תכנות מונחה עצמים מתקדם
עבודת הגשה מס ' 1

מבוא

זהו התרגיל הראשון בקורס ומהווה בסיס לסדרת תרגילים שיינתנו בהמשך הקורס, בנושא בקרת תנועה.

מטרת התרגיל היא כתיבת מחלקות המייצגות דרכים, צמתים וכלי רכב.

נא לקרוא את כל המסמך לפני תחילת העבודה!

<u>דגשים להגשה</u>

- ניתן להגיש עבודה זו בזוגות – רק אחד מהסטודנטים יגיש את העבודה במודל. בתיעוד בכל קובץ יש לציין שם ות.ז. של מגיש\ים, בתוך תיעוד ה javadoc.
- חלק מניקוד העבודה מתבצע ע"י בדיקות אוטומטיות ולכן חשוב מאוד להגדיר את כל המחלקות, המטודות והשדות בדיוק כפי שצוינו במסמך.
- על העבודה לעבוד בצורה מדויקת עם קובץ ה program שתקבלו בהמשך.
- לכל שאלה לפנות למתרגל האחראי על עבודה ראשונה – סופיה קרימברג במייל sofiak@ac.sce.ac.il. נושא ההודעה חייב להיות "aoop20_hw1".
- חובה לתעד כל קובץ, מחלקה ופונקציה ע"י javaDoc - ניתן להיעזר בתיעוד באתר oracle או בקבצים הרלוונטיים במודל.

<u>דגשים לעבודה זו</u>

- עבודה זו מהווה בסיס לשאר העבודות בקורס אך אינה מכילה עקרונות בסייסים בתכנות מונחה עצמים, ולכן חלק מהדרישות יעמדו בסתירה לעקרונות מתקדמים שלמדנו. בחלקים מסויימים המבנה אינו נכון במכוון (או אינו הנכון ביותר), ע"מ שבעבודות הבאות תוכלו להבין את חשיבות התכנות הנכון.
- אין להשתמש בהורשות או ממשקים.
- על כל העבודה להיות פרויקט יחיד המחולק לpackages לפי המטלות.
- <u>על כל השדות בכל המחלקות להיות פרטיים בלבד. גישה אליהם תתבצע בעזרת get/set .</u>
- <u>לכתוב בכל מחלקה את הבנאים הדרושים כדי שתעבוד המחלקה הראשית Program .</u>
- על כל הקבועים להיות תחת שדות final. שימו לב, כלל ההתייחסויות במהלך הקוד לערכים ספציפיים ייעשו באמצעות גישה לקבועים הנ"ל.

1. מחלקות עזר: package: utilities

1.1. Program – המחלקה הראשית

1.2. Point – מגדירה מיקום על ציר דו מימדי. הצרים נעים בין הערכים הבאים (יש לשמור את ערכים אלו **ككבועים במחלקה**).

1.2.1. Fields:

- double x נע בין 0 ל1000000
- double y נע בין 0 ל 800

1.2.2. Constructors:

- Point(double, double)

1.2.3 Methods

- Getters & Setters
- toString()
- equals()

2. רכיבים: package: components

2.1 Junction – מייצגת את הצמתים על מפת המשחק. הצמתים יכולים להיות מרומזרים ולא מרומזרים, כאשר התנועה בצומת מתנהלת באופן הבא: אם הצומת לא מרומזרת, העדיפות היא לפי רשימת הדרכים הנכנסות, כלומר לדרך הראשונה ברשימת הדרכים הנכנסות עדיפות גבוהה ביותר, לאחרונה – עדיפות נמוכה ביותר. אם הצומת מרומזרת, ניתן לחצות אותה כאשר הדרך היוצאת הרצויה מקבלת אור ירוק ברמזור.

2.1.1 Fields:

- junctionName: String
- location: Point // *location of the junction on the map*
- enteringRoads: ArrayList <Road> *// holds the list of the roads that enter to the junction.*
- exitingRoads: ArrayList <Road> *// holds the list of the roads that exit the junction.*
- hasLights: boolean *// checks if the junction has traffic lights.*
- delay: int *// delay time in seconds*
- vehicles: ArrayList<Road> *//list of entering roads with cars waiting on the junction*

2.1.2 Constructors:

- public Junction (String name, Point loc);

2.1.3 Methods:

- getters & setters
- changeLight() *//make the next entering road in the list green (open) and //all the others (exiting only) red (closed).*

- checkAvailability (Road r) *//for vehicle that arrived to the junction //from road r, checks if there are some other vehicles on the roads with //a higher traffic priority on the junction.*
- toString()
- equal()

2.2. Road – מייצגת את קטעי דרך המחברים בין הצמתים. כל קטע מותאם לכלי תחבורה מסויימים.

2.2.1 Fields:

- fromJunc: Junction
- toJunc: Junction
- allowedVehicles: ArrayList< String> *// holds the list of vehicle types //that are allowed to move on the road.*
- isOpen: boolean *// True when the light is green.*
- isEnabled: boolean *//appears on the map*
- length: double *// the distance between the two junctions.*
- maxSpeed: int

2.2.2 Constructors:

- public Road (Junction from, Junction to) *//creates an instance of the //class and sets the values of allowedVehicles, isOpen and isEnabled //randomly. Sets the length value to the calculated one.*
- public Road (Junction from, Junction to, ArrayList<String> allowed, boolean open, boolean enabled)

2.2.3 Methods:

- getters & setters
- addVehicleType(String type)
- countLength() *//calculates the length of the road using the coordinates //of its junctions.*
- toString()
- equal()

2.3. Route – מייצגת מסלול המחבר בין צומת לצומת, מסלול הוא אוסף של צמתים וקטעי דרך.

2.3.1 Fields:

- junctions: ArrayList<Junction> *// list of junctions on the route by the //order of movement.*
- roads: ArrayList<Road> *// list of roads on the route by the order of //movement*
- delay: double *//time that will take the vehicle to make this route.*
- vehicleType: String

2.3.2 Constructors:

- public Route(ArrayList<Junction> juncs, ArrayList<Road> roads, String vehType)
- public Route(Junction start, Junction end, String vehType) *// not implemented in this task.*

2.3.3 Methods:

- getStart()
- getEnd()
- calcDelay() *//set length to be a sum of delay values of all the junctions on the route //and the time that will take this type of vehicle to pass all the roads. Time is calculated by dividing the distance by min(average speed, maxSpeed). The delay time on junctions is calculated according to worse case: if there is a traffic lights on the junction, we use it's delay value multiplied by (number of entering roads minus one). If there is no traffic lights on the junction, the delay time is the priority level of the road that leads us to this junction (the index of this road in the list of roads).*

2.4. Map – מייצגת את המפה המכילה את כל הצמתים וקטעי הדרך.

2.4.1 Fields:

- junctions: ArrayList<Junction>
- roads: ArrayList<Road>

2.4.2 Constructors:

- public Map() *//Creates a map with 20 random junctions and connects all of them one to another with roads.*
- public Map (int junctions, int roads) *//Creates a random map with given quantity of junctions and roads.*
- public Map (ArrayList<Junction> juncs) *//Creates a map with given junctions and connects all of them with roads.*
- public Map (ArrayList<Junction>juncs, ArrayList<Road>roads) *//Creates a map with given junctions and roads.*

2.4.3 Methods:

addRoad(Road r)

removeRoad(Road r)

addJunction(Junction junc)

removeJunction(Junction junc) *//removes the junction and all connected to it roads //from the map.*

Vehicles 2.5 – מייצגת את הרכבים הנעים על המפה.

2.5.1 Fields:

- id: int
- type: String
- speed: int *//average speed for this type of vehicle.*
- currentRoute: Route
- lastJunction: Junction *//current junction or last junction where the //vehicle visited*
- lastRoad: Road
- movesNow: boolean *//True if the vehicle is on the road between the //junctions.*
- spentTime: double *//time passed from the beginning of movement //on the route.*

2.5.2 Constructors:

- public Vehicle(int id, String type, Junction lastJunction)

2.5.3 Methods:

- getters & setters
- move() *// wait for the current point delay time and move to the next //point of the route*.
- status() *//prints the details about the vehicle including current //position, time spent on the route and the first and last junctions on the route.*
- checkIn() *//if arrived to a junction, update the junction waiting list //and calculate the delay time before the next move*.

VehicleType 2.6

2.6.1 Fields:

- typeName : String
- speed: int *//average speed of vehicle type*

2.6.2 Constructors:

- Public VehicleType(String name, int speed)

2.6.3 Methods:

- toString()
- equal()

3. משחק: package: game
Driving 3.1

3.1.1 Fields:

- numOfJuncs: int
- numOfVehicles: int
- currentMap: Map
- currentVehicles: ArrayList<Vehicle>
- drivingTime: double *// time passed from the beginning of driving //session*
- maxTime: int *// total round time*

3.1.2 Constructors:

- public Driving(int juncs, int vehicles, int maxTime)

3.1.3 Methods:

- getters & setters
- addMap() *//creates a map with random (10-25) junctions quantity.*
- addVehicles() *//creates random number (2-8) of vehicles of different types.*
- startDrive(int maxTime): void
- toString()
- equal()

עבודה נעימה!