**מטלה 0, מבוא לתכנות מונחה עצמים**

**מגישים:** אהוד פלקסין סיימון לנקרי.

המערכת מקבלת נתיב לתיקיה.

היא מחפשת קבצי CSV מהאפליקציה Wigle Wifi 1.4.

היא בודקת את תקינות הקלט ולאחר מכן מכניסה את המידע לתוך מבנה נתונים.

התכנית מוציאה כפלט קובץ csv לאחר סינון ומניפולציות כנדרש.

לאחר כן התכנית קוראת את קובץ הcsv ה"מעובד", מבצעת עליו סינונים עפ"י דרישת הלקוח,

וכותבת אותו לתוך קובץ kml

לבסוף המערכת מוציאה כפלט קובץ kml לפי דרישת הלקוח

**שיפור ממטלה קודמת:**

קיבלנו ביקורת מהמתרגלת רויטל ומשני חברים לתואר.

בסה"כ הם אמרו שהפרויקט נראה טוב, יש שימוש במונחה עצמים, וניכר שהוא נותן את הפלט הראוי.

הוספנו את הApi לkml בנוסף כתבנו junit test לכל מחלקה.

כמוכן כתבנו את הסינונים לקובץ הkml מחדש.

(בגרסה הקודמת היינו יוצרים Deta Base חדש שאותו מסננים ואז כותבים, עכשיו אנחנו מסננים תוך כדי ריצה על הקובץ המקורי וחוסכים בזיכרון)

והוספנו סינונים חדשים, (למשל סינון המקבל מהמשתמש נקודה ורדיוס וכותב לקובץ רק נקודות הנמצאות במעגל המוגדר ע"י הנקודה והרדיוס. שימוש במחלקה Java Date לסינון עפ"י תאריכים ועוד.)

שיפור של התיעוד, עדיין יש עבודה לעשות אך יש שיפור מפעם קודמת.

**לשיפור:**

כתיבת תיעוד יותר טוב ואיכותי, צמצום בשורות הקוד היכן שניתן.

שיפור הjunit , הפכתי חלק מהפונקציות מprivate לprotected ע"מ שאוכל לבדוק אותם.

להבא אחזיר אותם לprivate אך אעשה בדיקה יותר איכותית על המתודות הציבוריות, והאובייקטים.

**בעיות:** הסינון של התאריך בעייתי, מקבל שעה אך נכנס לתוך parse של תאריך. אני מתכוון לעשות את כל הסינונים מחדש במחלקות נפרדות, לכן אתקן את זה אז.

המחלקה של Tokml מסורבלת נורא, בעיקר בגלל שהסינונים נמצאים בה. אכתוב את הסינונים מחדש בצורה טובה יותר ובמחלקה נפרדת.

הjunit ברמה יחסית נמוכה, כמו שכתוב לעיל כתבתי test לכל מחלקה ולכל פונקציה והפכתי את חלק מהפונקציות לpublic מprivate עקב כך ומפאת קוצר הזמן הסינון לא היה ברמה גבוהה.

**Wifispot**

המחלקה מייצגת נקודת WIFI, יש לה getterim,setterim ובנאים כדרוש.

היא מכילה את המאפיינים של נקודת WIFI שניתן למשוך מהאפליקציה.

**WifiSpots**

המחלקה מייצגת אוסף של נקודות Wifi מהוראות המטלה הסקנו שלאוסף של נקודות Wifi

יכולות להיות תכונות משותפות (גובה מיקום, זמן).

ולכן במחלקה זו יש משתני מחלקה שתפקידם להחזיק את מידע זה.

את נקודות הWIFI אנחנו מחזיקים בתוך ArrayList

**Scan**

המחלקה scan מייצגת אוסף של סריקות. ההגדרה של סריקה היא רשתות הWifi אשר נקלטו בפרק זמן מסוים. כאשר אנחנו מקבלים קובץ CSV עם שניות מדובר על רשתות שנסרקו באותה שנייה בדיוק.

ובקבצים בהם אנחנו מקבלים קובץ CSV עם דקות מדובר על רשתות שנסרקו באותה דקה בדיוק.

משתנה המחלקה הוא Arraylist של WIFISPOTS.

**CSV**

המחלקה מייצגת אוסף של קבצי CSV

משתנה המחלקה PATH מחזיק את הנתיב אל התיקייה ממנה אנו רוצים למשוך את קובצי הCSV.

משתנה המחלקה Scans מחזיק אוספים של אוספים של סריקות. כלומר אוסף של מידע מעובד מכל קובץ. כאשר כל אוסף של סריקות מייצג קובץ. אז אוסף של אוסף של סריקות מייצג קבצים.

משתנה המחלקה files מחזיק את הנתיב אל הקבצים השונים.

**Tokml**

המחלקה מטפלת בכתיבה של קבצי KML משתנה המחלקה שלה, DB מחזיק את כל המידע שמשכנו מהתיקייה. עליו אנחנו יכולים להפעיל סינונים שונים בעזרת פונקציות המחלקה.

משתנה המחלקה MACIM הוא Arrayliist<Wifispot> המחזיק את המיקום החזק ביותר של כל נתב בDB.

**API'S וopencode**

השתמשנו בAPI של Commons CSV על מנת למשוך את המידע מקובצי הCSV.

נעזרנו בדוגמאות באתר שלהם ע"מ לכתוב את פונקציית הקריאה.

בחרנו בcommons csv לאחר נסיון עם כמה Api אחרים שלא עבדו. עשינו את זה כבר למטלה 0.

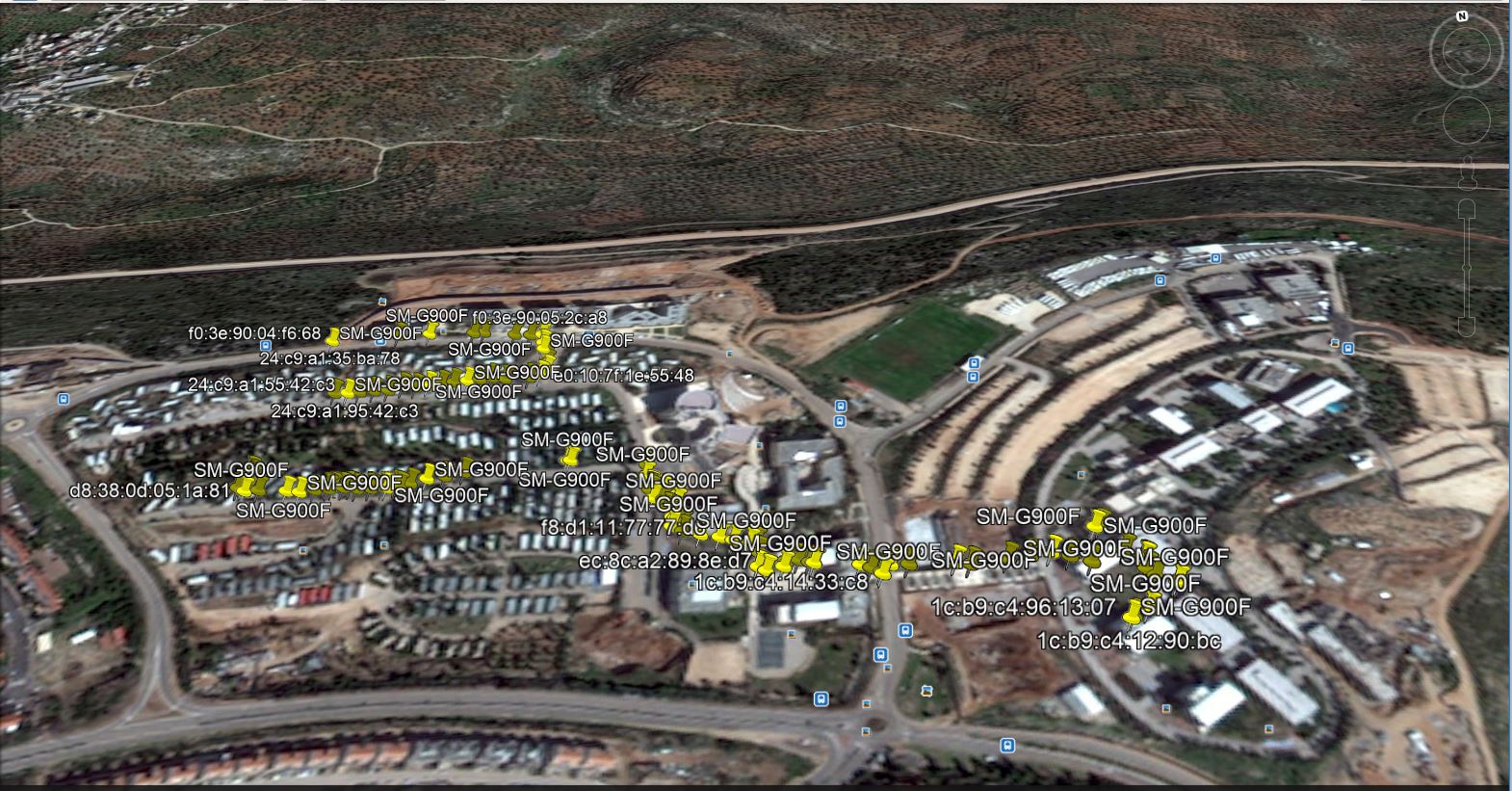
הפיצ'ר של קריאה ומשיכה אוטומטית של מידע למשתנים היה מועיל מאוד, והקל על עבודתנו.

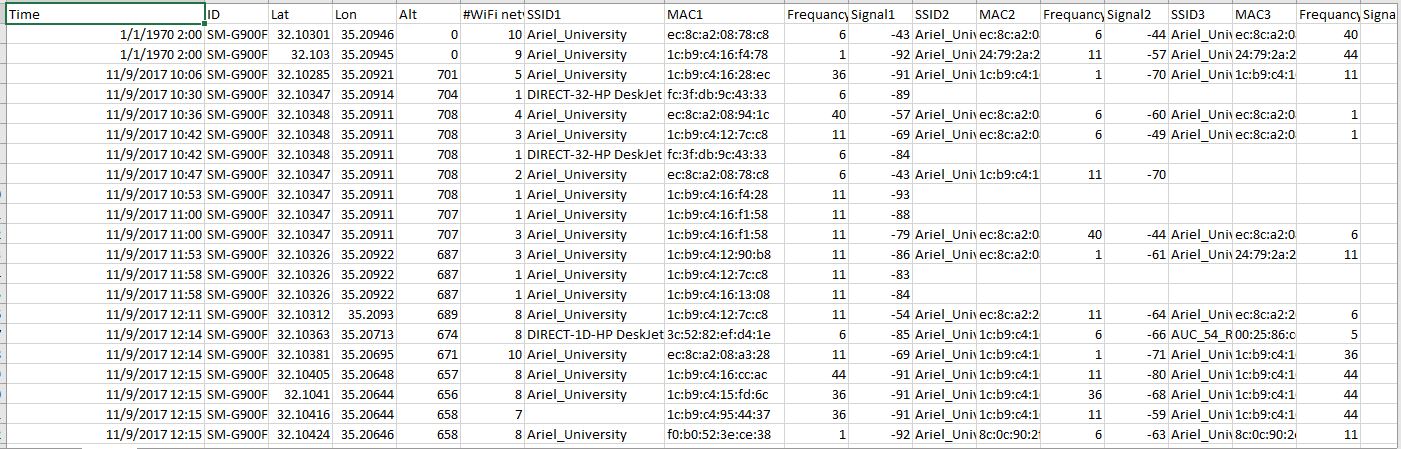
כמו כן השתמשנו בAPI בשם "JAK" בחרנו בו, כי לאחר חיפוש באינטרנט זה מה שמצאנו.

במבט לאחור אולי היינו מחפשים יותר טוב. למרות שהאתר נראה מרשים נתקלנו בקושי למצוא מידע על הAPI הזה, כמו דוגמאות ומה עוד יש לו להציע.... הדוגמא באתר פשוטה מאוד ובחיפוש באינטרנט לא מצאנו הרבה מידע.

לחישוב המרחק בין שני קואורדינאטות השתמשנו בפונקציה <http://www.geodatasource.com/developers/java>

מחיפוש מהיר באינטרנט זה מה שמצאנו, הפונקציה נוחה לשימוש. בדקתי אותה למול כמה דוגמאות מגוגל maps וקיבלתי את הרושם שהיא מדויקת מאוד.





קשרי גומלין בין המחלקות

**∞**

**1**

**∞**

**1**

**∞**

**∞**

**1**

**inherits from Comparator**

**inherits from Exception**

**Custom Comparator**

**Data Exception**

**1**

**1**

**∞**