**דו"ח מטלה 1**

מגישים: גלעד פוקס 203254370

מתן מעברי 305266652

קישור לGitHub: <https://github.com/igilfu/EX01>

/חבילת /Main

**מחלקת Main:**

המערכת מתחילה בפונקציה,main שם היא יוצרת את הנתיב להעלאת קבצי csv ואת הנתיב (כולל שם הקובץ) ששם תיצור קובץ .csvלאחר מכן הולכת למחלקה Q2 ומפעילה שם את הפונקציה בשם: ReadDir.

/חבילת Convert/

**מחלקה Q2 :**

במחלקה קיימת הפונקציה ReadDir שמקבלת מהמשתמש את הנתיבים של הקבצים והיא מעלה קובץ מ Wiggle Wi-Fi שיוצר List של אובייקטים מסוג Spot. כשכל Spot הוא Wi-Fi.

היא נעזרת בפונקציה ReadFileIntoLIst2 מהמחלקה ReadandWriteCsv , לאחר מכן היא משתמשת במחלקת CalculateQ2 ולבסוף יוצרת קובץ Csv לאחר הסינון ש Calculate עשתה, הקובץ נוצר בעזרת הפונקציה WriteListIntoFile מהמחלקה ReadandWriteCsv.

**מחלקת CalculateQ2 :**

המחלקה מקבלת List עם כל ה Spot שהועלו מהקובץ Csv ומקבלת List של Row (אם זה בקריאה הראשונה ל Calculate ה List ריק).

הפונקציה יוצרת List של Wi-Fi ונכנסת ללולאה שרצה על כל ה List של הספוטים במטרה לסנן את הנתונים וליצור List של Row. כתנאי ראשון של הסינון היא בודקת אם הסוג של הספוטים הוא סוג Wi-Fi ולאחר מכן היא בודקת אם הזמן והמיקום אותו דבר כמו ה Spot הקודם, במידה וכן היא יוצרת אובייקט Wi-Fi ומכניסה אליו את ה Ssid ו Mac ן signal ו Chanel. במידה וקיימים יותר מעשרה Wi-Fi באותו זמן ומיקום, היא לוקחת רק את העשרה עם ה Signal החזק ביניהם, תוך כדי שהיא נעזרת בפונקציה Findminsignal שמחזירה את ה Wi-Fi עם הקליטה הכי גרועה.

במידה והזמן והמיקום שונים היא מכניסה את הנתונים שנוצרו מה List הקודם של ה Wi-Fi לתוך List של Row ומתקדמת לזמן ולמיקום החדשים ופותחת להם שורה חדשה. בסופו של דבר היא מחזירה את ה List של ה Row המעודכן.

**מחלקת findMinSignal :**

המחלקה מקבלתList של Wi-Fi ובעצם מחזירה את ה Wi-Fiעם הסיגנל הכי גרוע.

/חבילת /Filter

**מחלקת Q3:**

המחלקה מקבלת נתיב להעלות קיבצי CSV לאחר הסינון שעשינו ב Q2. היא משתמשת בפונקציה ReadFileIntoLIst3 ממחלקת ReadandWriteCsv ושולחת את ה List לסינון ראשוני על ידי מחלקת Filter ולאחר מכן סינון על ידי מחלקת Mac , בסוף הסינון של מחלקת Mac נוצר קובץ KML, בעזרת מחלקת WriteToKML .

**מחלקת Filter :**

המחלקה מקבלת List של Row ונכנסת ללולאה ששואלת את המשתמש אם ירצה לסנן לפי זמן שם או מיקום. השתמשנו בפקודה Switch . במידה שהמשתמש בוחר לפי זמן, הוא צריך להכניס את הזמן ההתחלתי והזמן הסופי שהוא רוצה ועל פיו לסנן ואז נקרא לפונקציה: CalculateByTime. היא תחזיר List מעודכן, לפי הזמנים שהוזנו.

במקרה שהמשתמש ירצה לסנן לפי שם, הוא יקליד את השם וה-List ישלח לפונקציה CalculateById. שיחזיר List רק עם ה Id שהוזן.

במקרה שהמשתמש ירצה לסנן לפי מיקום, הוא יתבקש להזין את הקורדינטה ואת הרדיוס שעל פיהם הוא רוצה לסנן וה-List ישלח לסינון על פי הקורדינטה והרדיוס שהוא הזין על פי הפונקציה CalculateByILocation .

הסבר על הפונקציות שמחלקת Filter משתמש בהם:

**: CalculateByTime** הפונקציה משתמשת במחלקת DateFormat של Java שמכניסה את מה שהמשתמש הזין וממירה את ה String שיש ב List ל DateFormat. במידה וכן ונמצא שזה בטווח של הזמנים, היא מכניסה אותו לרשימה.

**CalculateById**: הפונקציה משווה את ה Id שהוזן על ידי המשתמש ואת ה Id הקיימים ב List. במידה והם שווים, היא מכניסה את ה Id ל List החדש.

**CalculateByILocation**: הפונקציה ממירה למשתנה מסוג Double את הקורדינטה שקיימת ברשימה, במידה שהיא בטווח שהמשתמש הזין, כלומר המרחק בין 2 הקורדינטות קטן או שווה לרדיוס, היא מכניסה את Wi-Fi ל List.

**מחלקת MacQ3 :**

הפונקציה מקבלת List של Row ורצה על כולו, כך שהיא משווה בין כל ה Mac של כל ה Wi-Fi שקיימים. במידה ונמצא Mac זהה, הפונקציה בודקת את החוזק של הקליטה ומשאירה את ה Wi-Fi עם הקליטה החזקה יותר ומוחקת את ה Wi-Fi עם הקליטה הנמוכה יותר.

/חבילת /Objects

אובייקט Row : האובייקט מכיל שני אובייקטים, שאחד מהם הוא List של אובייקטים של Wi-Fi. והשני זה אוביקט של Details.

אובייקט Wi-Fi: האובייקט מכיל ארבעה משתנים שמכילים ארבעה דברים משתנים שאנחנו לוקחים מכל Wi-Fi שנמצא בדגימה של אותו זמן ומיקום.

אובייקט Details: האוביקט מכיל שישה משתנים שהם בעצם ששת המשתנים הקבועים שנמצאים בדגימה של אותו זמן ומיקום.

אובייקט Spot: האובייקט מכיל עשרה משתנים, שהם בעצם עשרת המשתנים הראשוניים שאנו מקבלים מהמשתמש בקריאה הראשונית של הקובץ מ WigleWifi .

אובייקט Mac : האובייקט מכיל ארבעה אובייקטים, שלושת הראשונים הם המיקום הגיאוגרפי והרביעי הוא העוצמה של אותו mac.

אובייקט Listmac : האובייקט מכיל חמישה אובייקטים, הראשון הוא בעצם רשימה של האובייקט Mac, והשאר אובייקטים של זמן, שם, ID ועוצמה.

/חבילת Read\_Write/

**מחלקת WriteToKml :**

הפונקציה משתמשת במחלקת Jar API של Java.

הפונקציה מקבלת List של Wi-Fi ומעלה אותם לנקודותWi-Fi בתוך קובץ Kml.

מהמיקום שבו נלקחה הדגימה ויוצרת רצועת זמן בעזרת TimeStamp לאחר מכן נעשית בדיקה שנרשמו דגימות Wi-Fi ל KML, במידה מכן, הפונקציה מוצאת את הדגימות לקובץ KML , במידה ולא הפונקציה לא יוצרת קובץ KML ומדפיסה שאין לה דגימות כדי ליצור איתם קובץ KML.

**מחלקת CopyListToList** :

הפונקציה מקבלת List ויוצרת עוד אחד זהה לו ומחזירה אותו.

**מחלקה ReadandWriteCsv :**

\*פונקציית ReadFileIntoLIst2 : הפונקציה מקבלת את הנתיב של הקובץ ויוצרת List של אובייקטים מסוג Spot , כך שכל Spot הוא בעצם Wi-Fi שנסרק שקיים בקובץ Csv.

\*פונקציית ReadFileIntoLIst3 :

הפונקציה קוראת את הקובץ Csv שאנחנו יצרנו לאחר הסינון ומחזירה List של אובייקטים מהסוג Row.

\*הפונקציה WriteListIntoFile:

הפונקציה מקבלת List של אובייקטים מהסוג Row ויוצרת קובץ Csv, כך שהשורה הראשונה היא שורת התוכן ושאר השורות הנוספות הן אובייקט מסוג Row שמועלה לרשימה של Csv.

/חבילת Junit\_Testing/

**מחלקת בדיקות Junit:**

CalculateByIdTest:

הפונקציה קוראת קובץ Csv שאנחנו יצרנו לאחר הסינון ומסננת לפי Id שמזינים ומסננת בעזרת הפונקציה CalculateByIdולאחר מכן היא מפעילה את הפונקציה בשנית ומשווה עם הגודל של הרשימה השתנה בין ההפעלה הראשונה לשניה. במידה ויש שינוי היא תחזיר שגיאה.

CalculateByLocationTest: הפונקציה קוראת קובץ Csv שאנחנו יצרנו לאחר הסינון ומסננת לפי קורדינאטות שמזינים ומסננת בעזרת הפונקציה.

CalculateByLocation ולאחר מכן היא מפעילה את הפונקציה בשנית ומשווה עם הגודל של הרשימה השתנה בין ההפעלה הראשונה לשנייה. במידה ויש שינוי היא תחזיר שגיאה.

CalculateByTimeTest: הפונקציה קוראת קובץ Csv שאנחנו יצרנו לאחר הסינון ומסננת לפי זמנים שמזינים ומסננת בעזרת הפונקציה.

: CalculateByTimeולאחר מכן היא מפעילה את הפונקציה בשנית ומשווה עם הגודל של הרשימה השתנה בין ההפעלה הראשונה לשנייה. במידה ויש שינוי היא תחזיר שגיאה.

: CalculateQ2TestCountIn Details

הפונקציה מקבלת List של Row ובודקת אם יש שוויון בין האיבר Count באובייקט Details לבין מספר ה Wi-Fi שקיימים. במידה ואין שויון היא מחזירה שגיאה.

MacQ3Test:

הפונקציה מקבלת List של Rowמפעילה את הפונקציה MacQ2 ובודקת אם לאחר הפעלתה קיימים mac זהים במידה וכן היא מחזירה שגיאה.

WriteToKmlTest:

הפונקציה בודקת אם במידה שאפילו יש רק Wi-Fi יחיד נוצר קובץ kml .

WriteToKmlTestEmpty:

בודקת אם במידה והתקבל קובץ ללא שום נקודת Wi-Fi הפונקציה לא מייצרת קובץ Kml במידה וכן היא מחזירה שגיאה.

ReadAndWriteCsvNoWigeleFile:

מקבלת נתיב של קבצי csv ללא קובץ שלWigeleWifi ומוודאת שהפונקציה מחזירה False .

כלי תוכנה:

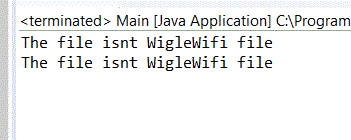
Excel, Wigele-Wifi, Eclipse, Google-Earth, j-unit, Jar API java Kml.

**ניסויים:**

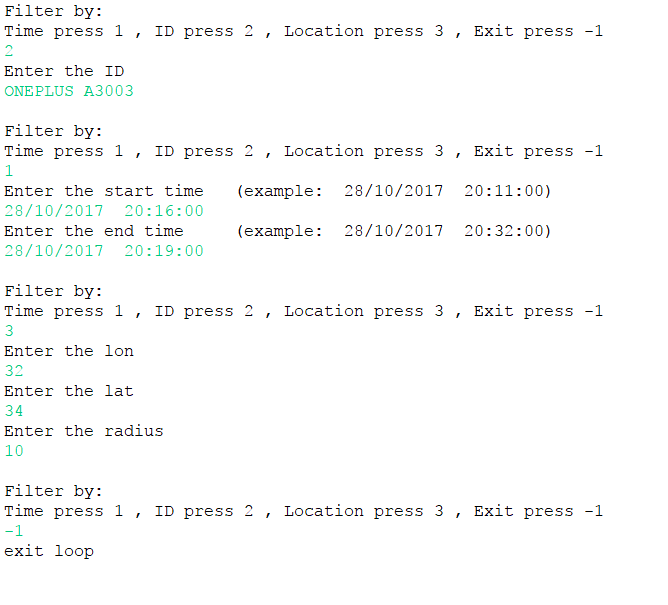
עשינו מספר ניסוים שהתוכנה עובדת, העלנו כל מיני קבצי Wigele-Wifi ,סיננו, ובנוסף בדקנו קבצים ריקים וכו' .

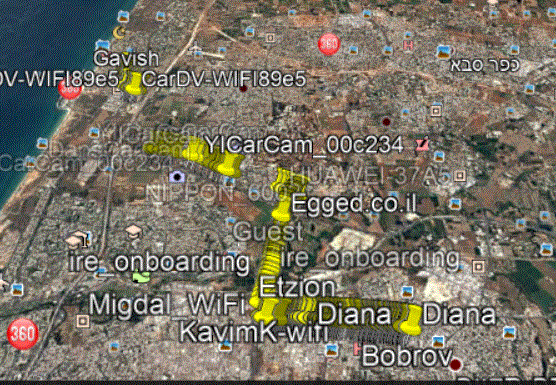
להלן תמונות של חלק מהניסוים:

נתיב ללא קבצים מתאימים -> לא מגיע לפילטר

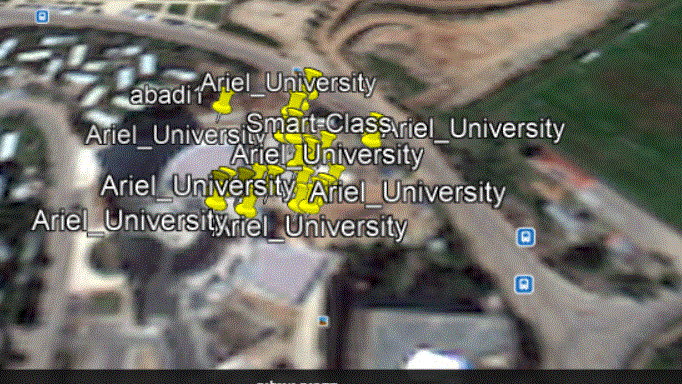


דוגמא לסינון – בהתחלה לפי ID ולאחר מכן לפי תאריך ולאחר מכן לפי מיקום

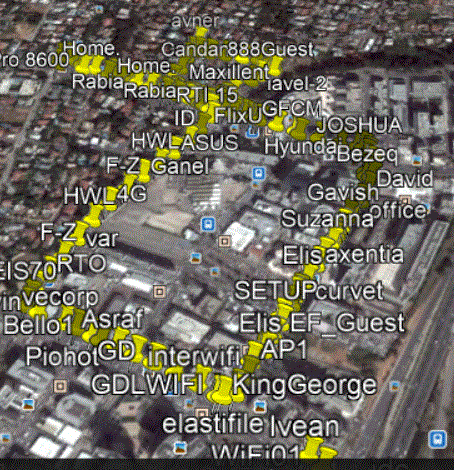


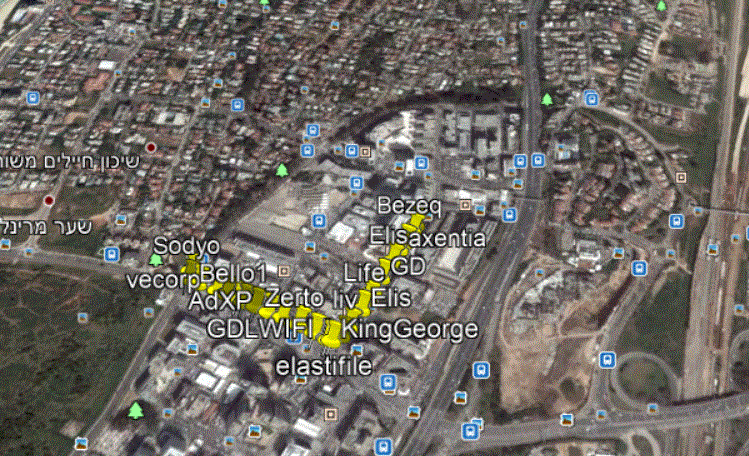
סינון רק ע"י משתמש

סינון לפי המיקום של האוניברסיטה בדגימה שלנו



דוגמא לסינון לפי ID : Lenovo



דוגמא לסינון לפי Date לאחר סינון לפי ID : Lenovo