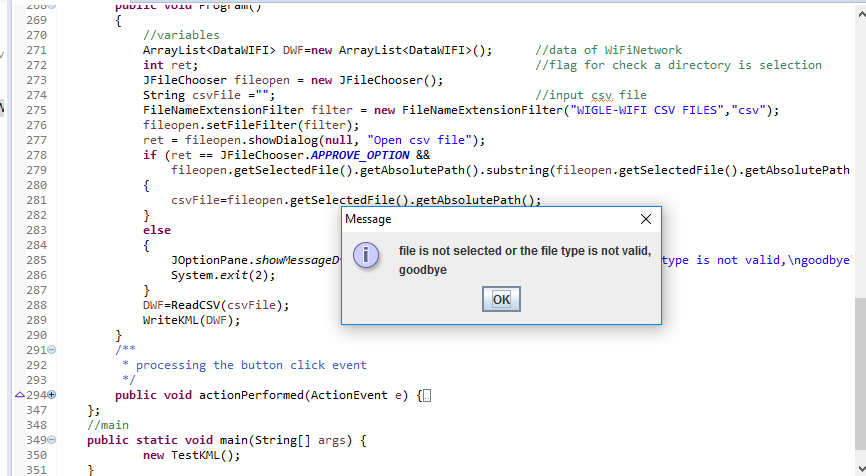
אם לא נבחרת שיטת סינון, התוכנית מסתיימת.



אם הכנסתה נתונים לא נכונים, הוצג הודעה ונתן לשנות את הנתונים.

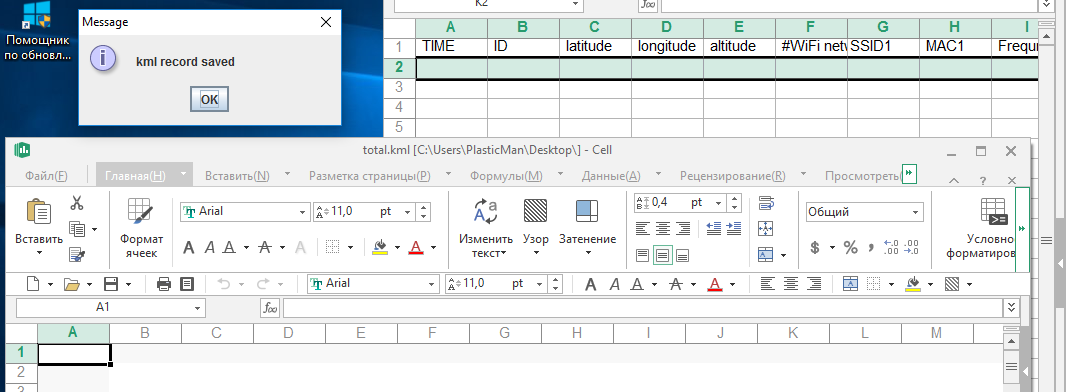
בדיקה: לא בחרנו קובץ CSV (נסיון פתיחת קובץ אחר)



תוצאה: 

התוכנית מסיימת את ההרצה.

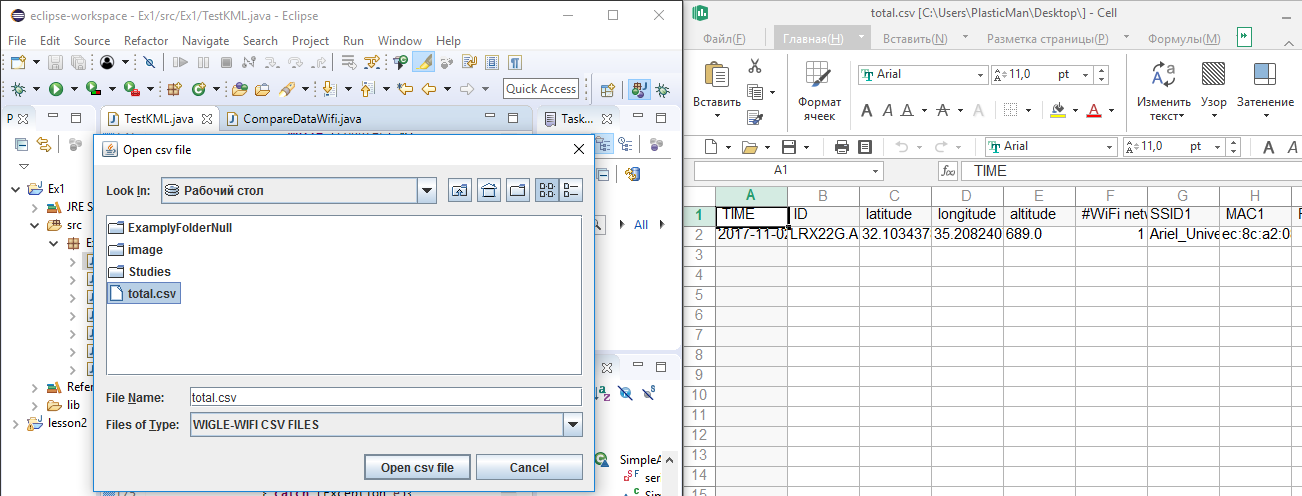
בדיקת קובץ CSV ריק

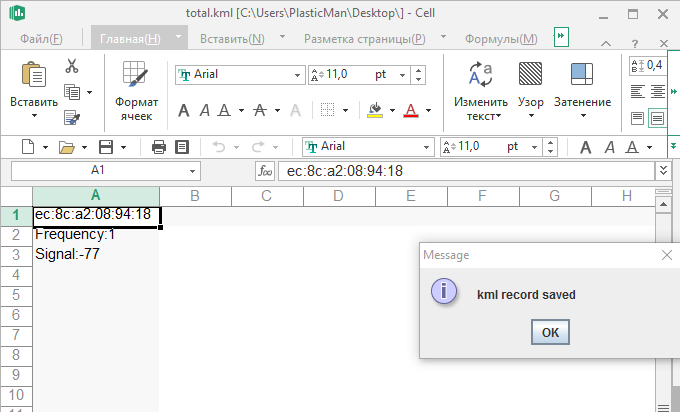


תוצאה:

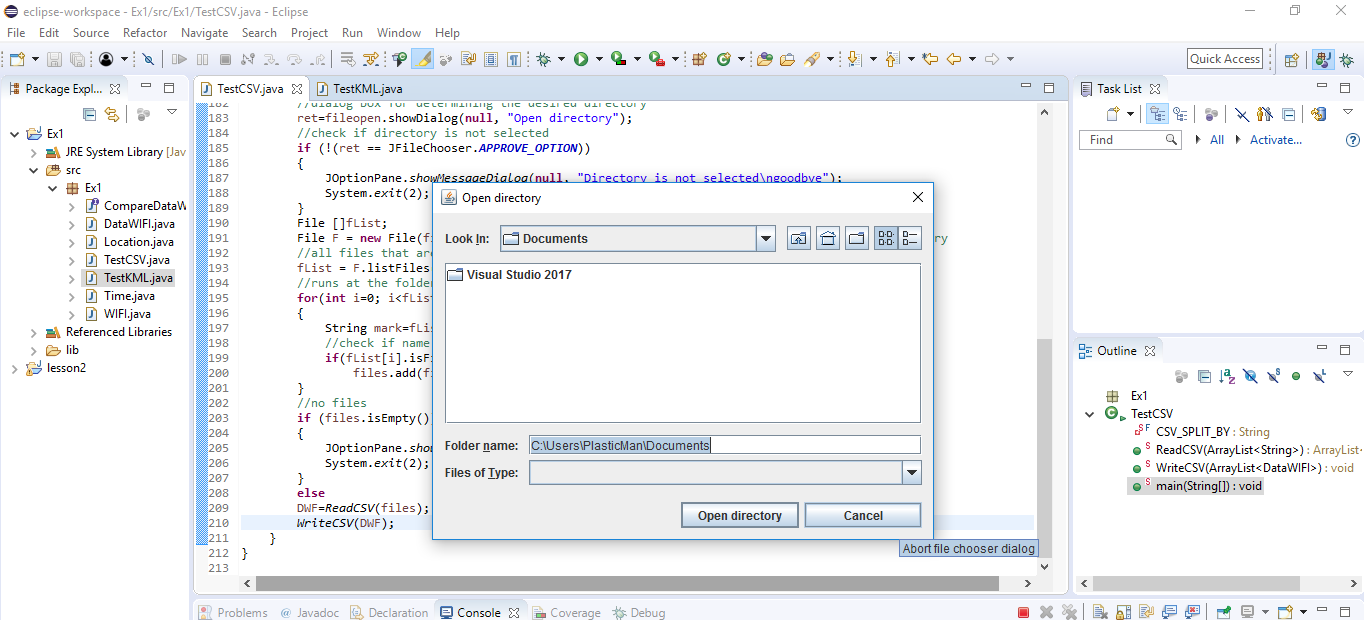
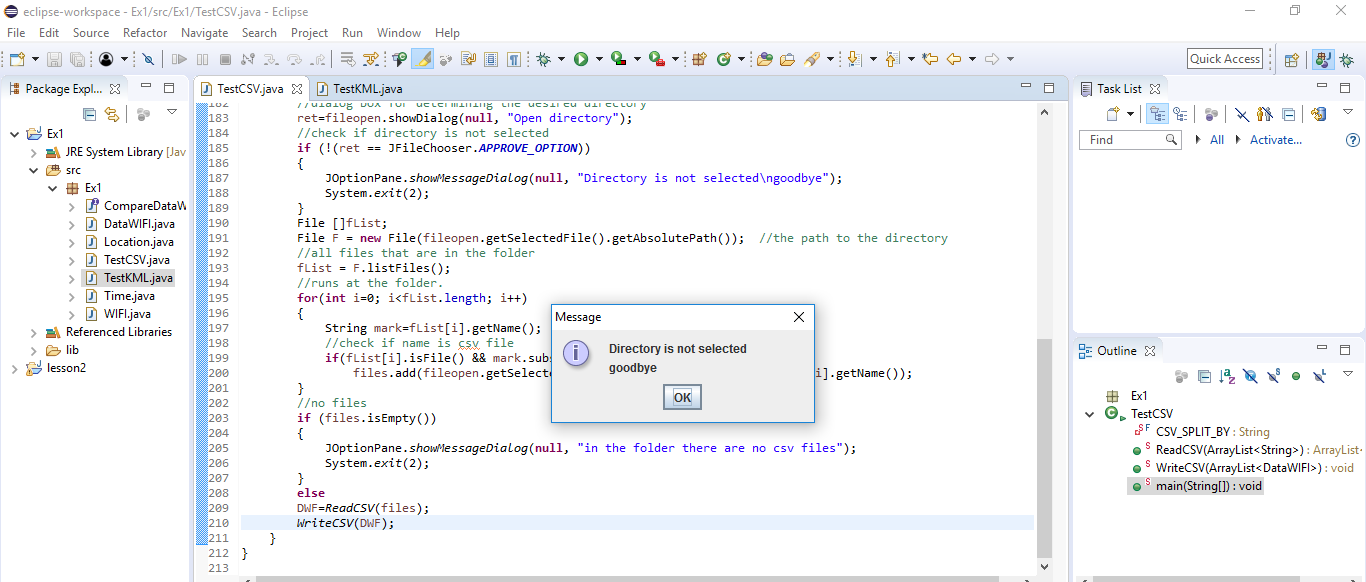
קובץ KML ריק.

בדיקת קובץ CSV תקין:



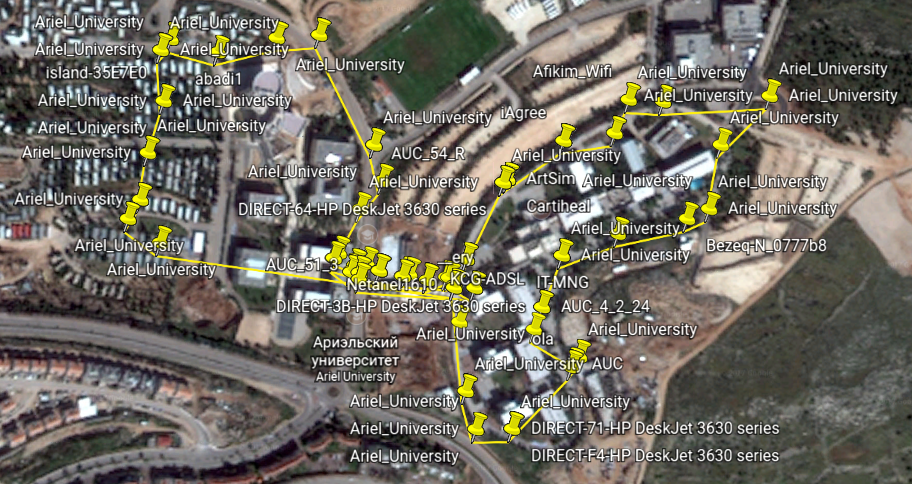
תוצאה:

הנתונים נכנסו ויצרו קובץ KML

בדיקה :אי בחירת תקייה או קובץ:תוצאה:סיום ריצה

עיבוד קובץ KML שנוצר בgoogle earth

נקודות WIFI בקמפוס:



תיאור המערכת (חלק ב) - (EX2):

אלוגוריתם אחד:

בהינתן מזהה ויפי ע"י כתובת מאק נשערך את מיקומו בעזרת המידע הקיים במערכת

נעשה זאת ע"י לקיחת כמה דגימות הכי חזקות של המאק המבוקש ונחשב ממוצע משוקלל שאותו נחזיר ככתובת משוערכת

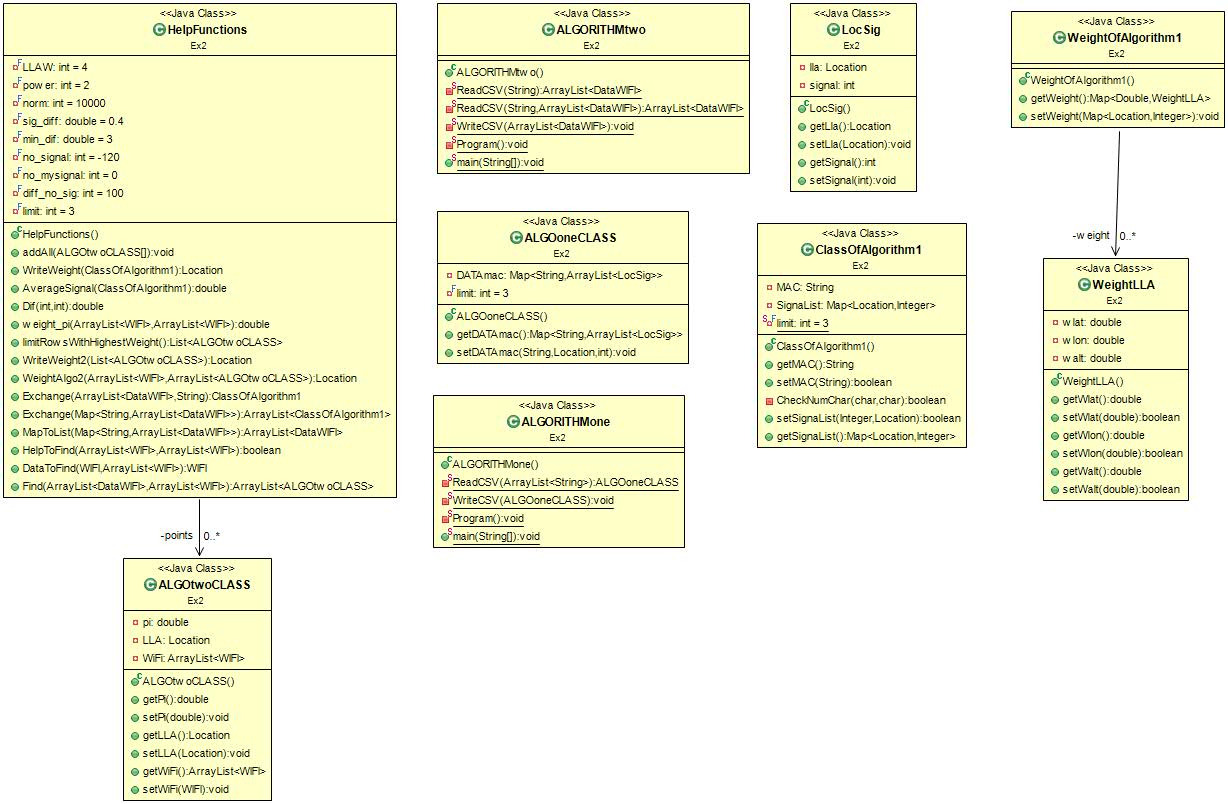
אלגוריתם שני:

בהינתן מספר דגימות ויפי ועוצמת סיגנל שלהם

נשערך את מיקום המשתמש. שתמש באלגוריתם אחד בשילוב עם שיטה לבדיקת מידת ההתאמה של כל אחת מהדגימות בקלט שלנו

לכן יצרנו מחלקות שיבצעו עבודה זאת

UML:



פירוט תפקידי המחלקות:

ALGORITHMone

קורא קבצי CSV ויוצר קובץ CSV חדש ע"פ אלגוריתם 1 (לא מבצע חישובים)

ALGORITHMtwo

קורא קובץ של הבסיס נתונים (DATABASE) וקובץ של נתונים חסרים ומחזיר קובץ CSV חדש אחרי חישובים

ALGOoneCLASS

מחלקה לאלגוריתם 1 שמחזיקה נתונים שצריך לצורך החישוב

ALGOtwoCLASS

מחלקה לאלגוריתם 2 שמחזיקה נתונים לצורך החישוב

ClassOfAlgorithm1

מחזיק נתונים לכתובת MAC אחד

HelpFunctions

מחילה פונקציות חישוב לאלגוריתמים אחד ושתיים

LocSig

מחלקה נוספת לצורך של משקל האות באלגוריתם 2

WeightLLA

מחזיקה נתונים שך לוקישן אחרי חישוב משקלל משוקלל

WeighOfAlgorithm1

עושה את חישוב המשקל של אלגוריתם אחד

את כל בדיקות ה- JUNIT הכנסנו לחבילה נפרדת בשם UTest

חבילה זאת מחילה את הבדיקות לEX1 ול EX2