**תכנות מונחה עצמים מתקדם**

**עבודת הגשה מס' 2**

**להגשה עד ה 18/04/19 ב**  **23:55-**

**דגשים להגשה**

* ניתן להגיש עבודה זו בזוגות – רק אחד מהסטודנטים יגיש את העבודה במודל. בתיעוד של קובץ יש לציין שם, ת.ז וקמפוס של מגיש\ים, בתוך תיעוד ה-javadoc
* לכל שאלה אנא עברו על מסמך ה-[FAQ](https://docs.google.com/document/d/1HMrVvBNH6an6Lb3saF6PYrLMQyeU0vNN-blXaBaX76E/edit?usp=sharing) ובדקו האם היא נענתה כבר. במידה ולא, ניתן להוסיף הצעת עריכה בקובץ או לפנות למתרגל האחראי – שירז בר, במייל [shirazbarsce@gmail.com](mailto:shirazbarsce@gmail.com)  
  **על כל פניה** להכיל את פרטי הסטודנט המלאים כולל ת.ז, קמפוס, ושם המתרגל.
* חובה לתעד כל קובץ, מחלקה ופונקציה ע"י javaDoc  
  ניתן להיעזר [בתיעוד באתר oracle](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html) או בקבצים הרלוונטיים במודל
* העבודה מתבססת על עבודת הגשה 1 – עליכם לעדכן/להרחיב את המחלקות הקיימות במידת הצורך ולהשתמש בהן. כל המחלקות החדשות ניתן לארגן בחבילה בשם-graphics

1. **מבוא**

זהו התרגיל השני בקורס, בתרגיל זה נתרגל יישום של ממשקים עם שימוש ב-GUI.

מטרת התרגיל:

* בניית סביבת **GUI**.
* יישום ממשקים.
* שימוש ב-Exceptions.

יש לבצע את העבודה לפי הדרישות המצוינות במסמך הנתון. בעבודה זאת יש לממש את ממשק GUI לניהול **עיר המכילה כלי רכב** ע"י שימוש בפעולות המוגדרות עליו.

**שינויים מעבודה מס' 1:**

* רכב שמונע בסולר יהיה בעל צריכת דלק של 1 ליטר פר קילומטר
* רכב שמונע בבנזין יהיה בעל צריכת דלק של 2 ליטר פר קילומטר
* מהירותו של כלי רכב מסוג Car הינה 4 קמ"ש

1. **הגדרות כלליות:**

**אתם מייבאים את כל המחלקות ישר מן התרגיל הקודם.**

בנוסף יש להגדיר את ממשקים ומחלקות (ב- package **graphics**):

* public class **CityFrame** extends JFrame – המחלקה תכלול פונקציית ***main***
* public class **CityPanel** extends JPanel
* public class **AddVehicleDialog** extends JDialog
* public interface **IDrawable**
* public interface **IClonable**
* **IMoveable** public interface
* **IMoveable** public interface **IAnimal** extends
* **Public class PackAnimal implements IAnimal, IClonable**
* רק את מחלקה זו יש להגדיר בpackageVehicle

1. ממשק **IDrawable** צריך להגדיר בצורה הבאה:

**public** **interface** IDrawable {

**public** **final** **static** String ***PICTURE\_PATH*** = "…";

**public** **void** loadImages(String nm);

**public** **void** drawObject (Graphics g);

**public** String getColor();

}

1. ממשק **IMoveable** צריך להגדיר בצורה הבאה:

**public** **interface** IMoveable {

**public** String getVehicleName();

**public** **int** getSpeed();

**public** **int** getFuelConsumption();

**public boolean** move(Point p);

}

1. ממשק **IAnimal** צריך להגדיר בצורה הבאה

**public** **interface** IAnimal **extends** IMoveable{

**public** String getAnimalName();

**public boolean** eat();

}

1. ממשק **IClonable** : ניתן לעשות שימוש במה שקיים ב-Java.

מחלקות Point, Engine יממשו את הממשק הנ"ל(בנוסף למחלקות שיצוינו בהמשך).

1. מחלקת **PackAnimal - יש להגדיר אותה בתוך ה-Package של הרכבים.**

צריך להגדיר בצורה הבאה:

המחלקה מממשת את ממשק **IAnimal** ו- **IClonable** ובנוסף מכילה את הנתונים הבאים:

- שם החיה.

-רמה מקסימלית קבועה של אנרגיה היא 1000

-כאשר חיה אוכלת רמת האנרגיה שלה מתעדכנת להיות מקסימלית.

- עבור כל קילומטר שהחיה זזה רמת האנרגיה שלה יורדת ב-20.



1. מחלקת Carriage תעודכן באופן הבא:

במקום להכיל מחרוזת של סוג בע"ח, היא תכיל אובייקט מסוג **PackAnimal .**



1. בנוסף להגדרה של **Vehicle**  מתרגיל הקודם: בתרגיל הזה **Vehicle**  מממש גם את

**, IDrawable IMoveable, IClonable**

**public** **abstract** **class** Vehicle **implements** IMoveable, IDrawable, IClonable { … }

במחלקה **Vehicle** יש להגדיר את השדות נוספים הבאים:

**protected** **int** size;

**protected int** id;

**protected Boolean** lights;

**protected** Color col;

**protected** **int** wheels;

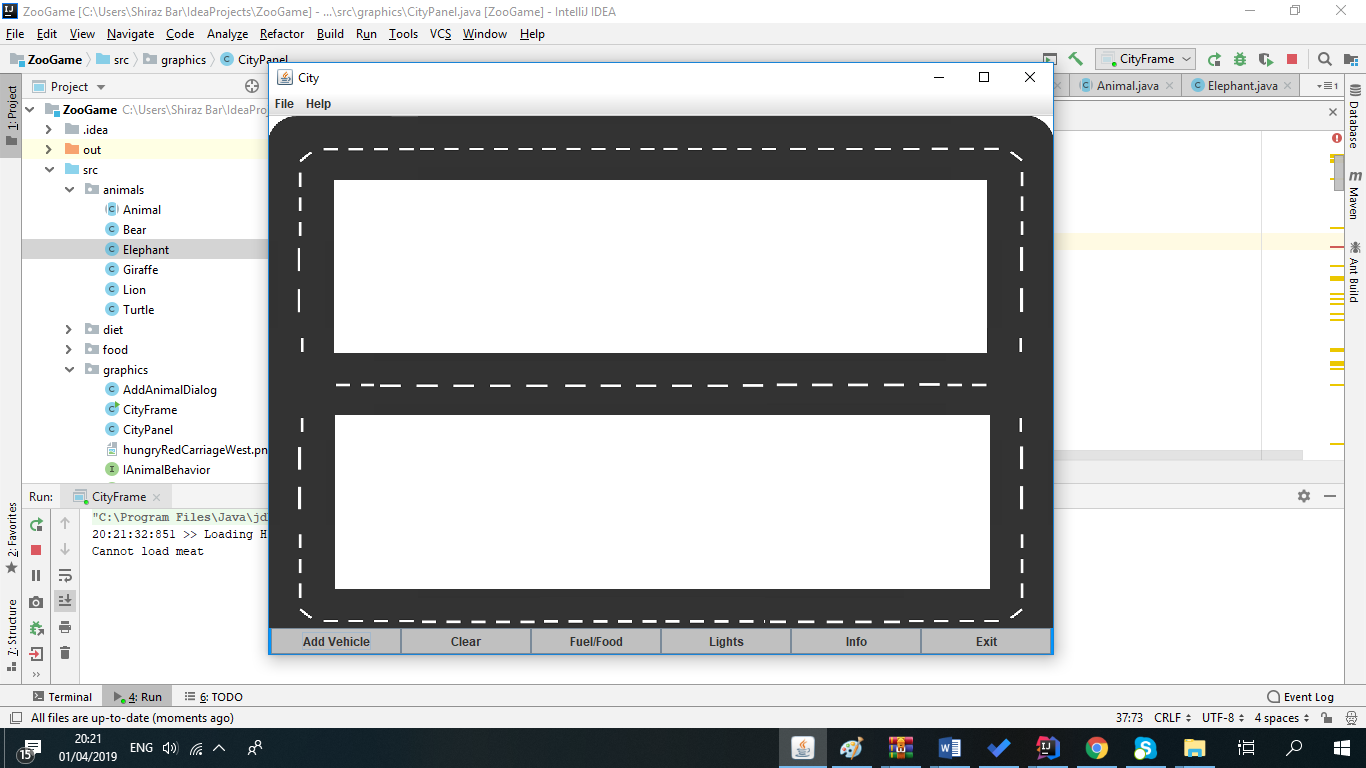
**protected** Location loc;

**protected** **int** fuelConsumption; (כמות הדלק סה"כ שבוזבזה)

**protected** CityPanel pan;

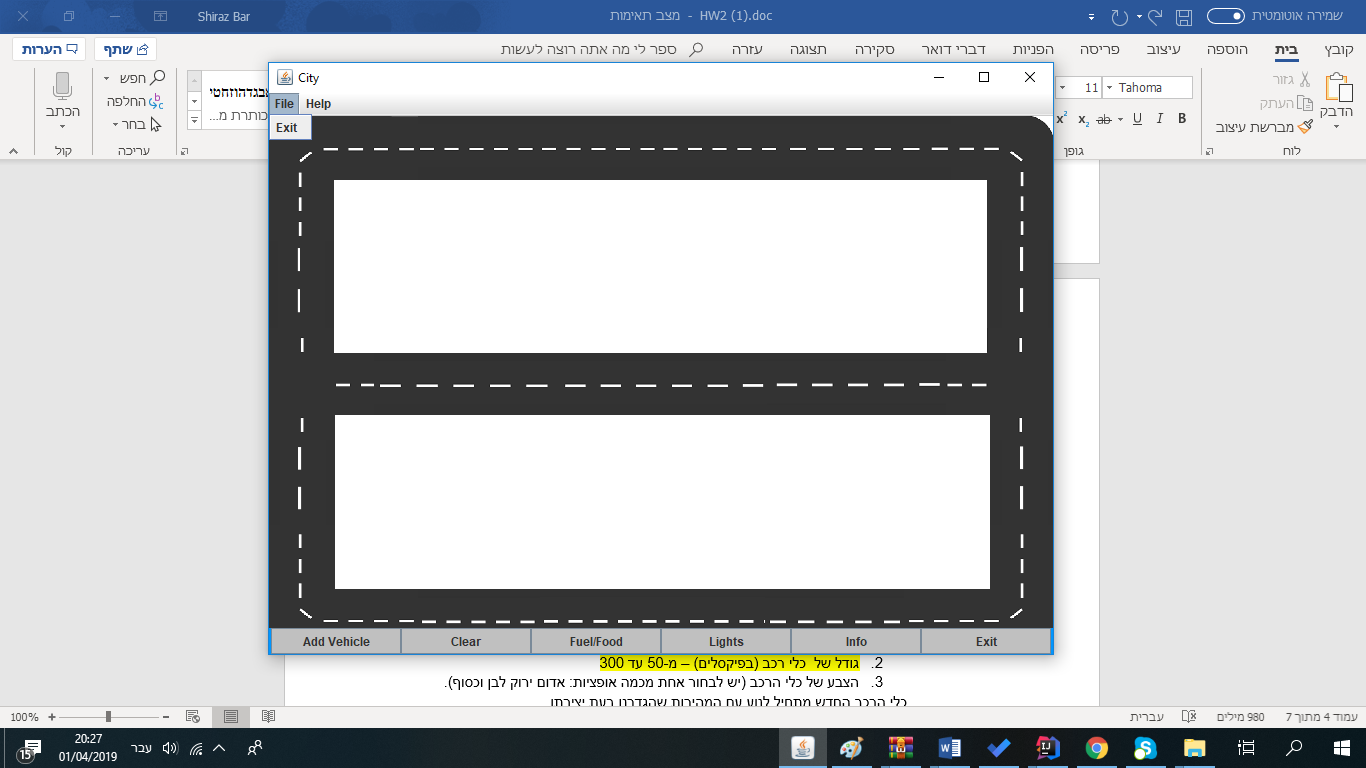
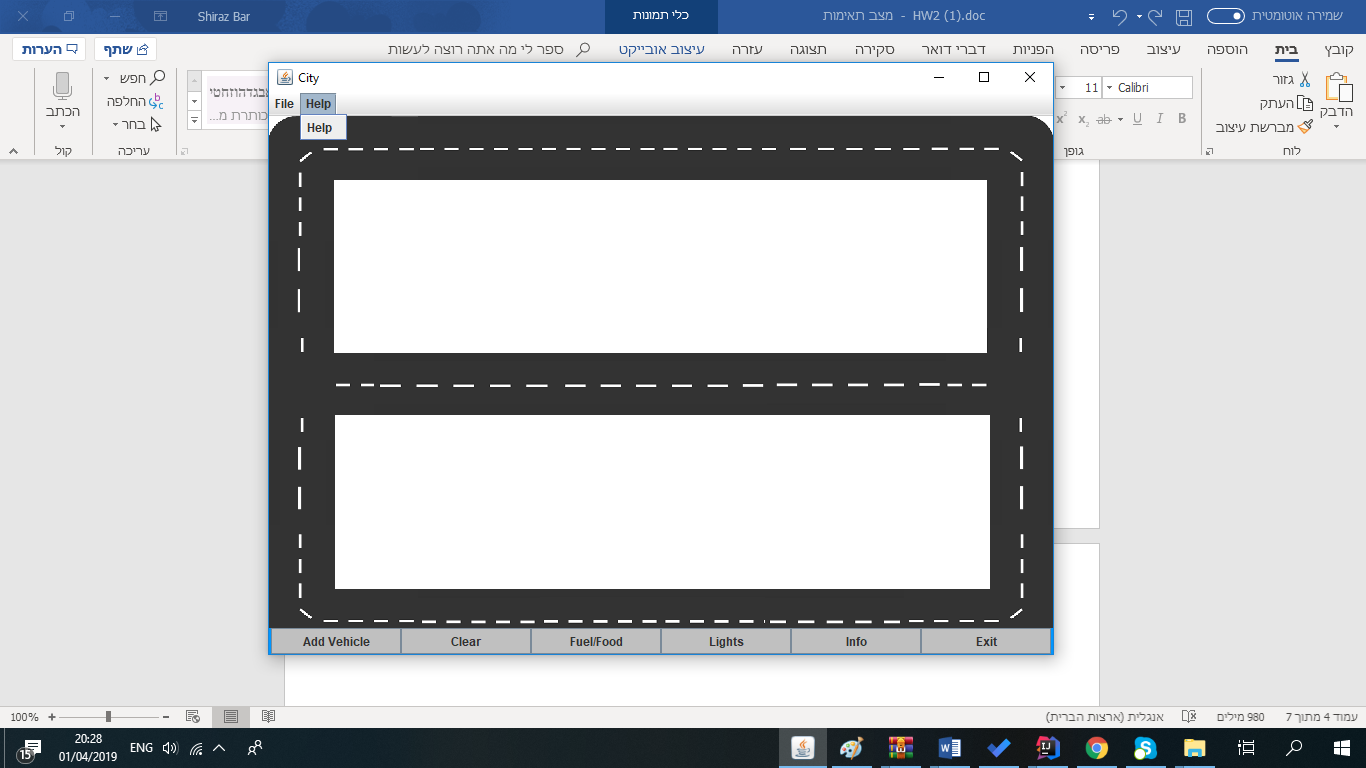
**protected** BufferedImage img1, img2, img3, img4;

במחלקות Vehicle, hasEngine ולכל מחלקות של חיות יש להגדיר בנאים נוספים לאתחול שדות וגם להוסיף פונקציות לפי צורך. **ניתן להרחיב גם מחלקות וממשקים אחרים.**

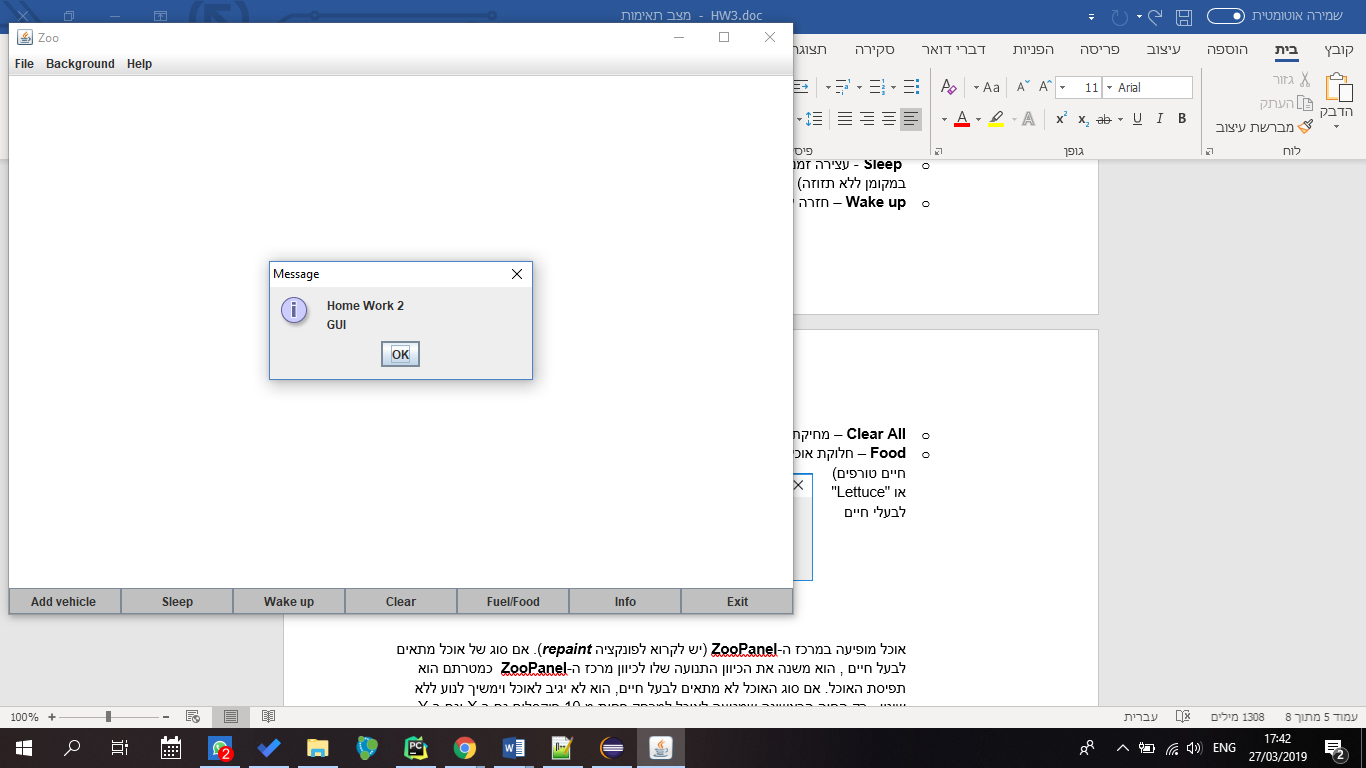
1. **הגדרות ה-GUI עבור התרגיל :**

* ה- **CityFrame** כולל menu ו**CityPanel** עם מספר כפתורים הנמצאים בתחתית הפאנל.
* ה-Background מעודכן להיות תמונת הכבישים עם תחילת התכנית
* כל השטח ה- **CityPanel**מיועד לציורי כלי הרכב.

ב- **menu** יש להגדיר את האפשרויות הבאות:



* + - **Exit** – סגירה של התכנית
    - **Help** – לפתיחת *showMessageDialog* בצורה הבאה:



* **הכפתורים** שנמצאים בתחתית ה- **CityPanel** מבצעים את הפעולות הבאות:
* **Add Vehicle** – פותח חלון של **AddVehicleDialog**, שמאפשר להגדיר כלי רכב חדש עם פרמטרים הבאים:

1. בחירת סוג של רכב: כרכרה, אופניים, רכב ממנוע בבנזין, רכב ממונע בסולר.
2. הצבע של כלי הרכב (יש לבחור אחת מכמה אופציות: אדום ירוק לבן וכסוף).

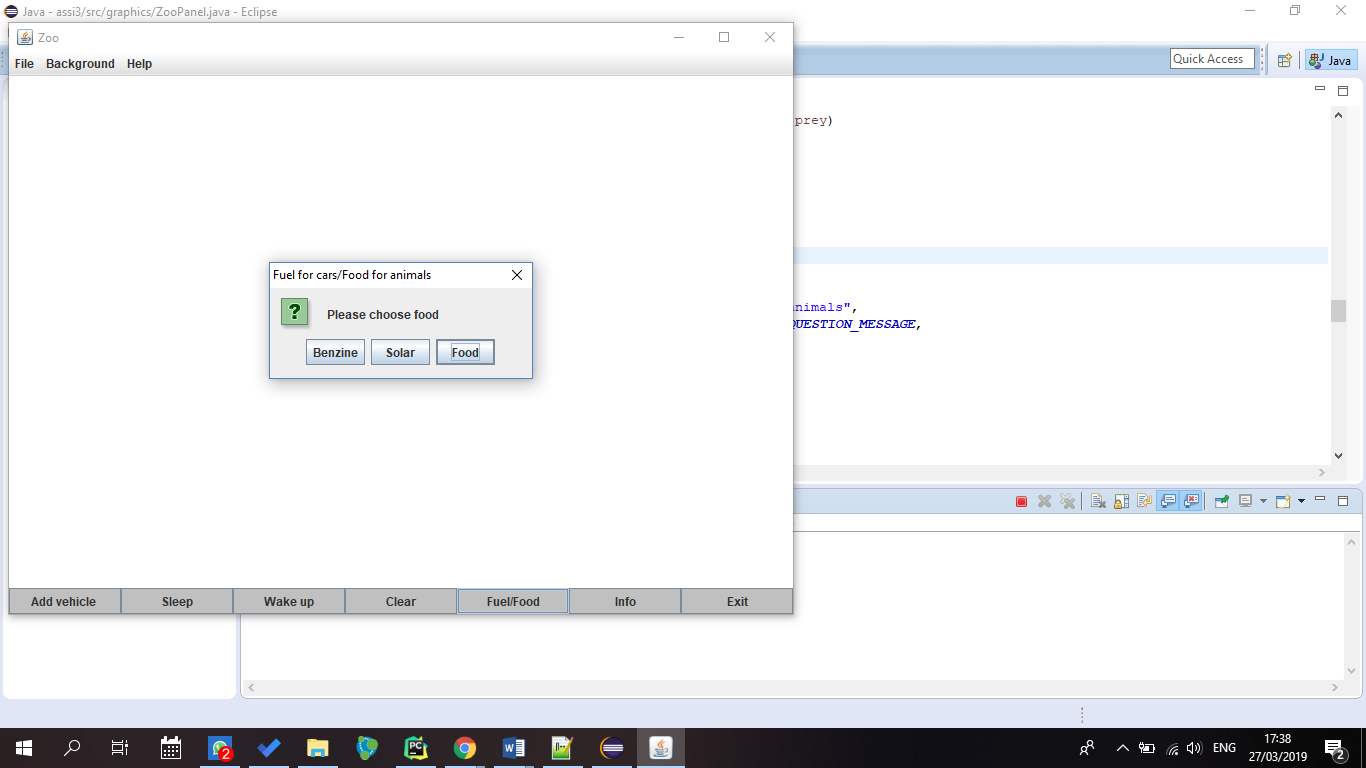
כלי הרכב החדש מתחיל לנוע עם המהירות שהגדרנו בעת יצירתו.

המיקום ההתחלתי עבור כל כלי הרכב הוא (0,0). כל הרכבים בעת יצירתם מתחילים את

תנועתם לכיוון מזרח. את כל הרכבים מציירים על המסך בגודל של 65 פיקסלים.

**Clear** – מחיקת כלי הרכב מהפאנל.

* **Fuel/Food** – חלוקת דלק/אוכל. יש להוסיף אפשרות לבחור סוג של הזנה: "Benzine" או Solar (שמתאים לכלי רכב ממונעים) או “Food” (שמתאים לכרכרה):



במידה וסוג ההזנה לא מתאים עבור כלי הרכב שנמצא על המסך יש לזרוק חריגה שתוביל להצגת *showMessageDialog עם הודעה מתאימה.*

*אחרת, במידה וסוג האוכל תקין יש להפעיל מתודות refuel עבור הרכב או eat עבור החיה.*

* **Lights** – משנה את המשתנה Lights להיות True במידה והיה False ולהפך.
* **Info** – מופיע כל המידע על כלי הרכב בצורת ה-JTable (בחלון נפרד או ב-( **CityPanel** :

Vehicle: שם כלי הרכב.

ID: מספר מזהה של הרכב

Color: צבע הרכב

Wheels: מספר הגלגלים.

Speed: מהירות הרכב

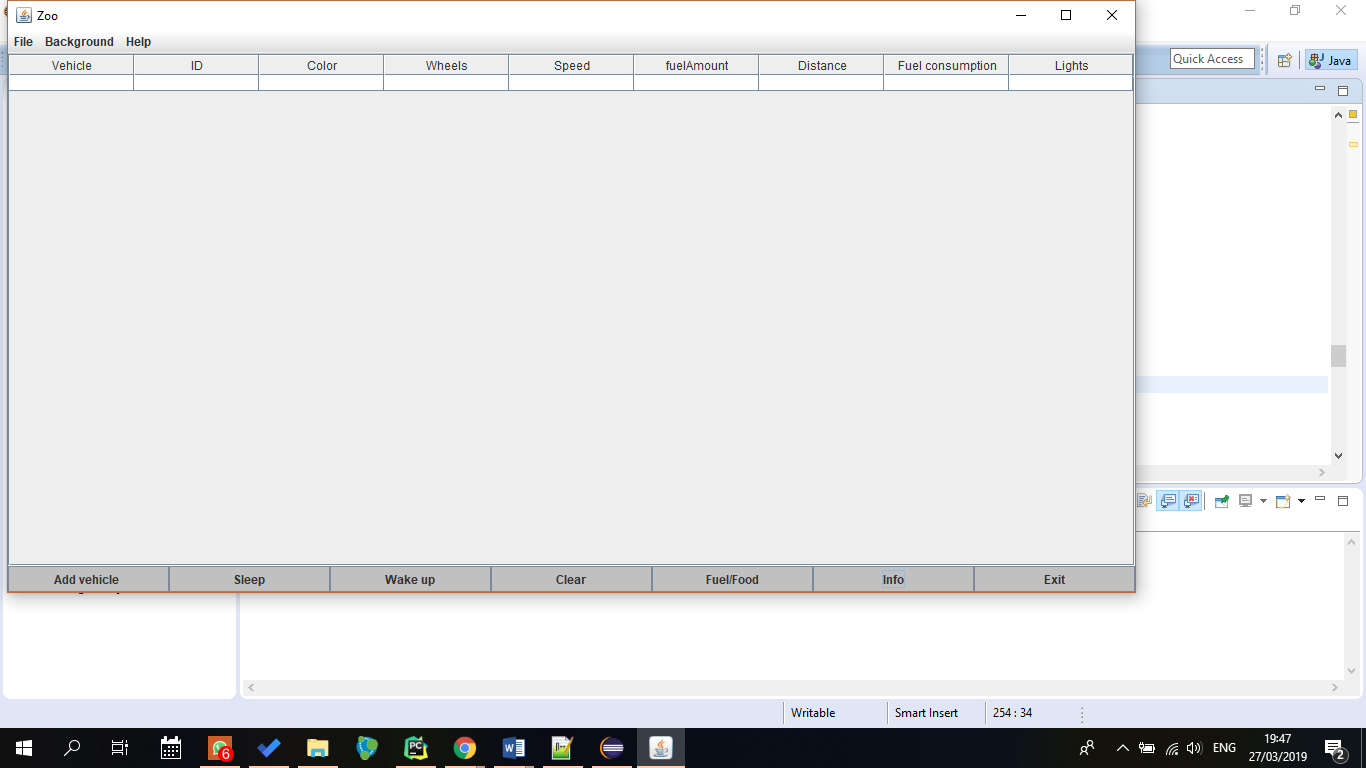
fuelAmount: כמות הדלק הנוכחית במיכל.(תתעדכן לאחר כל תזוזה)

Distance: סה"כ המרחק שכלי הרכב עבר

Fuel consumption: סה"כ כמות הליטרים/האנרגיה שכלי הרכב קיבל.(לאחר כל תדלוק כמות הליטרים /האנרגיה שקיבלו מתווספים לסכום).

Lights- True במידה והאורות דולקים. אחרת, False.

כאשר רכב נמחק מהפאנל, המידע השמור עליו יישאר טבלה כפי שהיה לפני שנמחק.



* **Exit** –סגירה מסודרת של התוכנית.

1. **הגדרות הגרפיקה עבור התרגיל:**

* לכל כלי הרכב יש לממש פונקציית **drawObject()** (המוכרזת בממשק **IDrawable**) ולקרוא לה ב**-CityPanel.paintComponent()**.

בשביל ציורי כלי הרכב יש להשתמש ב- **BufferedImage** . דוגמא של שימושו:

1) **הגדרת image**:

**private BufferedImage img=null;**

2{)  **טעינה של תמונה**:

**try { img = ImageIO.read(new File(BACKGROUND\_PATH)); }   
  catch (IOException e) { System.out.println("Cannot load image"); }**

פה  BACKGROUND\_PATH זה full path של קובץ, למשל "*D://image.png*" .  
באופן דומה טוענים תמונות לרכבים.

3) **ציור של תמונה כרקע**

**public void paintComponent(Graphics g) {  
     super.paintComponent(g);   
     g.drawImage(img,0,0,getWidth(),getHeight(), this);  
     .....**

**{**

4) **לכל כלי רכב יש 4 שדות מסוג** **BufferedImage** :

**img1** לצורך תנועה לכיוון צפון

**Img2**  לצורך תנועה לכיוון דרום

**Img3**  לצורך תנועה לכיוון מזרח

**Img4**  לצורך תנועה לכיוון מערב

דוגמא של פונקציה **drawObject (Graphics g)** במחלקה **Car** :

**public** **void** drawObject (Graphics g)

{

g.setColor(col);

**if**(orientation==***NORTH***) // car drives to the north side

g.drawImage(img1, location.x-size/2, location.y-size/10, size, size\*2, pan);

**else if(**orientation==***SOUTH)*** // car drives to the south side

g.drawImage(img2, location.x, location.y-size/10, size, size, pan);

**else if(**orientation==***EAST)*** // car drives to the east side

g.drawImage(img3, location.x, location.y-size/10, size\*2, size, pan);

**else if(**orientation==***WEST)*** // car drives to the west side

g.drawImage(img4, location.x, location.y-size/10, size\*2, size, pan);

}

בדוגמא הזו משתמשים במשתנים הבאים:

* **col** – צבע של כלי הרכב (protected data member של מחלקת **Vehicle**)
* **size** –גודל של כלי הרכב (protected data member של מחלקת **Vehicle**)
* **location.x** ו- **location.y**- קואורדינטות של "פה" כלי הרכב (**protected Point location** הוא שדה של מחלקת **Vehicle**  , הקואורדינטות האלה משתנים בפונקציית **drive.**

1. **הגדרות תנועת הרכבים על הכבישים:**
   * כלי הרכב ינוע באופן הבא: התזוזה תתבצע בהתאם למהירות הרכב- כלומר שינוי הקואורדינטות יהיה לפי המהירות המתאימה.
   * על מנת שהנסיעה תבוא לידי ביטוי בצורה ויזואלית יש להשתמש בפונקציה sleep של מחלקת Thread למשך 100 מילישניות לפני כל שינוי מיקום.
   * כאשר קדמתו של הרכב נמצא באחת הקצוות של המסך(דורש חישוב מתאים), על הרכב לשנות את כיוונו לפי כיוון הכביש וכמו כן תמונת הרכב צריכה להשתנות בהתאם.
   * כאשר קדמתו של הרכב נמצא באחד הצמתים שנמצאים באמצע המסך, עליו להגריל את המשך כיוונו. לדוגמא: אם הרכב נוסע דרומה בכביש הימני ביותר על המסך ומגיע לצומת שבאמצע, עליו להגריל האם להמשיך דרומה או לשנות את כיוונו מערבה. הסיכויים הם 50-50 בכל המקרים האפשריים שבהם רכב מגריל את כיוונו.
   * ברגע שלכלי רכב נגמר הדלק/ לחיה נגמרת האנרגיה – מפסיקה תנועתם על הכביש והם עומדים במקום.

**התרגיל הזה ישמש כבסיס לתרגיל הבא, לכן יש להשקיע בכתיבת קוד איכותי. בתרגיל הבא תתבקשו להרחיב את המימוש ולהוסיף תבניות עיצוב שונות.**

***עבודה נעימה***