מטלה 1 – מבוא לתכנות מובנה עצמים

מגישים:

נתנאל חוגי, ת"ז: 203553490

שמעון חג'ג', ת"ז: 311367536

חננאל מירון, ת"ז: 302326137

במטלה 0 בנינו תוכנה הממפה נקודות אינטרנט אלחוטי על גבי google earth, באמצעות קבצי KML.
התוכנה מקבלת קובץ CSV מאפליקציית "WIGLE WiFi", ושומרת את המידע על הרשתות במבנה נתונים.
התוכנה מסננת את הנתונים(ניתן גם לייצא קובץ לא מסונן), ומייצאת קבצי CSV ו-KML לפי הסינון המבוקש.

ניתן לסנן לפי:
1. תאריך.
2. מזהה משתמש.
3. מיקום.
4. שעה.

במטלה 1 שיפרנו את הקוד, ותיקנו את הבאגים ממטלה 0.
חילקנו את הקוד ליותר מחלקות, הוספנו מחלקות בדיקה(JUNIT), שינינו את אופן יצירת קובץ הkml על ידי חבילת תוכנה(JAK), והוספנו חותמת זמן בכל רשת בקובץ, כדי להציג את הנקודות על ציר זמן בgoogle earth.

שיפורים ממטלה 0:

1. חילקנו את הפרויקט ליותר מחלקות.
2. הוספנו פונקציית סינון לפי מיקום.
3. הרשתות מופיעות בקובץ הkml לפי "המיקום החזק".
4. שימוש בחבילת תוכנה עבור יצירת קובץ KML.
5. שימוש בjunit.
6. הוספת חותמת זמן בכל נקודה בקובץ הkml, להצגת Time Line.

המחלקות:

1. **wifiScanner-** המחלקה מכילה את הפונקציה **filesReader**, המקבלת מחרוזת עם נתיב לתיקייה, עוברת על הקבצים המתאימים מהאפליקציה שנמצאים בתיקייה, ושומרת את כל המידע בתוך מבנה נתונים. הפונקציה מחזירה מטריצה עם כל המידע.
בנוסף המחלקה מכילה את פונקציות העזר:
- **findCoordinate:** מחזירה את הקורדינטות של מיקום ביצוע הדגימה.
- **findMinRow:** מציאת השורה במטריצה של הרשת עם הסיגנל הנמוך(כדי לשמור רק את ה10 החזקות).
- **findRssi:** מחזירה את עוצמת הקליטה.
- **timeAndExt:** בודקת האם הקובץ בעל סיומת מתאימה(מדלגת עליו במידה ולא), ומחזירה את זמן הבדיקה(משם הקובץ).
- **wifiNumber:** מחזירה את מספר הרשתות בדגימה.

J	I	H	G	F	E	D	C	B	A	
Signal1	Chennel1	MAC1	SSID1	#WiFi networks	Alt	Lon	Lat	ID	Time	1
-47	11	14:ae:db:53:3f:15	tamir_bezeq	10	646	35.2013	32.10452	LG-V400	29/10/2017 17:56	2
										3

הקובץ המיוצא לאחר שימוש במחלקות wifiScanner ו- toCSV

2. **Filters-** המחלקה מתעסקת בסינון המידע מהקבצים שקיבלנו מהאפליקציה, הפונקציות מקבלות את המטריצה שקיבלנו מ**wifiScanner.filesReader**, עוברות על שורות המטריצה ומסמנות **TRUE** בעמודה 0, במידה והשורה עומדת בתנאי הפילטר.

הפילטרים:

- **dateFilter:** סינון לפי תאריך מדויק.
- **IDfilter:** סינון לפי מזהה מדויק.
- **locFilter:** סינון לפי מיקום מדויק.
- **timeFilter:** סינון לפי שעות מסוימות.
- **UnFilter:** ללא סינון.

3. **ReadCsvMatalaFormat-** בעזרת מחלקה זו נעביר את הנתונים מהקובץ CSV שיצרנו בעזרת המחלקה wifiScanner. הפונקציה **ReadFile** מקבלת נתיב לקובץ שיצרנו לפי הפורמט שביקשו במטלה 0, ומעבירה את כל הנתונים למטריצה, ומחזירה אותה.

פונקציות עזר:

- **rowNumber:** מחזירה את מספר הבדיקות בקובץ.
- **wifiCount:** מחזירה את מספר הרשתות בקובץ(שעברו את הסינון).
- **wifiMatrix:** מקבלת את המטריצה מReadFile ומעבירה את הנתונים(מהבדיקות שעברו את הסינון) למטריצה אחרת(מחזירה אותה)בצורה הבאה:

"BOOLEAN"	TIME	ID	LAT	LON	ALT	SSID	MAC	CHANNEL	SIGNAL
#WIFI(1)									
.									
.									
.									
#WIFI(n)									

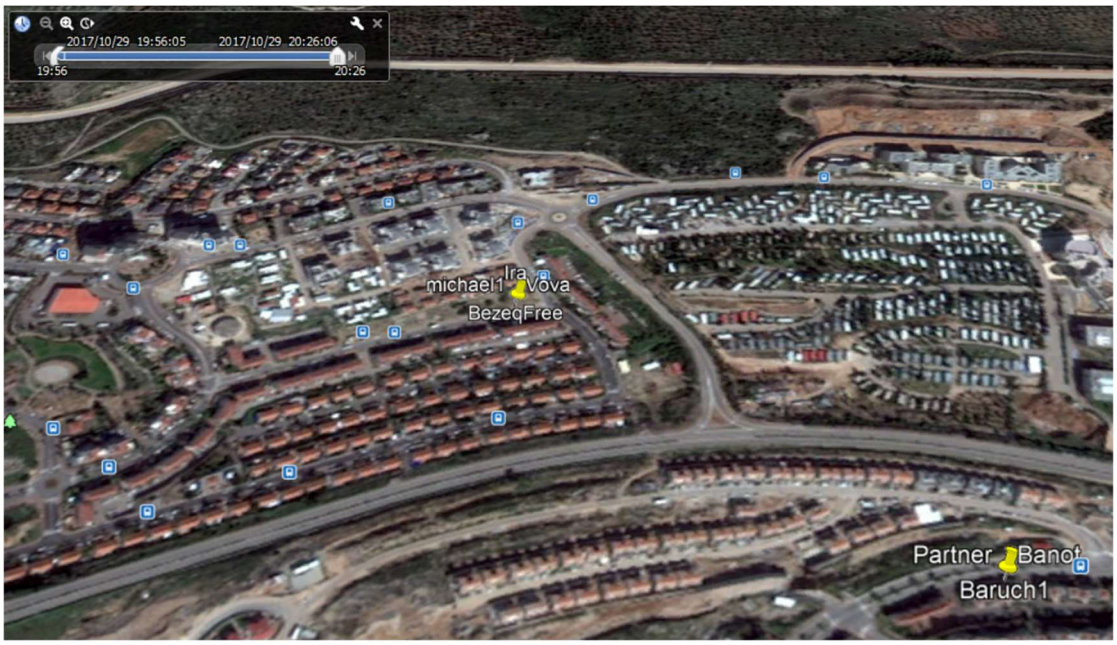
4. **MatrixSortByMAC-** במחלקה זו קיימות שתי פונקציות:

- **MacSort:** מקבלת את המטריצה הנ"ל, וממיינת אותה לפי כתובות mac.
הפונקציה מכילה קומפרטור שמשווה בין כתובת הmac.
- **BestSignal:** הפונקציה מקבלת את המטריצה הממויינת. במידה ורשת(לפי כתובת הmac) מופיעה רק פעם אחת, הפונקציה תסמן "TRUE" בטור 0, אחרת תבדוק מה הפעם שהרשת מופיעה עם הקליטה החזקה ביותר, ותסמן "TRUE" לידה.

5. **ToCsv-** מכילה פונקציה בשם זהה.
הפונקציה מקבלת מטריצה ויוצרת קובץ CSV עם הנתונים מהמטריצה.
הפונקציה עוברת על שורות המטריצה, ובמידה בטור 0 מופיע "TRUE"(לפי הסינון),
הנתונים יודפסו אל הקובץ.

6. **ToKML-** במחלקה זו השתמשנו בחבילת התוכנה JAK(java api for kml).
המחלקה מכילה שתי פונקציות:
- **ToKml:** הפונקציה מקבלת מטריצה(לאחר סינון ומיון) עם נתוני הרשתות ומדפיסה לקובץ KML חדש את השורות שמסומן בהן "TRUE".
הפונקציה משתמשת בפונקציות של חבילת התוכנה, ובפונקצייה שמתחת כדי ליצור נקודה חדשה בקובץ.
- **CreatePlacemarkwithChart:** הפונקציה יוצרת נקודה חדשה בקובץ KML.
את הפונקציה לקחנו מכאן, שינינו אותה כדי שתתאים למטלה, והוספנו חותמת זמן, כדי להציג את הנקודות על ציר זמן בGoogle Earth.

בנוסף לכל מחלקות אלו, ישנן מחלקות בדיקה JUNIT, ומחלקת MAIN.



צילום מסך כולל הTime Line, מתוך Google Earth.