מבוא לתכנות מונחה עצמים
עבודת הגשה מס' 2

מבוא

תרגיל זה הוא המשך של התרגיל הראשון, בו בנינו מערכת בקרת תנועה

נא לקרוא את כל המסמך לפני תחילת העבודה!

<u>דגשים להגשה</u>

- ניתן להגיש עבודה זו בזוגות – רק אחד מהסטודנטים יגיש את העבודה באתר. בכל קובץ יש לציין שם ומס' ת.ז. של מגיש\ים, בתוך תיעוד ה javadoc.
- יש להגיש את העבודה כתיקייה מכווצת בפורמט zip. כאשר שם הקובץ המוגש מורכב ממספרי ת.ז. של המגישים, מופרדים ע"י קו תחתון. לדוגמה: 123456789_987654321.zip
- חלק מניקוד העבודה מתבצע ע"י בדיקות אוטומטיות ולכן חשוב מאוד להגדיר את כל המחלקות, המתודות והשדות בדיוק כפי שצוינו במסמך.
- על הקבצים המוגשים יש לעבוד בצורה מדויקת עם קובץ Program אותו תקבלו יחד עם המשימה תוך כדי התאמה מלאה לדוגמאות ההדפסה המסופקות.
- לכל שאלה לגבי העבודה ניתן לפנות למתרגלת במייל krsofi@gmail.com
- חובה לתעד כל קובץ, מחלקה ופונקציה ע"י javaDoc - ניתן להיעזר בתיעוד באתר oracle או בקבצים הרלוונטיים באתר.

<u>דגשים לעבודה זו</u>

- בעבודה זאת אתם מתבקשים לבצע שינויים בקוד אותו הגשתם בחלק הראשון של הפרויקט.
- <u>על כל השדות בכל המחלקות להיות פרטיים בלבד. גישה אליהם תתבצע בעזרת get/set .</u>
- <u>לכתוב בכל מחלקה את הבנאים הדרושים כך שתעבוד המחלקה הראשית Program .</u>
- על כל הקבועים להיות תחת שדות final. שימו לב, שכל ההתייחסויות במהלך הקוד לערכים ספציפיים ייעשו באמצעות גישה לקבועים הנ"ל.
- ניתן ליצור משתנים ומתודות במחלקות לשיקולכם, אבל אין לבצע שום שינוי בקובץ Program.
- דוגמאות הרצה הן <u>מחייבות</u> והפתרון שלכם צריך להפיק את אותו הפלט בדיוק פרט למקומות בהם מדובר על ערכים רנדומליים.

<u>תיאור המערכת:</u>

הפרויקט עוסק בבקרת תנועה. בתחילת המשחק נוצרת מפה, המורכבת מצמתים וכבישים שמחברים ביניהם. כמו כן, נוצרים רכבים וכל רכב מקבל מסלול רנדומלי כפי שזה עבד בחלק הראשון. לחלק מהרכבים, הנבחרים באופן רנדומלי, המסלול מוחלף אוטומטית למסלול הקצר ביותר בין אותם שני צמתים אותם חיבר המסלול הרנדומלי. הרכבים מתקדמים במסלול תוך כדי התייחסות לרמזורים בצמתים שמגיעים לנקודת היעד. לכל רמזור יש זמן השהייה, זה הזמן שלאורכו הרמזור נשאר ירוק עבור כביש כלשהו שנכנס לאותו צומת.

<u>משימות למימוש:</u>

1. מחלקת Timer
   - המחלקה סופרת זמן מתחילת המשחק. מרגע שהטיימר מופעל, הוא מדפיס מדי שניה את הזמן שעבר מתחילת המשחק. הטיימר מסיים את עבודתו ברגע שכל הרכבים מסיימים את המסלול שלהם.
2. המשחק לא מתנהל יותר בפעימות אלא עובד לפי זמן אמת וכל הזמנים שהגדרנו עד כה מתייחסים לשניות. בכל שניה הטיימר והרכבים מדפיסים את המצב שלהם. רכב, שהגיע ליעדו, מפסיק להדפיס. רמזורים מבצעים הדפסה רק כאשר הם נדלקים בפעם הראשונה ובעת ביצוע החלפת אורות.
3. במחלקת מפה מתווספת מתודה calcShortestPath, אשר מקבלת כארגומנטים שני צמתים ומחזירה את המסלול הקצר ביותר ביניהם. ניתן להיעזר באלגוריתם של Dijkstra למציאת מסלול קצר ביותר בגרף מכוון (ראו נספח).
4. במחלקת DrivingGame לאחר יצירת כל רכב, התוכנית מגרילה, האם להחליף לו את המסלול למסלול קצר ביותר. ההגרלה מתבצעת בתנאי שצומת ההתחלה והסיום במסלול המקורי הם לא אותו צומת (שהמסלול המקורי אינו מעגלי).
5. תנועת הרכבים מתבצעת מעכשיו תוך כדי התייחסות לרמזורים. אם רכב הגיע לצומת מרומזר והכביש ממנו הגיע אינו תחת אור ירוק ברגע זה, הרכב ימתין עד שהאור הירוק יאיר לכביש שבו הוא ממתין, וידפיס הודעה מתאימה (ראו דוגמת הרצה).
6. כל אחד מהרכבים, הרמזורים וגם הטיימר אמורים לעבוד כתהליכונים נפרדים לאורך כל ההרצה, וכל אחד מהם מבצע את ההדפסות הקשורות אליו. כאשר כל הרכבים במשחק מסיימים את המסלול, הרמזורים והטיימר מסיימים את עבודתם והמשחק מסתיים.

עבודה נעימה!

נספח: הסבר על האלגוריתם של Dijkstra למציאת המסלול הקצר ביותר בין שתי נקודות בגרף.

1. המפה במשחק היא למעשה גרף מכוון. כל צומת הוא קודקוד בגרף, וכל כביש – צלע בגרף.
2. לכל צלע נתון משקל אי-שלילי (במקרה שלנו אורך הכביש).
3. לטובת מימוש האלגוריתם נצטרך מערך של מרחקים עבור כל אחד מהצמתים. הערכים במערך זה יתייחסו למרחק של כל הצמתים מצומת ההתחלה של המסלול. עבור צומת ההתחלה מרחק זה יאותחל ל-0, עבור שאר הצמתים – לאינסוף. (ניתן להשתמש ב- Double.MAX_VALUE).
4. בנוסף, נצטרך מערך השומר את "דרכי הגעה" (הדרך שממנה הגענו לצומת הנוכחי במסלול הקצר ביותר שמצאנו עד כה). את הערכים במערך נאתחל ל-null.
5. במהלך הריצה של האלגוריתם נצטרך לסמן את הצמתים בהם כבר טיפלנו. ניתן לעשות זאת ע"י מערך נוסף של ערכים בוליאניים.
6. כל עוד קיים צומת שלא נבדק, נבצע את הפעולות הבאות:
   - בין הצמתים שלא נבדקו נמצא את הצומת עם המרחק המינימלי.
   - עבור כל אחת מהדרכים היוצאות של צומת זה נבצע:
     - אם המרחק של צומת הסיום של הדרך גדול ממרחק צומת ההתחלה בתוספת אורך הדרך:
       - נחליף את מרחק צומת הסיום לסכום של מרחק צומת ההתחלה ואורך הדרך.
       - נשמור את הדרך כ"דרך הגעה" של צומת סיום.
   - נסמן שסיימנו את הבדיקה של הצומת.
7. קיבלנו עבור כל צומת ערך של "צומת קודם" ממנו ניתן להגיע לצומת הנוכחי בדרך הקצרה ביותר. כעת נוכל לשחזר את המסלול מהסוף להתחלה:
   - נתחיל מצומת הסיום ונשמור את דרך ההגעה שלו כדרך אחרונה במסלול שלנו.
   - נתייחס לצומת ההתחלה של דרך זו ונשמור את הדרך ההגעה שלה כדרך הלפני אחרונה במסלול שלנו.
   - נחזור על פעולות אלה עד שצומת ההתחלה של דרך שהוספנו תהיה צומת תחילת המסלול (שדרך ההגעה שלה היא null).