<u>דו"ח מטלה 1</u>

מגישים: סמואל ביסמות – 342533064. אוראל שלום – 315532101

הסבר על המערכת ותיאור רכיבי התוכנה:

התוכנה מאפשרת קריאה של תיקייה עם קבצים והפיכתם לקובץ של טבלת csv מסודרת המכילה את המידע הגיאוגרפי של עוצמת הקליטה של הטלפון, נתוני הגלישה, מיקום, מהירות, מיפוי ועוד. בנוסף, התוכנה לוקחת קובץ זה ומאפשרת סינון של המידע לפי מיקום, זמן, או רשימת משתמשים והצגת המידע על גבי מפה, כך שכל דגימה תוצג כנקודה, וכל נתב לפי כתובת ה- MAC שלו יוצג לפי המיקום הכי חזק שלו.

התוכנה עושה זאת ע"י בניית קובץ KML (בעזרת API) המבוסס על סינון המידע, ופתיחתו בתוכנת Google Earth התוכנה עושה זאת ע"י בניית קובץ הקובץ מכיל את הזמן של כל מדידה, מה שמאפשר לתוכנת Google Earth להציג את המדידות על קו זמן.

כמו כן, המשתמש פעיל במערכת. הוא נותן את שם התיקייה שהוא בוחר (שנמצאת בסביבת העבודה), ולאחר מכן בוחר שם עבור הקובץ שהוא רוצה ליצור. בנוסף, הוא צריך לבחור באיזה אופן לסנן את קובץ ה- KML, על ידי הזנת גבולות לפי בחירתו, ולתת שם לקובץ זה.

מכיוון שלא ידענו עד הסוף תכנות מונחה עצמים ומפאת חוסר זמן, העדפנו להשתמש במחלקות וכלים שאנחנו מכירים יותר כדי להגיש את 'מטלה 0' בזמן. במהלך העבודה הבנו כי מונחה עצמים יהיה יעיל ורלוונטי יותר, כיוון שקידוד התוכנה היה קצת ארוך ומסורבל, והכיל מספר די רב של מחלקות עמוסות.

כעת, לאחר השימוש במונחה עצמים ושיפור התוכנה, הקושי העיקרי היה לשלוט בכל קלט הניתן מהמשתמש. רצינו שהתוכנית תהיה יותר יעילה וקריאה יותר לבוחן, והשימוש במונחה עצמים היה פתרון מועיל ופרקטי יותר.

התוכנה בנויה משלושה ממשקים, המהווים את הפעולות העיקריות שלה: קריאת קובץ, כתיבת קובץ, וסינון קובץ. למחלקות בתוכנה יש מחלקות של JUnit הבודקות ע"י טסטים שאין חריגות ושהקוד רץ כראוי.

פירוט מחלקות התוכנה:

:Main

במחלקה זו צריך המשתמש לתת את שם התיקייה, שאיתה הוא רוצה שהתוכנה תעבוד.

:Read

זוהי מחלקת ממשק המגדירה פונקציית קריאה.

:ReadFile

מחלקה זו קוראת קובץ ומכניסה לרשימת מערכים את האובייקט Wifi. היא משתמשת ב- RPI (Commons-csv: API. מחלקה זו היא מחלקה יורשת של הממשק 'Read'.

:Wifi

מחלקה זו מגדירה אובייקט לרשת Wifi.

:ReadFolder

מחלקה זו קוראת תיקייה ושולחת את כל הקבצים בה לפונקציית 'readfile'. מחלקה זו היא מחלקה יורשת של הממשק 'Read'.

:OpenFile

מחלקה זו מאפשרת פתיחת קובץ כאשר מריצים את התוכנה בעזרת הטרמינל. מחלקה זו היא מחלקה יורשת של הממשק 'Read'.

:Sort

עבור כל תקופה של זמן, המחלקה מסננת את רשתות ה- Wifi לפי הסיגנל.

:UserChoice

המחלקה שואלת את המשתמש באיזה אופן לסנן את קובץ ה- kml (לפי ID, זמן או מיקום).

:Filtering

זוהי מחלקת ממשק המגדירה פונקציית סינון.

:FilteringId

מחלקה זו מסננת את הנתונים לפי ה- ID. רק ה- Wifi עם ה- ID שהמשתמש בוחר יופיע ב- kml. מחלקה זו היא מחלקה יורשת של הממשק 'Filtering'.

:FilteringPlace

מחלקה זו מסננת את הנתונים לפי המיקום. רק ה- Wifi עם המיקום שהמשתמש בוחר יופיע ב- kml. מחלקה זו היא מחלקה יורשת של הממשק 'Filtering'.

:FilteringTime

מחלקה זו מסננת את הנתונים לפי הזמן. רק ה- Wifi עם הזמן שהמשתמש בוחר יופיע ב- kml. מחלקה זו היא מחלקה יורשת של הממשק 'Filtering'.

:WriteFile

זוהי מחלקת ממשק המגדירה את הפונקציות: initialize, פונקציה אבסטרקטית checkData, checkData וcreateFile.

:WriteCsv

מחלקה זו כותבת את קובץ ה- csv. מחלקה זו היא מחלקה יורשת של הממשק 'WriteFile'.

:WriteKml

זוהי מחלקה אבסטרקטית הכותבת קובץ kml, מחלקה זו משתמשת ב- kmlframework :API. מחלקה זו היא מחלקה יורשת של הממשק 'WriteFile'.

:WriteKmlld

מחלקה זו כותבת את קובץ ה- kml לפי סינון ה- ID. מחלקה זו היא מחלקה מורחבת של 'WriteKml' היורשת של הממשק 'WriteFile'.

:WriteKmlPlace

מחלקה זו כותבת את קובץ ה- kml לפי סינון של המיקום. מחלקה זו היא מחלקה מורחבת של 'WriteKml' היורשת של הממשק 'WriteFile'.

:WriteKmlTime

מחלקה זו כותבת את קובץ ה- kml לפי סינון של הזמן. מחלקה זו היא מחלקה מורחבת של 'WriteKml' היורשת של הממשק 'WriteFile'.

:WriteKmlWithoutFilter

מחלקה זו כותבת את קובץ ה- kml ללא סינון. מחלקה זו היא מחלקה מורחבת של 'WriteKml' היורשת של הממשק 'WriteFile'.

:InputException

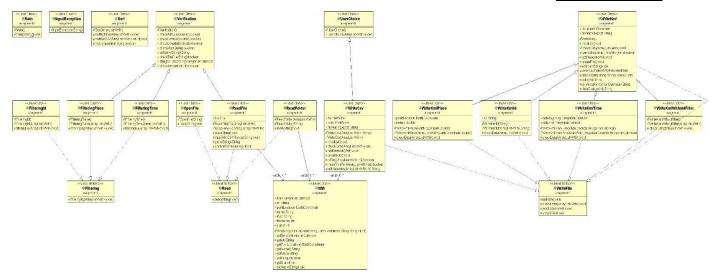
מחלקה זו יורשת ממחלקת Exception של Java, המאפשרת לנו להגדיר את החריגה.

:Verification

במחלקה זו יש פונקציות שאנחנו משתמשים בהם במחלקות אחרות.

* בנוסף למחלקות אלו, כדי לבדוק שהתוכנה תקינה ושאין חריגות, יצרנו מחלקות של JUnit (ב- 'package 'test') שרצות על הקוד בכל מחלקה, ובודקות אם יש בעיות. בעזרתן ידענו לאיזו מחלקה בקוד לפנות ולשפר אותו.

דיאגרמת המחלקות:



שיפורים שנעשו בתוכנה:

חלק ניכר משיפור התוכנה היה השימוש בתכנות מונחה עצמים הכולל ממשקים ומחלקות אבסטרקטיות. בנוסף שיפרנו את מבנה הקוד והעיצוב שלו. השתמשנו יותר באובייקטים של ספריות Java כמו GregorianCalendar ו- EarthCoordinate ובאובייקט Wifi שיצרנו.

יתר על כן, השתמשנו ב- Framework kml :API. מצאנו שלוש אפשרויות באינטרנט ל- API (קישור בהמשך הדו"ח), ובחרנו להשתמש ב- API זה משום שעם Jax היה קשה לנו לעבוד עם הפרוייקט ו- gekmlib היה יחסית ישן ולא מעודכן, כך שהיה מומלץ לא לעבוד איתו. בנוסף, השתמשנו ב- commons-csv :API.

כדי לבדוק שהתוכנה תקינה ושאין חריגות השתמשנו במחלקות של JUnit שרצות על הקוד בכל מחלקה, ובודקות אם יש בעיות. בעזרתם ידענו לאן לפנות בקוד ולשפר אותו. אחד הדברים שעזרו לנו היה יצירת מחלקה InputException, מחלקה יורשת ממחלקת Exception של Java, שאפשרה לנו להגדיר את החריגה.

בלי תוכנה שהשתמשנו בהם:

תוכנת Eclipse:

בתוכנה זו אנו כותבים את הקוד בשפת Java

:WiGLE WiFi אפליקציית

אפליקציה ששומרת את המידע הגיאוגרפי לפי נקודה במפה (מיקום), לפי זמן, לפי ID, לפי רשימת רשתות ה- WiFi WiFi כולל שם הרשת, ה- MAC שלה, תדר ועוצמת הסיגנל ועוצמת הקליטה של הרשת הסלולרית .

:Notepad++ תוכנת

תוכנה שעזרה להציג את קבצי ה- KML כדי שנוכל לראות איך הקובץ בנוי ולראות בעיות שצצו בבניית קובץ חדש. תוכנת Google Earth:

כדי להציג את קבצי ה- KML שבנינו כך שנראה את רשתות ה- WiFi על גבי מפה.

תוכנת Excel:

תוכנה המאפשרת לראות את תוכן הקבצים מסוג csv מאפליקציית WiGLE WiFi. תוכנה זו עזרה לראות את תוצאות הקוד שכתבנו כך שנוכל לשפרו אם צצו בעיות.

:ArgoUml תוכנת

בעזרת תוכנה זו בנינו דיאגרמה המציגה את מחלקות המערכת.

:Oracle תוכנת

נעזרנו ב- Javadoc של ספריית Java כדי להשתמש באובייקטים שלה, במחלקות ועוד.

אתרים שנעזרנו בהם:

Maven: עזר לנו עם הסדר של כל הפרוייקט.

http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

JUnit FAQ: עזר לנו עם ה-

http://junit.sourceforge.net/doc/faq/faq.htm#tests_4

.kml -עזר לנו עם קובץ <u>:Google kml developer</u>

https://developers.google.com/kml/documentation/kmlreference

.csv -של קובץ ה- API :Commons-csv

https://commons.apache.org/proper/commons-csv/

.kml - עזר לנו עם קובץ stackoverflow

https://stackoverflow.com/questions/2952024/which-java-libraries-can-be-used-to-generate-kml

:Github האתר אליו העלינו את הקוד.

https://github.com/SamuelBismuth/Assignment 0

הסבר על הניסוי:

בניסוי התבקשנו למפות כמה שיותר מנקודות ה- wifi באוניברסיטת אריאל. פתחנו את אפליקציית wiglewifi והסתובבנו ברחבי והסתובבנו ברחבי הקמפוס. האפליקציה שמרה את הנתונים של רשתות ה-wifi הקרובות למיקום שלנו בעוד שאנו ממשיכים ללכת.

.WiGLE WiFi תמונות מסך מאפליקציית



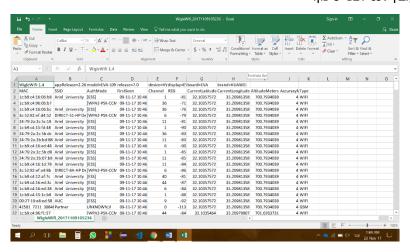


:Google Earth תמונות מסך מתוכנת

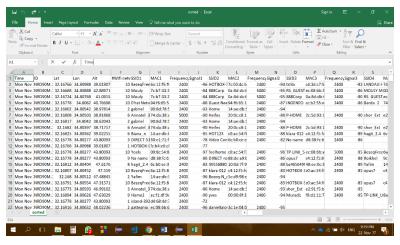




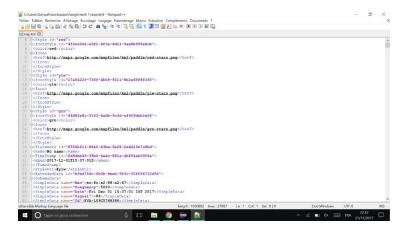
תמונות מסך של קובץ csv לפני סינון:



תמונות מסך של קובץ csv אחרי סינון:



תמונות מסך של קובץ kml (עם



תמונות מסך מתוכנת Google Earth):



בעיות בתוכנה:

התוכנה סה"כ תקינה ורצה כראוי, אך אנחנו לא בטוחים שהמחלקות של JUnit אכן בודקות את כל האפשרויות לחריגות.