**דו"ח מטלה**

מגישים: גלעד פוקס 203254370

מתן מעברי 305266652

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

/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 FilterAnd / FilterNot :**

שתי המחלקות מממשות את הפונקציות שבממשק FilterInterFace .

FilterNot מבצעת את אותם פעולות רק בצורת המשלים שלה.

הסבר על הפונקציות שמחלקת FilterInterFace משתמשת 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 Filter:**

המחלקה בעצם מקשרת את המידע שמועבר ממחלקת הGUI אל שאר המחלקות, בנוסף המחלקה הזאת מכילה את מבנה הנתונים העיקרי שאיתו מתעסקים. כל בקשה מהמשתמש שמגיע מהGUI מפעיל פונקציה במחלקה הזאת. המחלקה מכילה מספר פונקציות שמגיעות מהמשתמש כגון, פונקציות של קריאת קובץ, קריאת קבצים מתיקייה, כתיבה לקובץ, הפעלת אלגוריתמים והפעלת טרדים.

בנוסף יש את פונקציית filtermain שהיא אחראית על הסינון שהיא מקבלת מהGUI לפי איזה פרמטרים לסנן של OR או AND או לפי מיקום או לפי שעה או לפי שם.

**מחלקת MergeList:**

המחלקה לוקחת שתי רשימות ומאחדת אותם לרשימה אחת ולאחר מכן מוחקת שורות כפולות במקרה וקיימות.

/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 זה יורש חלק מהתכונות של האובייקט WIFI.

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

אובייקט FilterInfo: האובייקט מכיל ארבעה אובייקטים, הראשון הוא בעצם מראה אם ביצענו סינון ע"פ AND או OR, השני מראה אם סיננו ע"פ NOT או REGULAR, השלישי אם סיננו לפי זמן או שם או מיקום והרביעי את המידע של הסינון, למשל, בסינון שם המידע יהיה השם עצמו.

/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 Algorithms/

**מחלקת Algo1mac :**

הפונקציה מקבלת קובץ סריקה ושם של mac, הפונקציה מחפשת את כל המאקים עם השם ומכניסה אותם לרשימה

שמכילה את האובייקט mac ושולחת לבסוף את הרשימה לחישוב במחלקה MacW.

**מחלקת Algo1MacW:**

הפונקציה מקבל רשימה של האובייקט mac ויוצרת רשימה חדשה שבו מכניסים את המיקום חלקי עוצמת הקליטה בריבוע.

ולבסוף היא מחברת את כל התוצאות מחלקת בקליטה ומחזירה את התוצאות הסופיות לAlgo1mac.

**מחלקת Algo2mac:**

הפונקציה מקבלת מהמשתמש רשימה של סריקות, שורה של סריקה ,והיא מחפשת 4 שורות הכי דומות לפי האלגוריתם.

הפונקציה יוצרת מערך של דאבל בגודל של הליסט ומכניסה בכל מיקום את ערך הדימיון.

לאחר מילוי המערך הפונקציה שולחת את המערך למחלקת FindFive כדי למצוא את השורות הכי דומות.

ומחזירה מערך עם המיקומים של השורות הכי דומות ולאחר מכן היא שולחת שורות אלה למחלקה PI כדי למצוא את המיקום המשוערך.

ולבסוף היא מחזירה את השורה עם המיקום המשוערך.

**מחלקת PiAlgo2:**

פונקציה pi

המחלקה מקבלת את השורות הכי דומות ואת השורה ומחשבת להם את הpi לצורך חישוב המיקום ולבסוף מכניסה את הpi למערך של דאבל.

לבסוף שולחת לפונקציה insertpi את הנתונים ונעשה החישוב הסופי לפי האלגוריתם למציאת המיקום המשוערך ולבסוף היא מזינה בשורה את המיקום

ומחזירה אותה למחלקה ALGO2MAC.

**מחלקת FindFive:**

המחלקה מקבלת מערך ומוצאת את 4 המספרים הכי גבוהים תוך כדי העזרות בפונקציה min.

/חבילת /GUI

**מחלקת GUI :**

מחלקה שבעצם אחראית על הממשק הגרפי, כל האובייקטים שמופיעים גרפית, קרי כפתורים לייבלים ותיבות טקסט, נוצרים במחלקה זו. כל המידע שמועבר מהמשתמש דרך אותם אובייקטים נשלחים למחלקות אחרות ששם מתבצעים החישובים, לאחר מכן, הערך שחושב חוזר למחלקת הGUI ומשם "נפלט" למשתמש או נשמר במבנה נתונים שקיים במחלקת Filter ומשם ניתן לייצא את אותו מבנה נתונים לקובץ CSV או KML לפי בחירת המשתמש (לפי הכפתור שיבחר בGUI).

**מחלקת NioFileSupport :**

מחלקה שאחראית על יצירת THREAD מאזין, המחלקה הזאת מתבצעת באמצעות התראה מהמחלקת GUI. ברגע שלנתיב של התיקייה שמכילה את הקבצים שהמשתמש בוחר, בוצעו שינויים כגון הוספת / מחיקה / שינוי קובץ, הפונקציה בGUI שולחת את הנתיב של התיקייה לפתיחת thread שיאזין לתיקייה, ברגע שהtread זיהה שינוי, התיקייה נקראת מחדש והמבנה נתונים משתנה בהתאם. אם למשל המבנה נתונים קרא מקובץ מסויים וביצע מספר סינונים על פיו, ולאחר מכן, הקובץ נמחק אז כל שלבי הסינון שבוצעו על פיו ימחקו אבל לא ימחקו את כל הסינונים.

/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 .

MergeListTest:

יוצרת שתי רשימות זהות ולאחר מכן מאחדת אותם בעזרת הפונקציית merege ולאחר האיחוד בודקת אם ברשימה שהתקבלה אין כפילויות בעזרת בדיקה ידנית.

findMinSignalTest:

יוצרת רשימה של האובייקט wifi ומכניסה ידנית 10 Wifi-ים כשאנו יודעים לצפות באיזה אינדקס הקליטה הכי נמוכה, ובבדיקה אנו בודקים אם היא באמת מחזירה אותו.

FindFiveTest:

יוצרת מערך רנדומלי עם 100 ערכים, מפעילה את הפונקציה שמחזירה מערך עם המיקומים הכי טובים כשבמיקום ה0 יש את האחד עם הערך הגבוה הכי נמוך, ועוברת ידנית על הרשימה ובודקת שמספר הערכים שגדולים ממנו לא עובר את גודל הרשימה.

כלי תוכנה:

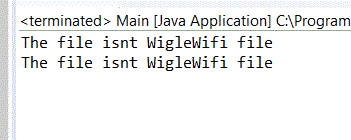
Excel, Wigele-Wifi, Eclipse, Google-Earth, j-unit, Jar API java Kml.

**ניסויים:**

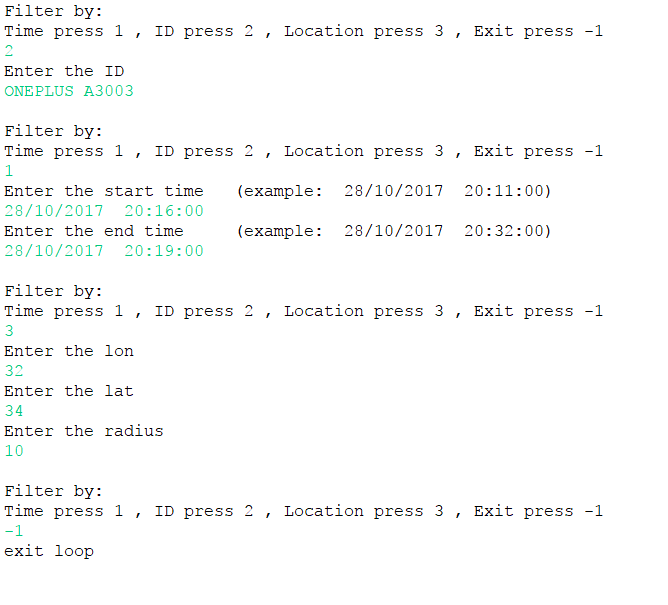
עשינו מספר ניסוים שהתוכנה עובדת, העלנו כל מיני קבצי Wigele-Wifi ,סיננו, ובנוסף בדקנו קבצים ריקים וכו' .

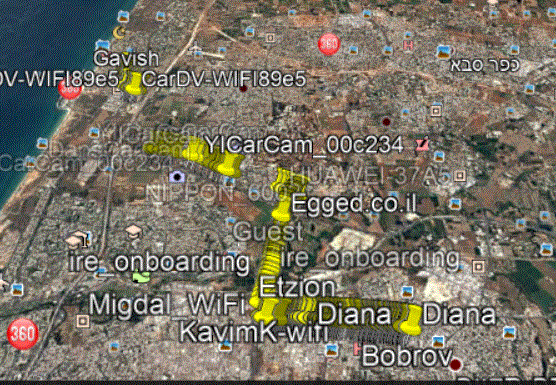
להלן תמונות של חלק מהניסוים:

נתיב ללא קבצים מתאימים -> לא מגיע לפילטר

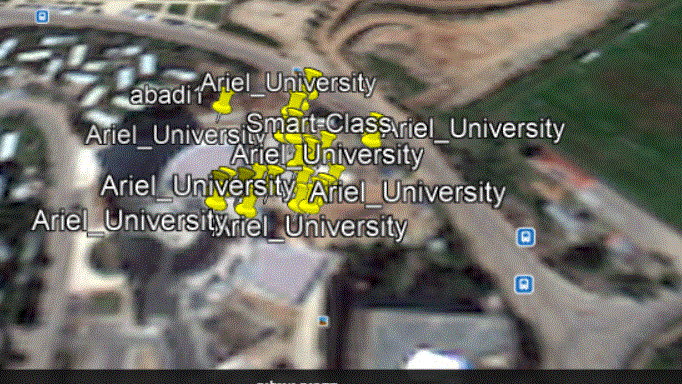


דוגמא לסינון – בהתחלה לפי ID ולאחר מכן לפי תאריך ולאחר מכן לפי מיקום

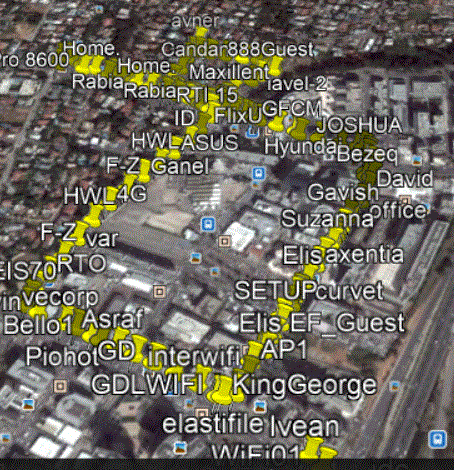


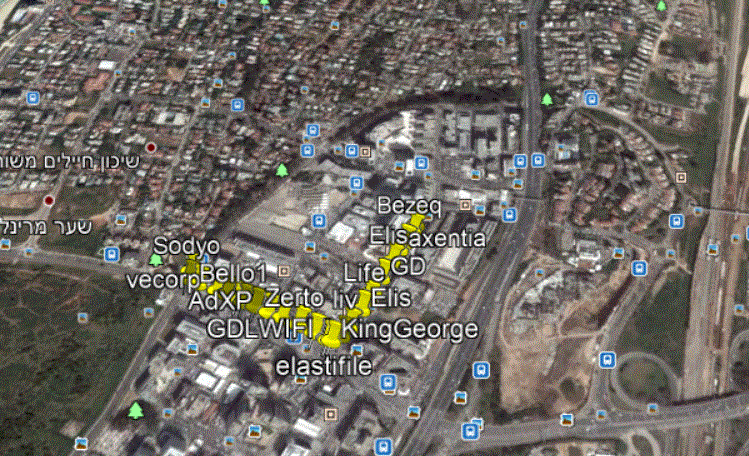
סינון רק ע"י משתמש

סינון לפי המיקום של האוניברסיטה בדגימה שלנו



דוגמא לסינון לפי ID : Lenovo



דוגמא לסינון לפי Date לאחר סינון לפי ID : Lenovo