הנחיות לפתרון תרגילי הבית

- על הקוד המוגש להיות מתועד היטב ועליו לכלול:
  - מפרט, כפי שהודגם בתרגול.
  - תיעוד של כל מחלקה ומתודה ושל קטעי קוד רלוונטיים.
  - במידת הצורך, יש להוסיף תיעוד חיצוני.
- יש להפעיל את הכלי Javadoc כדי ליצור קבצי תיעוד בפורמט HTML ולצרף אותם לפתרון הממוחשב המוגש. כדי לגרום לקובצי ה-HTML להכיל את פסקאות המפרט שבהן אנו משתמשים, יש לציין זאת במפורש. ב-Eclipse, ניתן לבצע פעולה זו באופן הבא: 1. לבחור Export מתפריט File, לבחור Java->Javadoc וללחוץ על כפתור Next, 2. לבחור עבור Javadoc command את הקובץ javadoc.exe מתוך התיקייה bin הנמצאת בתיקייה שבה מותקן ה-Java SDK, 3. לבחור את הקבצים שלהם מעוניינים ליצור תיעוד וללחוץ פעמיים על כפתור Next, 4. להקיש ב-Extra Javadoc options את השורה הבאה וללחוץ על כפתור Finish:

-tag requires:a:"Requires:" -tag modifies:a:"Modifies:" -tag effects:a:"Effects:"

- התנהגות ברירת המחדל של פעולות assert היא disabled (הבדיקות לא מתבצעות). כדי לאפשר את הידור וביצוע פעולות assert, יש לבצע ב-Eclipse את הפעולות הבאות: 1. מתפריט Run לבחור Debug Configurations , 2. בחלון שנפתח, לעבור ללשונית Arguments, 3. בתיבת הטקסט VM arguments לכתוב –ea, 4. ללחוץ על כפתור Debug.

הנחיות להגשת תרגילי בית

- תרגילי הבית הם חובה.
- ההגשה בזוגות בלבד.
- עם סיום פתירת התרגיל, יש ליצור קובץ דחוס להגשה המכיל את:
  - כל קבצי הקוד והתיעוד.
  - פתרון לשאלות ה"יבשות" בקובץ PDF. על הקובץ להכיל את שמות ומספרי תעודות הזהות **של שני הסטודנטים** המגישים.
- הגשת התרגיל היא **אלקטרונית בלבד**, דרך אתר הקורס **ע"י אחד מבני הזוג בלבד**.

- תרגיל שיוגש באיחור וללא אישור מתאים (כגון, אישור מילואים), יורד ממנו ציון באופן אוטומטי לפי חישוב של 5 נקודות לכל יום איחור.
- על התוכנית לעבור קומפילציה. על תכנית שלא עוברת קומפילציה יורדו 30 נקודות.

מועד ההגשה:
09/12/20

המטרות של תרגיל בית זה הן להתנסות בתחומים הבאים:

- כתיבת abstraction function ו-representation invariant.
- בניית היררכיית טיפוסים בעזרת הורשה ורב צורתיות.
- שימוש נכון במחלקות מופשטות ובממשקים.
- שימוש במכלים (containers) וב-iterators.
- בניית ממשק משתמש גרפי בשפת Java.

הצגת הבעיה

בתרגיל זה תממשו אוסף מחלקות המשמשות למתן הנחיות תנועה בין מקומות שונים. המחלקה Route מייצגת דרך, שהיא אוסף סדור של צעדים שביחד מובילים ממקום אחד לשני. דרך עוברת במספר כלשהו של מאפיינים גיאוגרפיים, למשל: מתחילים משדרות מוריה, פונים שמאלה לדרך פיק"א וממשיכים להיכל הספורט. המחלקה GeoFeature מייצגת אוסף צעדים שכולם במאפיין גיאוגרפי בעל אותו שם, למשל, כולם באותו רחוב. החלקים האטומיים המרכיבים Route ו-GeoFeature הם מקטעים גיאוגרפיים, המיוצגים בעזרת המחלקה GeoSegment. הקצוות של GeoSegment הן נקודות על פני כדור הארץ, המיוצגות בעזרת המחלקה GeoPoint. הנחיות תנועה מילוליות בדרך מיוצגות בעזרת המחלקה RouteFormatter. קיימות שתי מחלקות מטיפוס RouteFormatter – WalkingRouteFormatter, למתן הנחיות מילוליות להליכה בדרך, ו-DrivingRouteFormatter למתן הנחיות מילוליות לנסיעה בדרך.

שאלה 1 (70 נקודות)

א.

עבור כל המחלקות הנ"ל, נתון המפרט ושלד מימוש בקבצים בעלי השמות מתאימים. עליכם להשלים את שלד המימוש כך שיעמוד במפרט הנתון. כדאי לממש את המחלקות בסדר הבא:

1. GeoPoint – מחלקה המייצגת נקודה על כדה"א בעזרת קווי אורך ורוחב, תחת הנחת קירבה לטכניון. שימו לב כי קווי האורך והגובה הוכפלו ב10^6 כדי להימנע מבעיות השוואה של טיפוסי נקודה צפה. זכרו לכפול\חלק במספר המתאים כאשר אתם נדרשים לכך. המרחק בין 2 Geopint הוא המרחק האוקלידי, בנוסף נתון במפרט התרגיל יחסי ההמרה בין מעלות אורך\רוחב לק"מ.
2. GeoSegment – מחלקה המייצגת סגמנט בין 2 GeoPoint. heading היא זווית המצפן בה יש ללכת כדי להגיע מנקודה 1 ל2, שימו לב שהסדר חשוב! בנוסף שימו לב כי זוויות המצפן והתקדמותו שונה מזוויות ציר המספרים, בתוך המפרט נתונים לכם רמזי מימוש, השתמשו בהם.

3. GeoFeature – מייצגת אוסף סגמנטים בעלי אותו שם ופעולות עליהם.
4. Route – מייצגת דרך ע"ג כדה"א
5. RouteFormatter – מחלקה אבסטרקטית, הנותנת ייצוג מילולי של דרך כלשהי.
6. WalkingRouteFormatter - ייצוג מילולי של דרך כלשהי עבור הולך רגל.
7. DrivingRouteFormatter - ייצוג מילולי של דרך כלשהי עבור רכב.

חשוב מאוד:
בעת המימוש, עליכם לכתוב עבור כל ארבע המחלקות הראשונות (לא כולל ה-Formatters) representation invariant ו-abstraction function כהערות בתוך הקוד. יש לכתוב מתודת checkRep() ולקרוא לה במקומות המתאימים.

כדי לעזור לכם לוודא את תקינות הקוד שכתבתם, מסופקות המחלקות GeoSegmentTest, WalkingRouteFormatterTest ו-DrivingRouteFormatterTest. מחלקות אלו מדגימות שימוש במחלקות שכתבתם. שימו לב: מחלקות הבדיקה הן פשוטות למדי ואינן מכסות מקרי קצה שונים. אין להסתמך על בדיקה זו, משום שייתכן שתרגיל הבית ייבדק באופן שונה.

הנחיות:
1. כפי שמתואר במפרט של המחלקה GeoPoint, הדיוק של חישוב המרחקים והזוויות מובטח רק כאשר הנקודות על פני כדור הארץ הן "קרובות מספיק" לטכניון. ההנחה היא שכל הנקודות בהן אנו עוסקים הן כאלה ושניתן להתעלם מתנאי זה.
2. לפני תחילת המימוש, שימו לב להנחיות המצויות בהערות בתוך שלדי המימוש. בייחוד, שימו לב להנחיות במחלקה GeoPoint בנוגע לייצוג בעזרת מספרים שלמים ובנוגע למימוש המתודה headingTo().
3. אין לעגל מספרים בייצוג הפנימי, אלא רק לצורך הצגתם.
4. בכל מקום שבו לא מוגדר באופן מדויק מה אמורה להחזיר המתודה toString(), ניתן להחזיר כל תיאור של המחלקה במחרוזת הנראה לכם הגיוני.
5. המחלקות GeoFeature ו-Route מוגדרות במפרט כ-immutable. לכן, יש להגדיר את השדות שלהן כ-final. עליכם למצוא פתרון למימוש המתודות addSegment() של מחלקות אלה תוך שמירה על מגבלה זו.
6. שימו לב כי המחלקה Route יכולה להחזיר ייצוג של הדרך גם כ-Iterator לאובייקטים שהם מופעים של GeoFeature וגם כ-Iterator לאובייקטים שהם מופעים של GeoSegment. בשני המקרים מוחזר Iterator המייצג אותה דרך.

עליכם לצרף תיעוד חיצוני שיכלול תיאור קצר של: 1. שיקולי מימוש שונים, 2. בחירות שעשיתם בהסתמך על שיקולים אלה, 3. חלופות למימוש תוך הצגת יתרונותיהם וחסרונותיהם ביחס למימוש שבחרתם.

ב.
המתודה addSegment() של המחלקה Route מצפה לקבל ארגומנט מטיפוס GeoSegment שנקודת ההתחלה שלו היא נקודת הסיום של this. נניח כי תנאי ההתחלה של מתודה זו יוחלף בתנאי הבא:

```
// @requires gs != null && (gs.p1 == this.end || gs.p2 == this.end)
```

ב1. הסבירו בקצרה כיצד ישתנה מימוש המתודה כתוצאה משינוי זה.
ב2. האם המפרט החדש שנוצר הוא חזק יותר או חלש יותר מהמפרט המקורי?

ג.
התנהגות המתודה getHeading() של המחלקה GeoSegment אינה מוגדרת עבור מקטעים באורך אפס (שהם מקטעים חוקיים). עליכם לשנות את המפרט הקיים כך שיטפל במקרים אלה. יש לשנות אך ורק את המפרט של המתודה הנ"ל ושל המתודות getStartHeading() ו-getEndHeading() של המחלקות GeoFeature ו-Route.

ד.
האם Route הוא true subtype של GeoFeature? האם GeoFeature הוא true subtype של Route? הסבירו.

ה.
תנו **במילים** דוגמה למחלקה חדשה שתייצג הנחיות תנועה מילוליות (מסוג חדש) בדרך שיכולה להיות צאצא של המחלקה RouteFormatter. בנוסף, תנו דוגמה למחלקה חדשה שתייצג הנחיות תנועה מילוליות בדרך (מסוג חדש) שלא יכולה להיות צאצא של המחלקה RouteFormatter. הסבירו מדוע הדוגמאות שנתתם מתאימות או לא מתאימות להיררכיה הנ"ל.

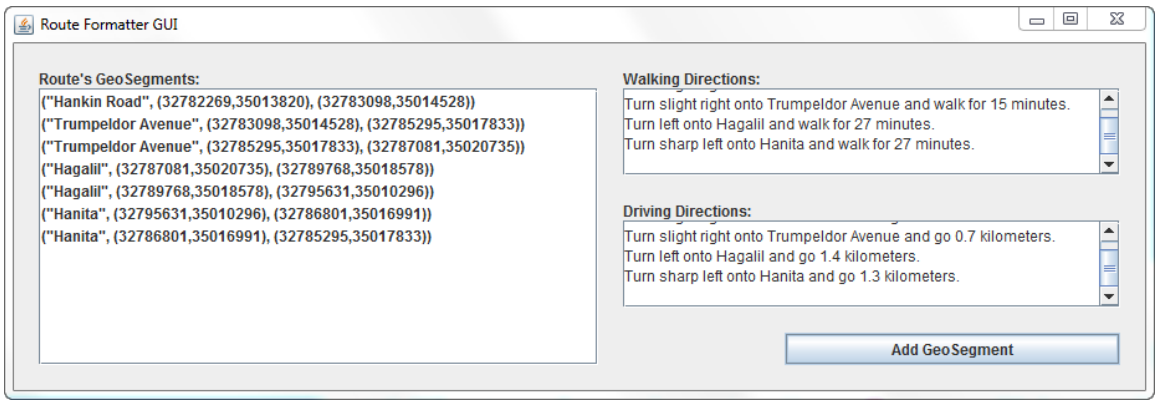
להגשה ממוחשבת:
מימוש המחלקות הנ"ל. על המימוש להכיל את ה-representation invariants וה-abstraction functions שכתבתם, מימוש וקריאות למתודת checkrep, כולל שינויי המפרט בהתאם לסעיף ג'.

להגשה "יבשה":
תיעוד חיצוני למחלקות הנ"ל, בהתאם לצורך. כמו כן, תשובות לסעיפים ב', ד', ה'.

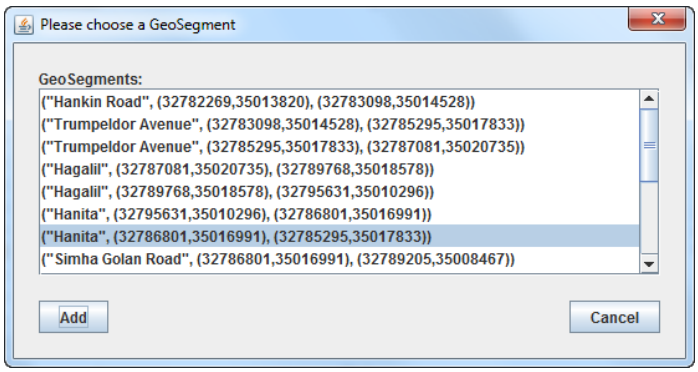
שאלה 2 (20 נקודות)

א.

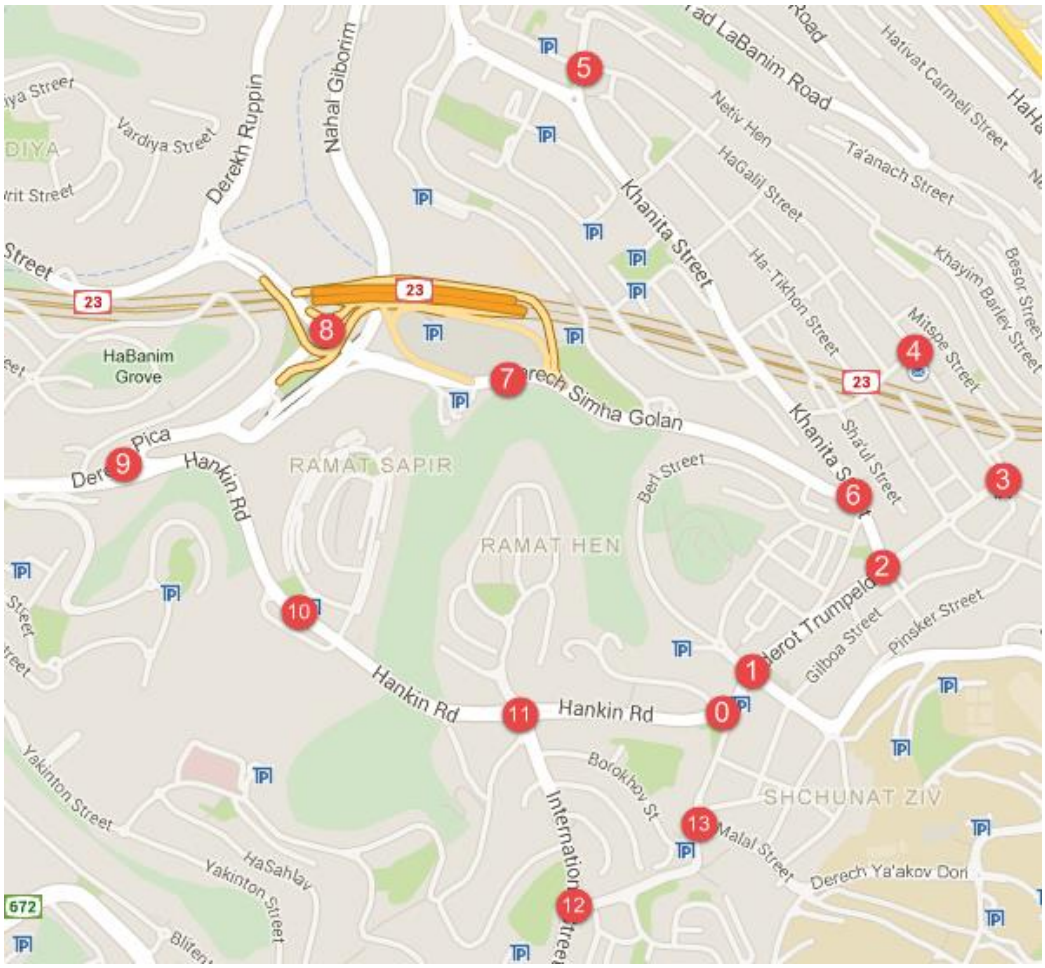
המחלקות RouteFormatterGUI ו-GeoSegmentsDialog יוצרות ממשק משתמש גרפי המשתמש במחלקות שכתבתם בשאלה 1. המחלקה RouteFormatterGUI היא המחלקה שממנה מתחילה הריצה. היא מכילה מופע של Route ומציגה את ה-GeoSegments המרכיבים אותו ואת הנחיות ההליכה והנחיות הנסיעה כדי לעבור אותו. מופע של RouteFormatterGUI עשוי להראות כך:



לחיצה על כפתור Add GeoSegment תפתח מופע של המחלקה GeoSegmentsDialog הנראה כך:



מחלקה זו מאפשרת לבחור GeoSegment מתוך רשימה של GeoSegments אפשריים. לחיצה על כפתור Add מוסיפה את ה-GeoSegment הנבחר למופע של Route המוצג במופע המתאים של RouteFormatterGUI. לחיצה על כפתור Cancel סוגרת את החלון המוצג על המסך. **רשימת ה-GeoSegments לבחירה נלקחת מתוך המחלקה ExampleGeoSegments**, המכילה אוסף של GeoPoints ו-GeoSegments בקרבת הטכניון, כפי שמתואר במפה הבאה:



נתונים מימושים חלקיים של המחלקות RouteFormatterGUI ו-GeoSegmentsDialog. עליכם להשלים מימושים אלה במקומות המסומנים כך שיבצעו את המתואר לעיל. על המימוש של המחלקה GeoSegmentsDialog לא לאפשר הוספת מקטעים ל-Route בסדר לא חוקי.

מחלקות שמומלץ להיעזר בהם(לחלקם ניתן לראות דוגמאות שימוש בתוך הקבצים המסופקים):

- JList
- JScrollPane
- JLabel
- JButton
- GridBagLayout
- GridBagConstraints
- JTextArea

ב.
המחלקה RouteFormatterGUI מכילה מופע של Route בשם route, שאותו היא מציגה על המסך. מבנה זה של התוכנה הוא קביל בתוכנה בסדר גודל קטן, כמו במקרה שלנו. הסבירו מה הבעיה העקרונית בדרך מימוש זו. תארו **במילים** דרך מימוש חלופית עדיפה.

ג.
הבנאי של GeoSegmentsDialog מקבל שני פרמטרים. הסבירו את המשמעות של כל אחד מפרמטרים אלה. מדוע יש צורך להעביר את שני הפרמטרים בעת יצירת מופע של GeoSegmentsDialog?

ד.
לקוח המשתמש בממשק הגרפי הנ"ל נזכר שיש לו דרישה חדשה – הוא מעוניין שב-GeoSegmentsDialog ניתן יהיה גם למחוק מקטעים מה-Route הנוכחי. הסבירו **במילים** אילו שינויים ידרשו בתוכנה כדי לתמוך ביכולת המבוקשת.

להגשה ממוחשבת: מימוש שתי המחלקות הנ"ל
להגשה "יבשה": תשובות לסעיפים ב'-ד'.

שאלה 3 (10 נקודות)

נתון חלק ממחלקה המייצגת פולינום:

```
// Polys are immutable polynomials with integer coefficients.
// A typical Poly is: c_0 + c_1*x + c_2 * x^2 + ...
public class Poly {

   private class PolyTerm {
      int coeff;
      int power;
   }

   private List<PolyTerm> terms;

   // @return the degree of this, i.e., the largest exponent with a non-zero
   //          coefficient, or return 0 if this is the zero Poly
   public int degree() {
      return terms.get(terms.size() - 1).power;
   }

   // @return the coefficient of the term of this whose exponent is p
   public int coeff(int p) {
      int c = 0;
      for (PolyTerm t : terms)
         if (t.power == p)
            c += t.coeff;
      return c;
    }
}
```

א.
כתוב abstraction function עבור המחלקה Poly כך שמימוש המתודות degree() ו-coeff() יהיה נכון.

ב.
כתוב representation invariant עבור המחלקה Poly כך שמימוש המתודות degree() ו-coeff() יהיה נכון.

ג.
כתוב representation invariant אחר מזה שכתבת בסעיף א' שיאפשר מימוש יעיל יותר של המתודה coeff(), והסבר בקצרה.

להגשה "יבשה": תשובות לסעיפים א'-ג'.