מבוא לתכנות מונחה עצמים-מטלות 4+3+2+1+0

מגישים: בן הורן 208569848 נועה חדד ת.ז. 305134694

הקדמה

בחרנו באפליקציית WigGle על מנת לאסוף מידע על נקודות wifi ברחבי האוניברסיטה. אפליקציה זו סיפקה לנו את כל הנתונים הנדרשים למטלה, וכמו כן אפשרה לנו לייבא את המידע לקובצי csv.

עיבוד המידע נעשה באמצעות התכנית שכתבנו:

מבנה התכנית-

1) מחלקות main, Wifi, combiningData, processingData, Weight, MAC, algorithms, frame1, connectSQL.
2) ממשק filter .
3) מחלקות בדיקותWifiTest , combiningDataTest, processingDataTest, algorithmsTest.

מחלקת Wifi-

יצרנו טיפוס מסוג Wifi, אשר מכיל את המשתנים הבאים: MAC, SSID, ALT, LON, LAT, ID, Time, signal, Frequency.

למחלקה יש מספר מתודות, כולן ציבוריות:

1) בנאי אשר מקבל את הנתונים על נקודת ה-wifi ומכניס אותם לתוך המשתנים.
2) מתודות (...Time/IT/LAT/) get- המחזירות את הנתון המבוקש.
3) מתודת (Wifi other)equals -המשווה בין שתי נקודות wifi, ומחזירה אמת אם הן זהות, אחרת מחזירה שקר.
4) מתודה סטטיק (Correct(Time,ID,LAT,.. המקבלת נתונים על נקודת wifi ובודקת את תקינותם. מחזירה אמת אם הנתונים תקינים, ושקר אחרת.
5) מתודת ()getCoordinates- המחזירה מחרוזת של נקודות LAT,LON.
6) מתודת (Document doc)kmlGenerator- מקבלת אובייקט Document ומוסיפה לו ייצוג נקודת הוואיפיי בפורמט kml

מחלקת combiningData -

המחלקה הזו בנויה כולה מפונקציות.

במחלקה זו –

1) קוראים מקבצי ה csv (שיובאו מהאפליקציה) את המידע על נקודות הwifi.
2) בודקים את תקינות המידע.
3) שומרים את המידע במבנה נתונים, וזאת לשם כתיבתו לקובץ חדש בפורמט הנדרש.

הפונקציות:

1) פונקציה פרטית סטטית `findCsvFiles (String folderPath)`- הפונקציה מקבלת את מיקום התיקייה שבא אמורים להימצא קבצי הcsv. מחזירה מערך של קבצים במידה וקיימים קבצים כאלה בתיקייה, ו- null אחרת.

2) פונקציה פרטית סטטית getFileFormat(File f)- פונקציית עזר לפונקציה הקודמת. הפונקציה מקבלת קובץ ומחזירה את סוגו.

3) פונקציה פרטית סטטית LinkedList<Wifi>fileToList(File[] listOfFiles)- הפונקציה מקבלת מערך של קבצי csv, ומחזירה רשימה מקושרת שכל איבר בה הוא מטיפוס Wifi ומכיל מידע על נקודת Wifi בודדת.

4) פונקציה פרטית סטטית legit(String str) – פונקציית עזר לפונקציה בסעיף הקודם. הפונקציה מקבלת מחרוזת של שורה מקובץ הcsv, ובודקת אם היא בעלת נתונים חלקיים/מיותרים. באמצעות פונקציה זו מסננים נקודות wifi . הפונקציה תחזיר אמת אם המידע תקין, ושקר אחרת.

5) פונקציה פרטית סטטית listToCSV(LinkedList<Wifi>wifis,String CSVpath)- הפונקציה מקבלת את הרשימה המקושרת עם המידע על נקודות הWifi , וכותבת את המידע לקובץ csv חדש בפורמט הנדרש. הקובץ נשמר תחת CSVpath.

6) פונקציה ציבורית סטטית toCSV(String folderPath, String CSVpath) – הפונקציה מקבלת כתובת של תיקייה. באמצעות הפונקציות האחרות במחלקה: היא בודקת אם יש בה קבצי csv, בודקת את תקינות המידע בקבצים, ומארגנת אותו לתוך קובץ csv חדש שנשמר בכתובת השנייה שקיבלה. הפונקציה מדפיסה הודעת אישור על הצלחת יצירת הקובץ החדש, או לחלופין הודעת שגיאה אם נכשל .

7) פונקציה ציבורית סטטית LinkedList <Wifi> list(String folderPath)- פונקציית עזר למחלקה הקודמת. הפונקציה מקבלת כתובת של תיקייה, ומחזירה רשימה מקושרת של נקודות wifi תקינות. פונקציה זו נבנתה על מנת שנוכל לבדוק את נכונות רשימת הנקודות שנתקבלו מהפונקציה הפרטית "fileToList " .

<u>מחלקת processingData</u> -

המחלקה הזו גם בנויה כולה מפונקציות.

במחלקה זו-

1) סינון המידע שבקובץ ה csv- בוחרים לפי איזה נתון (זמן/מקום/מזהה) רוצים לסנן ומהו המידע שרוצים למצוא בנתון זה.
2) ארגון המידע-מוצאים את כל הנקודות בעלות אותו MAC, משאירים נקודה אחת עם הסיגנל הכי חזק, ולה מצמידים מרכז כובד משוקלל.
3) כתיבת המידע לקובץ kml.

הפונקציות:

1) פונקציה פרטית סטטית "csvtoList"- הפונקציה מקבלת את כתובת קובץ הcsv המאוחד, הפרמטר/ים לסינון, המידע הנדרש מהם/ממנו ופעולת/פעולות הסינון הנדרשות. הפונקציה מחזירה רשימה מקושרת עם נקודות Wifi שעונות לדרישות הסנן. (בנוסף אליה, יש פונקציה ציבורית סטטית "list", שקוראת לה ובכך מאפשרת לנו לערוך בדיקות על התוצר שלה)

2) פונקציה בוליאנית "fit". הפונקציה מיישמת את חתימת הפונקציה שבממשק filter. הפונקציה היא למעשה פונקציית עזר למחלקה הקודמת. היא מקבלת מידע על נקודת wifi, הפרמטר לסינון והמידע הנדרש ממנו. מחזירה אמת אם הנקודה עונה לבקשת הסנן, ושקר אחרת.

3) פונקציה פרטית סטטית "organize"-הפונקציה מקבלת רשימה מקושרת של נקודות wifi ובודקת אם קיימות בה נקודות בעלות אותו MAC. הפונקציה מחזירה רשימה חדשה שבה לכל נקודה MAC יחידאי ולכל נקודה מוצמד מרכז כובד משוקלל. הפונקציה גם יוצרת קובץ csv שבו נמצאים נתוני הרשימה. (בנוסף אליה, יש פונקציה ציבורית סטטית "listOrganized", שקוראת לפונקציה "organize", ומאפשרת לבדוק את התוצר שלה)

4) פונקציה פרטית סטטית "kml" המקבלת רשימה מקושרת של נקודות WIFI וכתובת של קובץ KML. הפונקציה מחזירה אמת אם הצליחה לייצא קובץ KML , ושקר אחרת.

5) פונקציה ציבורית סטטית "CSVtoKML" המקבלת כתובת של קובץ CSV, כתובת של קובץ KML, פרמטר לסינון ,המידע הנדרש ממנו וכתובת של קובץ מרכז כובד משוקלל. הפונקציה מחזירה אמת אם יוצא קובץ KML, ושקר אחרת.

<u>מחלקות Weight, MAC-</u> המחלקה MAC מכילה מידע על mac ומרכז כובד משוקלל. המחלקה Weight מכילה מידע על משקל (דמיון בנקודה מסוימת) ומיקום. שתי המחלקות נבנו כדי לעזור באחסון נתונים לשם מימוש אלגוריתמים 1 ו-2.

<u>מחלקת algorithms-</u>

המחלקה מכילה מימוש של שני אלגוריתמים למציאת מיקום:

1) פונקציה פרטית סטטית "findPlace1"-פונקציית עזר למחלקה הקודמת, היא מקבלת כתובת של קובץ הcsv המאוחד וכתובת mac, ומחזירה טיפוס ממחלקה MAC המכיל פרטי MAC ומרכז כובד משוקלל. (בנוסף אליה, יש פונקציה ציבורית סטטית "findPlaceAlgorithm1" שקוראת לה, ומאפשרת לנו לבדוק את הפלט שלה).

2) פונקציה פרטית סטטית בשם findPlace2, המקבלת קובץ csv מאוחד עם נתוני GPS, קובץ ללא נתוני GPS, כתובת אליה תכתוב קובץ חדש שבו יוצמד לכל שורת מידע מיקום משוערך. הפונקציה קוראת את שני הקבצים, מחשבת מיקום משוערך, וכותבת את המידע לקובץ csv חדש. (בנוסף אליה, יש פונקציה ציבורית סטטית "findPlaceAlgorithm2" שקוראת לה, ומאפשרת לנו לבדוק את הפלט שלה).

פונקציה פרטית סטטית בשם findPlace2, המקבלת קובץ csv מאוחד עם נתוני GPS, שורה ללא נתוני GPS/עד 3 זוגות של mac ו-signal, ומחזירה מחרוזת עם נתוני מיקום משוערך (בנוסף אליה, יש פונקציה ציבורית סטטית "findPlaceAlgorithm2" שקוראת לה, ומאפשרת לנו לבדוק את הפלט שלה).

<u>מחלקת connectSQL–</u>
יצרנו טיפוס מסוג connectSQL, אשר מכיל את המשתנים הבאים: ip, url, user, password, table.

יש למחלקה שתי פונקציות:

()getData- קוראת את הנתונים מהטבלה, מכניסה אותם לרשימה מקושרת של נקודות Wifi. ומחזירה את הרשימה.

()lastModified-מחזירה מחרוזת עם התאריך בו הטבלה השתנתה לאחרונה.

מחלקת frame1- המחלקה שמיישמת ממשק משתמש גרפי, שבאמצעותו המשתמש יכול לבחור את המקורות לבניית מבנה הנתונים, מאפייני מבני הנתונים , ויצוא המבנה המבוקש לקבצים. נוסף לכך, המשתמש יכול להפעיל אלגוריתמים לשערוך מקום.

מחלקת Main- מריצה במקביל את הממשק הגרפי למשתמש - אשר מאפשר למשתמש לבנות את מבני הנתונים, ו-thread אשר תפקידו לעדכן את מבנה הנתונים בהתאם לשינויים בתיקיות/קבצים/טבלאות שבונים אותו.
זוהי למעשה התכנית שאותה מריצים, והיא מאגדת תחתיה את הפונקציונאליות של שאר המחלקות.

ממשק Filter- מכיל חתימה של פונקציה בוליאנית בשם "fit". הפונקציה מקבלת טיפוס מסוג Wifi, מחרוזת שמייצגת את הנתון שלפיו רוצים לסנן ומחרוזת המייצגת את המידע המבוקש.

מחלקות בדיקות

לכל אחת מהמחלקות Wifi, combiningData , processingData agorithms נבנתה גם מחלקת בדיקות. העיקרון המנחה היה לבדוק עבור מקרים מסוימים/מקרי קצה, אם מה שצפינו שהפונקציה תעשה –זה אכן מה שהיא ביצעה בפועל.

כלי תוכנה

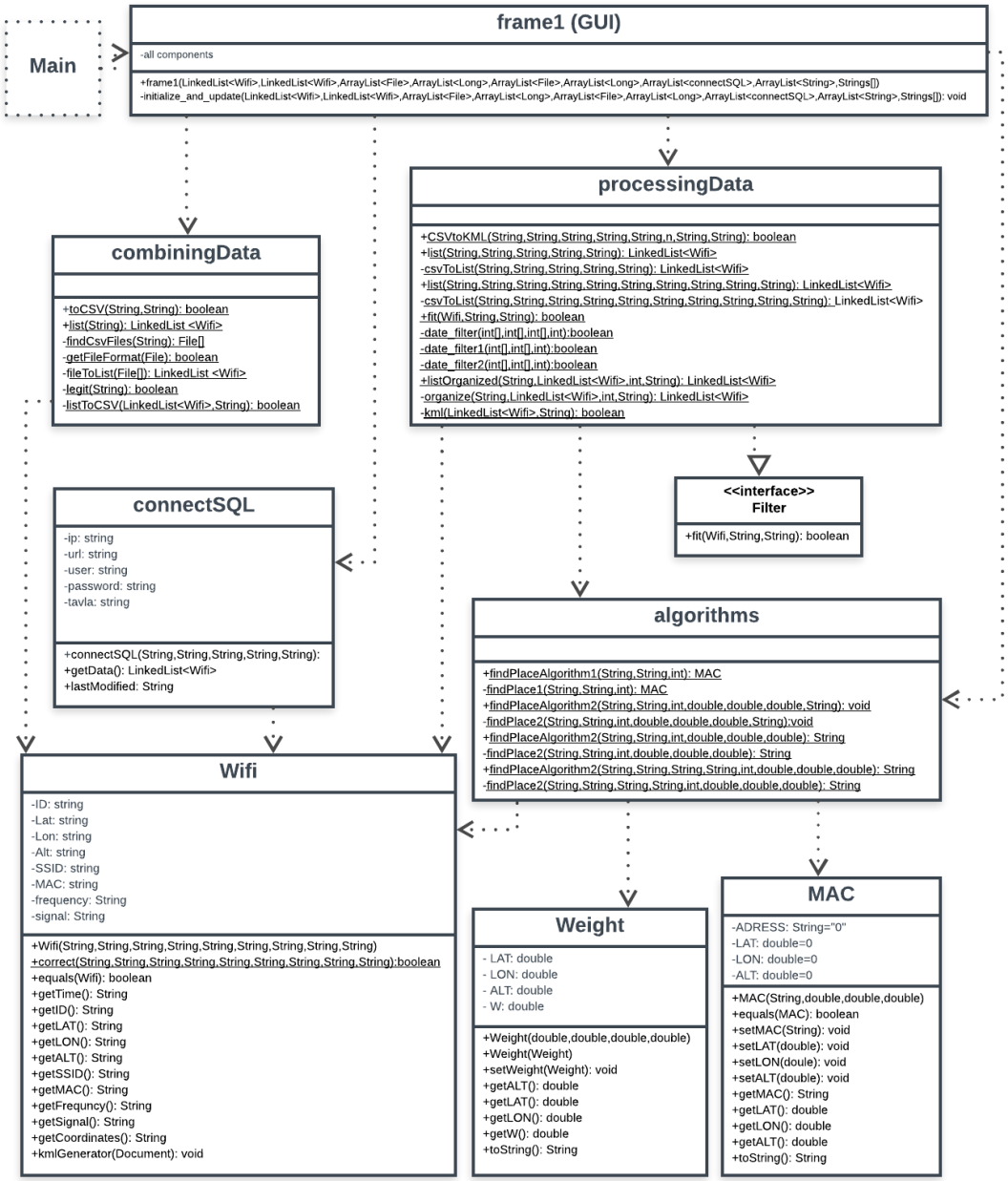
לצורך הפעלת התכנית השתמשו במספר כלי תוכנה:

יבוא java.io.File - על מנת לגשת לקבצים.
יבוא java.io.BufferedReader, java.io.FileReader - על מנת לקרוא את הקבצים.
יבוא java.io.FileNotFoundException, java.io.IOException- על מנת "לתפוס" חריגות בעת הרצת התוכנית.
יבוא java.util.LinkedList –על מנת להשתמש במבנה נתונים בדמות רשימה מקושרת.
יבוא org.junit.Assert.* - על מנת ליצור מחלקת בדיקות.
יבוא סיפריות JAK-java API kml המשמשת ליצירת קובץ kml לפי חוקי הפורמט. קובץ הkml מייצג נקודות על מפה, במקרה שלנו את נקודות הwifi. על מנת לייצא קובץ kml מרובה נקודות, בחבילת jak נכללת גם סיפריית Document, בה ניתן להוסיף מספר placemark , ולייצג גם יותר מנקודה בודדת. לכל placemark נוסף תיאור (Descriprion), בתיאור הנקודה צורף שם נקודת הוואיפיי, עוצמת הנקודה, הmac ושאר נתוני הוואיפיי. לכל placemark נוסף מיקום הנקודה בקורדינטות בעזרת פונקציית withCoordinates וחתימת זמן (תאריך ושעה) בעזרת SetTimeStamp.
יבוא java WindowBuilder לשם יצירת הGUI , יבוא java.awt.event.ActionListener , java.awt.event.ActionEvent לשם האזנה לאירועים בGUI, יבוא java.awt.EventQueue כדי לנהל את האירועים בGUI.
יבוא javax.swing.JFrame, javax.swing.JOptionPane javax.swing.JButton;, , javax.swing.JTextFiel על מנת ליצור את רכיבי הGUI.
יבוא java.io.FileInputStream, java.io.FileOutputStream, על מנת לקרוא ולכתוב לקובץ ser.*.
יבוא mysql-connector-java-5.1.45-bin על מנת לגשת לטבלאות ברשת.

שימוש במערכת בנייה gradle - המערכת למעשה מאפשרת לחבר את כל מרכיבי התוכנה, הן המקורות הפנימיים שלה והן המקורות החיצוניים שלה. באמצעות הרצת קובץ build שבו מוגדרת התכנית הראשית, מוגדרות הספריות החיצונית , המטלות ומה תלוי במה- המערכת מקמפלת את התכנית, עוברת על כל הטסטים שמרכיבים את התכנית ובודקת את תקינותם, ויוצרת קובץ jar עבור המשתמש.

נספחים

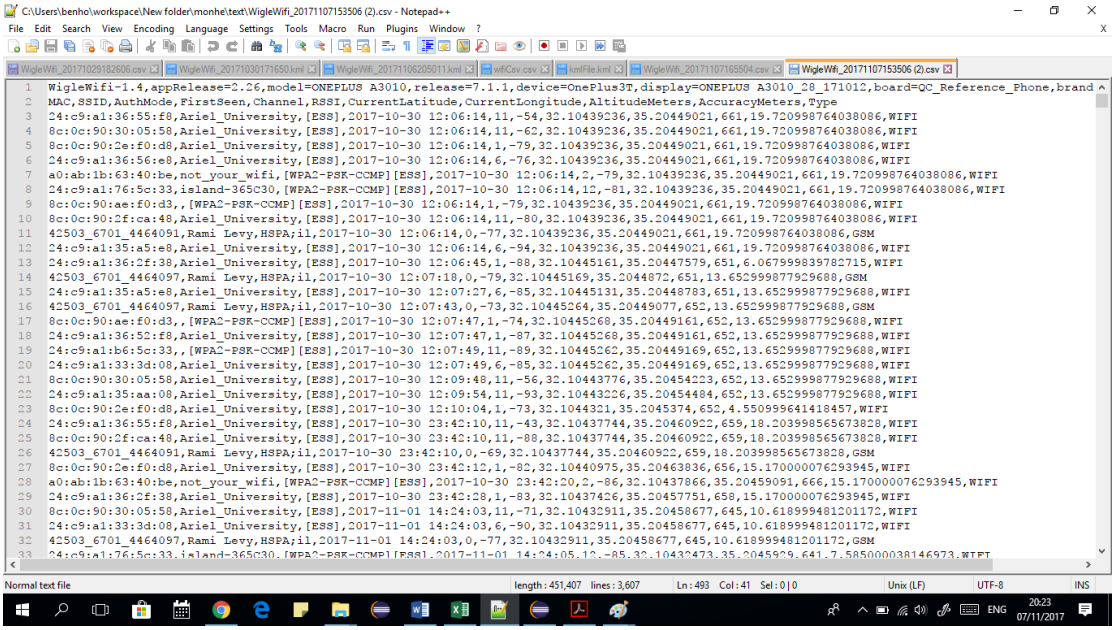
1. דיאגרמת מחלקות-



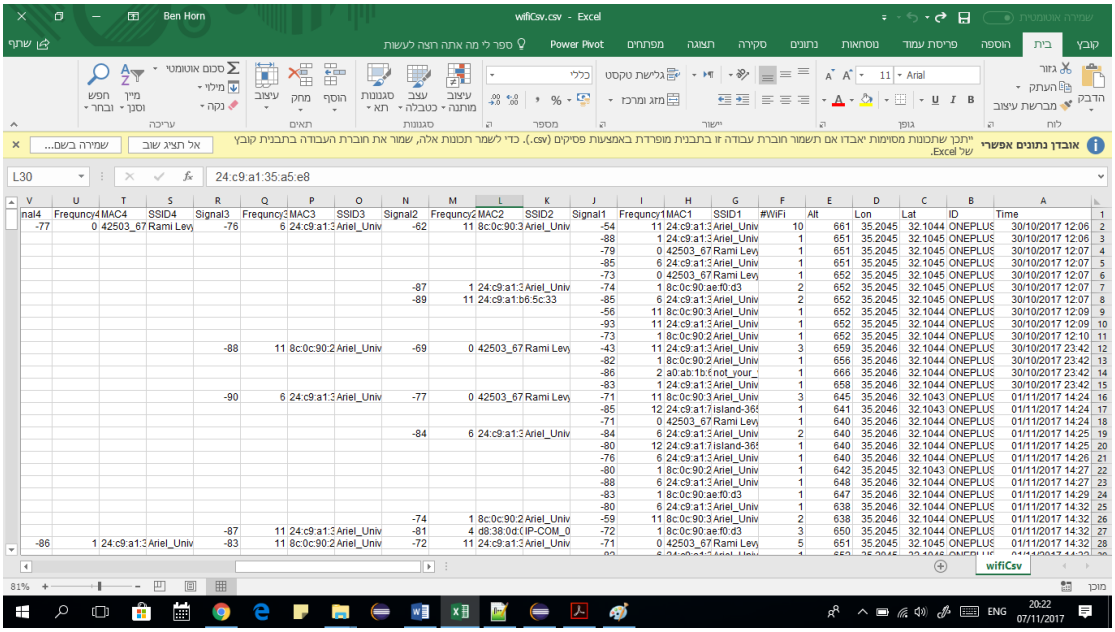
2. ניסוי שערכנו (מטלה 1+0)

בניסוי הדלקנו את אפליקציית קליטת נתוני הWifi במכשיר הטלפון למשך יום לימודים שלם- ממעונות הסטודנטים עד לכיתות הלימוד וחזרה. לאחר מכן ייצאנו לקובץ CSV והפעלנו את התוכנה. כך נראה הייצוא לקובץ KML על גבי מפה (לאחר הכנסת פילטר החיפוש: ID, OPPO3T - הID של המכשיר בו השתמשנו).

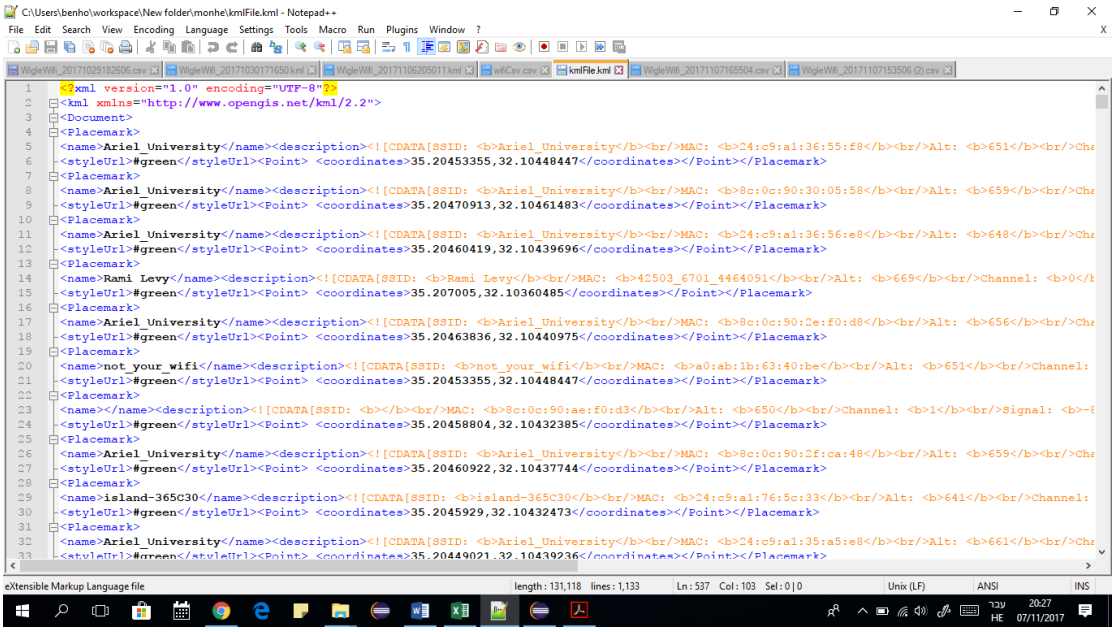
הקובץ שיובא מאפליקציית wigGLE שבמכשיר הפלאפון.



קובץ CSV שנוצר בעקבות הקריאה לפונקציה "combiningData.toCSV"

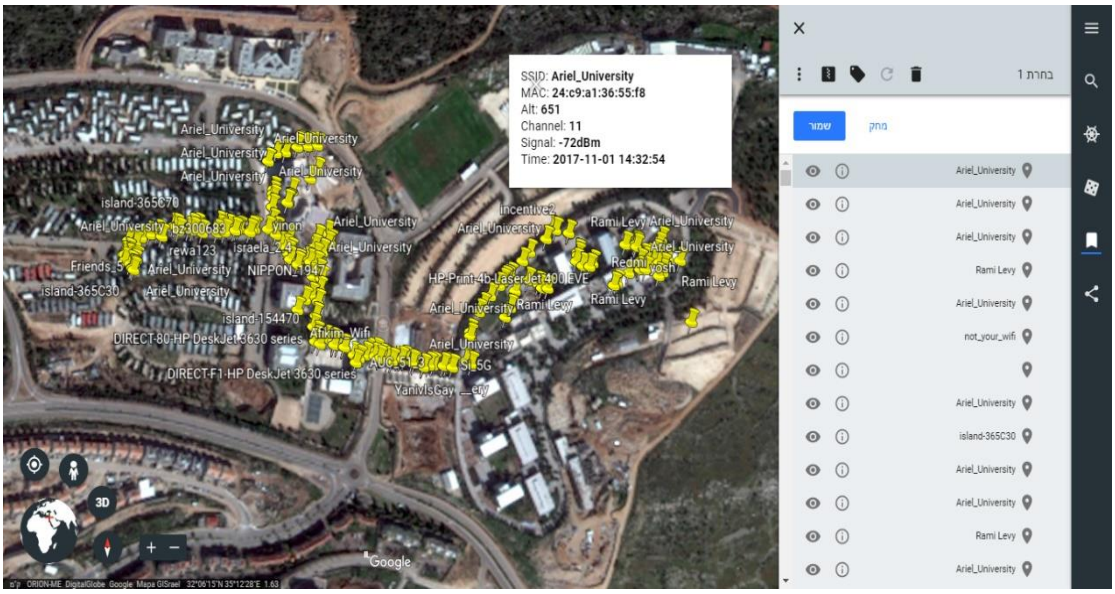


קובץ KML שנוצר בעקבות הקריאה לפונקציה "processingData.CSVtoKML" (הפונקציה סיננה נקודות WIFI ממכשירים אחרים) .



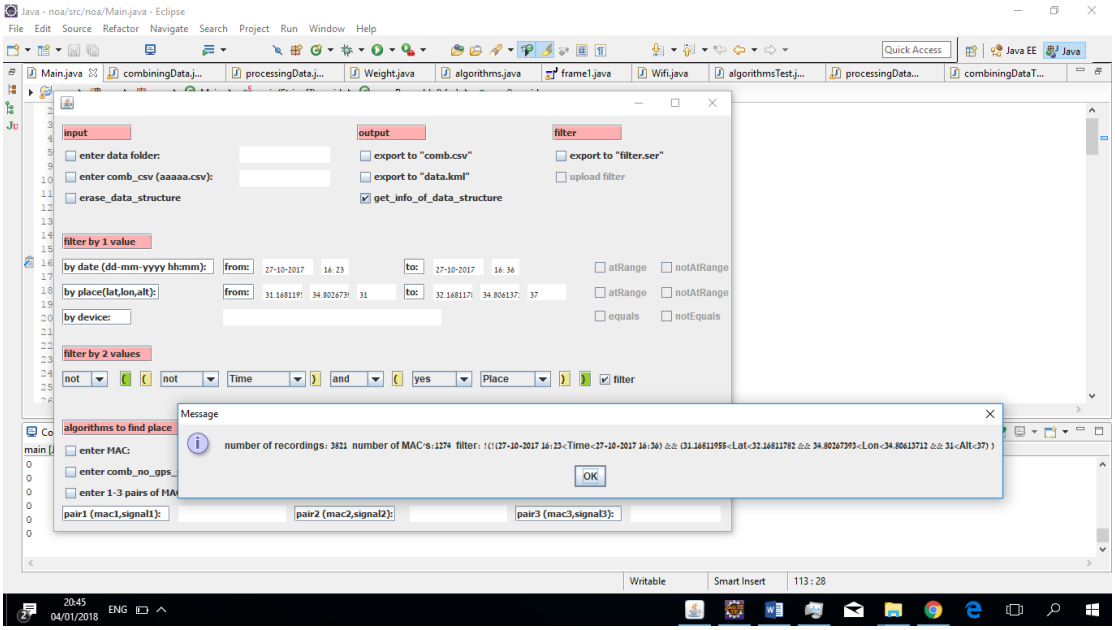
פתיחת קובץ KML בgoogle earth.

ניתן לראות את נקודות ה WIFI ברחבי האוניברסיטה ואת המידע עליהן.

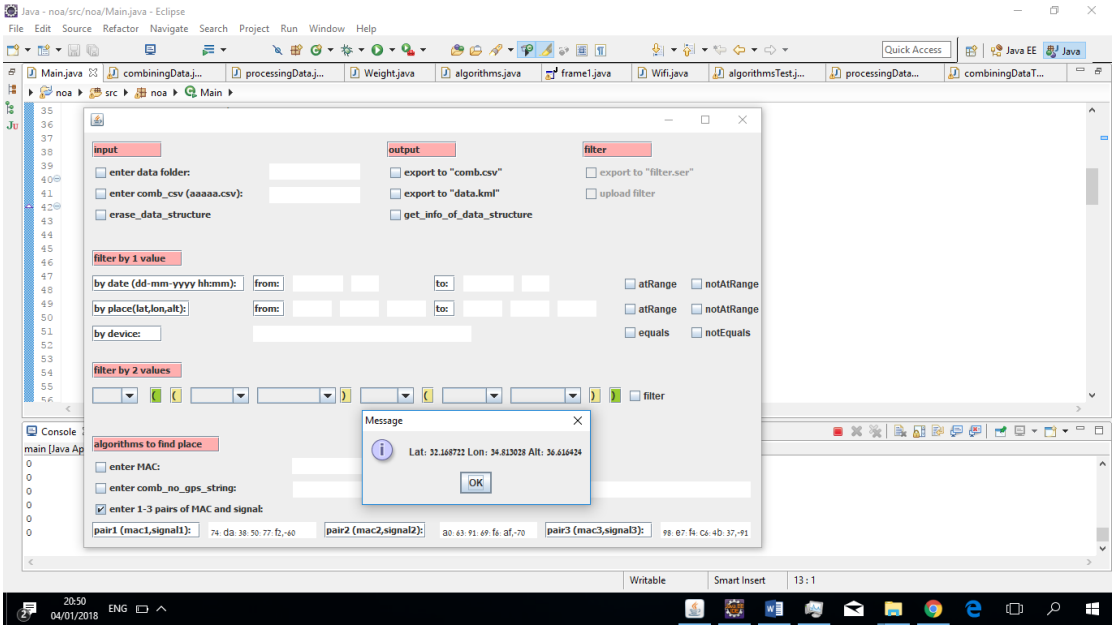


3. שימוש בממשק משתמש גרפי (מטלה 3)

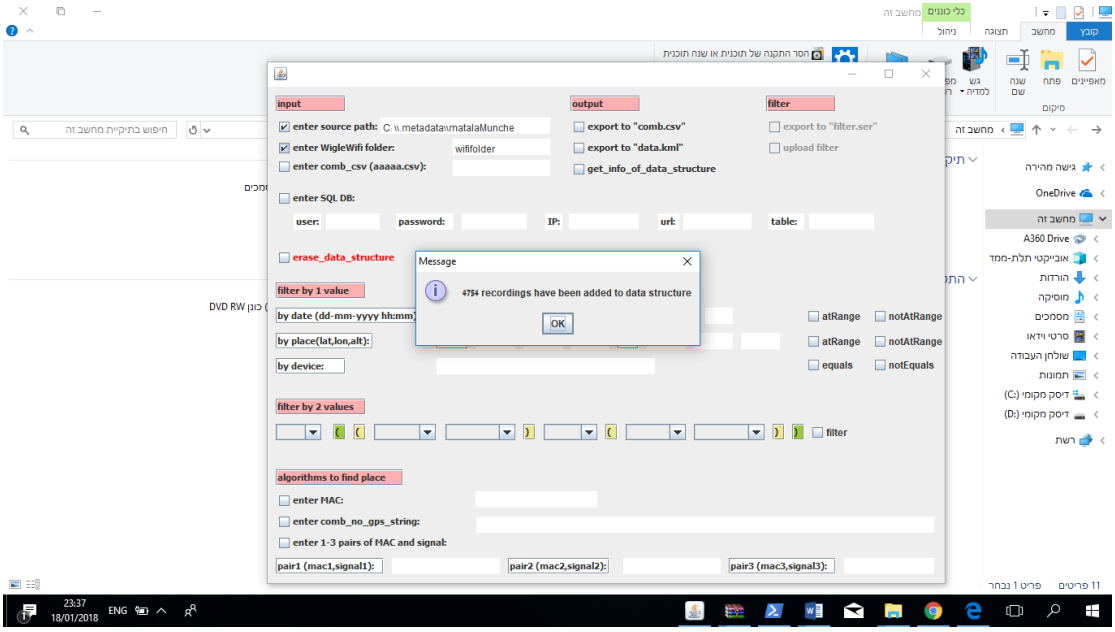
סינון לפי זמן ומקום, וקבלת מידע על מבנה הנתונים המסונן-



קבלת מיקום משוערך-



שימוש בממשק משתמש גרפי (מטלה 4):

ניתן לראות שכעת כל אחד יכול להשתמש באפליקציה ע"י הורדה של קובץ ה-jar שנמצא ב-gradle→build→libs→ project1-all.
במסגרת האפליקציה, המשתמש מכניס את הpath שבו נמצאים הנתונים שלו, מכניס את שמות התיקיות/הקבצים או טבלאות שהוא מעוניין שירכיבו את מבני הנתונים שלו, והוא יכול לסנן את הנתונים, לקבל מידע על הנתונים ולייצא את הנתונים.

ניתן לראות כי כעת הממשק לא תלוי במערכת eclipse-