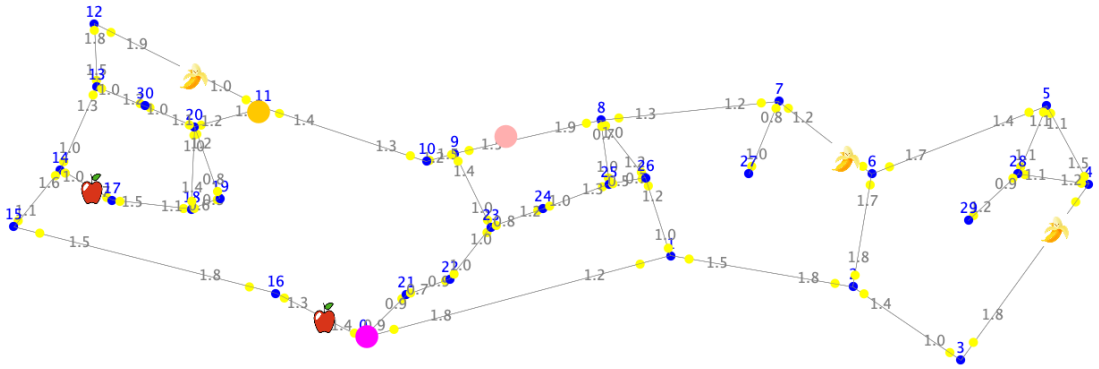
קורס מונחה עצמים – מטלה 3 – גרפים מכוונים: The Maze of Waze

כללית: במטלה זו נשתמש בתשתית של מטלה 2 (מבני נתונים ואלגוריתמים עבור גרף ממושקל, מכוון). היעד העיקרי במטלה הוא לפתח לוגיקה למשחק שבו קבוצת רובוטים צריכה לבצע משימות של תנועה (איסוף "מטבעות") ע"ג גרף ממושקל.
בהינתן גרף ממושקל שעל צלעותיו מיקומו אוסף "מטבעות" (מיוצגים כפירות) נרצה למקם קבוצה (בעלת גודל מוגדר של 1-5 רובוטים) ועבור כל אחד מהרובוטים לבצע תיכנון תנועה בזמן אמת כך שבסוף המשימה (30, או 60 שניות) סך "המטבעות" שכל הרובוטים יאספו יחד יהיה מקסימלי.



איור 1: דוגמא לגרף (שעליו מוקמו 5 פירות: 2 תפוחים ו3 בננות), שלושה רובוטים (מסומנים כעיגולים צבעוניים) צרכים לאסוף יחד כמה שיותר נקודות: על כל פרי שנאסף – יוצר פרי חדש במיקום אקראי. שימו לב שההבדל בין "תפוח לבננה" הוא בכיוון הצלע: לצלע מקודקוד נמוך לגבוהה סומנה "כתפוח" וצלע מקודקוד גבוה לנמוך סומנה "כבננה" – רובוט יכול לאסוף פרי רק אם הוא בכיוון תנועה מתאים!

כדי להקל עליכם את פיתוח המשחק מימשנו עבורכם מספר תת רכיבים, בפרט: "שרת" (קובץ jar) עליו אתם למעשה משחקים, עבודה עם "השרת" מתבצעת דרך ממשק מוגדר (ראו מטה) שמבוסס בעיקר על מחרוזות שמייצגות אובייקטי JSON. השרת מאפשר ליצור "תרחיש" (אחד מתוך 24 אפשריים) שכולל גרף, כמות ומיקום פירות, מספר רובוטים, ומשך פעולה. לאחר היצירה של התרחיש, עליכם למקם את הרובוטים ואז אפשר להריץ את המשחק כאשר הפעולה המרכזית שעליכם לעשות היא לכוון כל רובוט "לצלע הבאה" (כאשר הוא מגיע לקודקוד). בסוף המשחק תקבלו חיווי של כמות הנקודות שצברתם משחק.

שלבי עבודה:

1. בשלב הראשון עליכם לפתוח פרויקט ב github (עדיף חדש) ואת כל המטלה לעשות בו. כולל תיעוד (ב wiki), הגדרה של readme, וביצוע כל התיקונים וההתאמות. הקפידו להתחיל במשימה זו כדי שניתן יהיה לראות את כל ה commits שלכם. הקפידו שאם יש יותר מסטודנט אחד בקבוצה על כל חברי הקבוצה להיות רשומים כשותפים לכתיבה של הפרויקט ולתרום לו. הקפידו לוודא שאתם מצליחים להריץ את הדוגמא המצורפת מול השרת.
2. ממשו מחלקה בשם MyGameGUI שמייצגת ממשק גרפי (בתוך החבילה של gameClient): ומאפשרת לבנות תרחיש [0,23] למקם עליו רובוטים ולשחק "ידנית" ע"י כך

שמכוונים כל רובוט שהגיע לקודקוד לגבי היעד הבא שלו. המחלקה תאפשר הצגה בסיסית של תנועת הרובוטים על הגרף, ייצוג של הזמן הנותן והצגת התוצאה המסכמת.

3. כתבו מחלקה שמייצגת מערכת לניהול המשחק: המחלקה תאפשר "ניהוג אוטומטי" (יעיל) של הרובוטים כך שניתן יהיה להחליף את הפעולה הידנית בניהוג מחשב. מחלקה ליצור תרחיש "בשרת" (כל אחד מ24 התרחישים האפשריים), למקם את הרובוטים בצורה מיטבית ואז להתחיל משחק ולנהג אותם לאורך המחשק. לאחר השלמת המחלקה שלבו אותה בממשק הגרפי שמימשתם בסעיף 2 – כדי לאפשר בדיקה גרפית של התוצאות.

4. ממשו את המחלקה KML_Logger (בתוך החבילה של gameClient) אשר מאפשרת לייצא את הגרף, הרובוטים, והפירות לכדי קובץ KML, שימו לב שכל קובצי הגרפים שאתם קיבלתם הם בסביבת אריאל, ועליכם לאפשר הצגה בזמן של תנועת הרובוטים, וה"אכילה" של הפירות, הוסיפו את האופציה הזאת למחלקה הגרפית (MyGameGUI). מידע מפורט בנושא העבודה עם KML תוכלו למצוא בקישורים הבאים:

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

https://renenyffenegger.ch/notes/tools/Google-Earth/kml/index

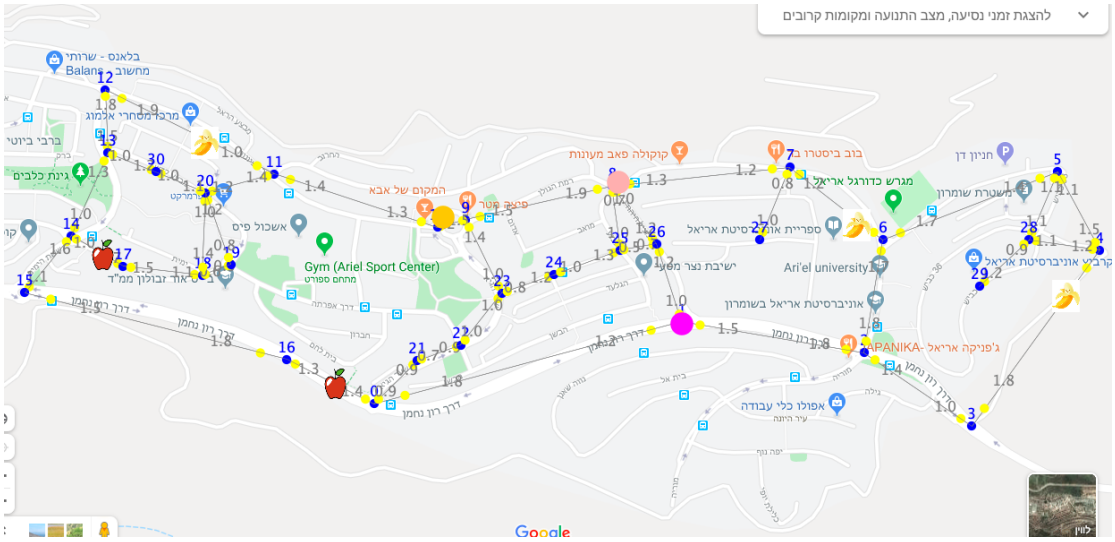
https://developers.google.com/kml/documentation/time

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

https://github.com/micromata/javaapiforkml

הערה חשובה: משימה זו מאפשרת חופש רב במימוש, הדבר החשוב הוא לאפשר ממשק גרפי (חיצוני) כדי שניתן יהיה לבדוק ולהציג את תנועת הרובוטים. בסוף המטלה עליכם להגיש 24 קבצי KML בשמות של 0.kml, 1.kml וכו' (בתוך תיקיית data) אשר כוללים את הביצועים המיטביים שהצלחתם לפתוח כל אחד מהתרחישים – ולפיכך יש להתייחס למשימה זו הן כמשימה מסכמת שתאפשר ניתוח של התוצאות שלכם על ממשק אחיד (KML).

5. כתבו תיעוד + הסברים מפורטים לגבי המערכת, מבנה הפתרון, מערכת התצוגה, וכמובן אופן השימוש בפרויקט מבחינת הורדה, והרצה. לסיכום חושב להציג את כל התוצאות שלכם בטבלה – עבור כל תרחיש לכתוב את התוצאה המיטבית שהצלחתם לקבל.



איור 2: כל הגרפים שקבלתם הם בסביבת העיר אריאל, כאשר המיקום הגיאוגרפי ניתן כך x,y הקורדינטה של x מייצגת את המיקום במזרחיות (lon) והקורדינטה של y מייצגת את כלפי צפון (lat).

חומרים רלוונטיים:

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method	Description
boolean	addRobot(int start_node)	This method allows the user to add & locate the robots, all should be located in order to start a game.
long	chooseNextEdge(int id, int next_node)	This method is the main logical functionality, allows the client algorithm to direct each robot to the "next" edge.
java.util.List<java.lang.String>	getFruits()	return a list of JSON string, each representing a fruit (fixed bonus coin).
java.lang.String	getGraph()	return a JSON representation of graph as a JSON String.
java.util.List<java.lang.String>	getRobots()	return a list of JSON string, each representing a Robot.
boolean	isRunning()	Returns the current status of the game (true: is running, false: NOT running).
java.util.List<java.lang.String>	move()	moves all the robots along each edge, if the robot is on the node (nothing is done - requires to chooseNextEdge(int id, int next_node)
long	startGame()	Start a new game
long	stopGame()	Stops the game, after this method the isRunning() will return false
long	timeToEnd()	return the number of mili-seconds till the game is over

איור 3: פירוט הפעולות (API) של הממשק לשרת המשחק (game_service).
כל החומרים לעבודה עם המטלה מצורפים בקובץ OOP_Ex3.zip.

הנחייה כללית:

- מטלה זו פתוחה למימוש שלכם, התוצרים המרכזיים במטלה הם מערכת משחק שתפתחו ,קובצי KML של תוצאות, ותיעוד הפרויקט ב github.
- עליכם לעשות שימוש בשרת (כפי שמודגם בקובץ הדוגמא), לפי הממשק המצורף (ראו מטה).
- כרגיל הקפידו להוסיף מחלקת בדיקה (Junit) לכל מחלקה לוגית שאתם כותבים (לא צריך לכתוב בדיקה למחלקות GUI, אבל לממיר קבצים כמו KML – כמובן שצריך).
-

הנחיות הגשה:
את המטלה יש להגיש כפרויקט github לפי ההנחיות מפורטות שנמצאות באתר – מטלות שלא תוגשנה לפי ההנחיות לא תזכנה בציון מלא.

בהצלחה.