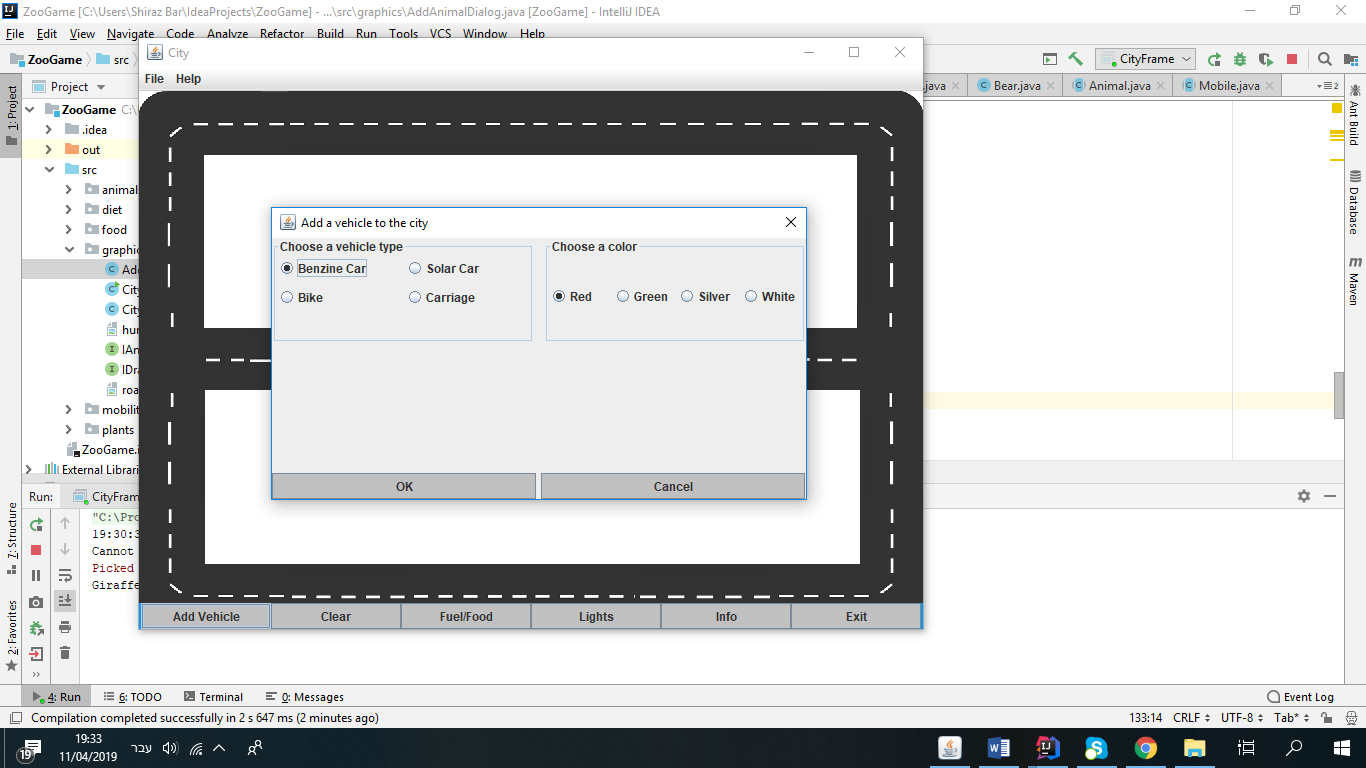
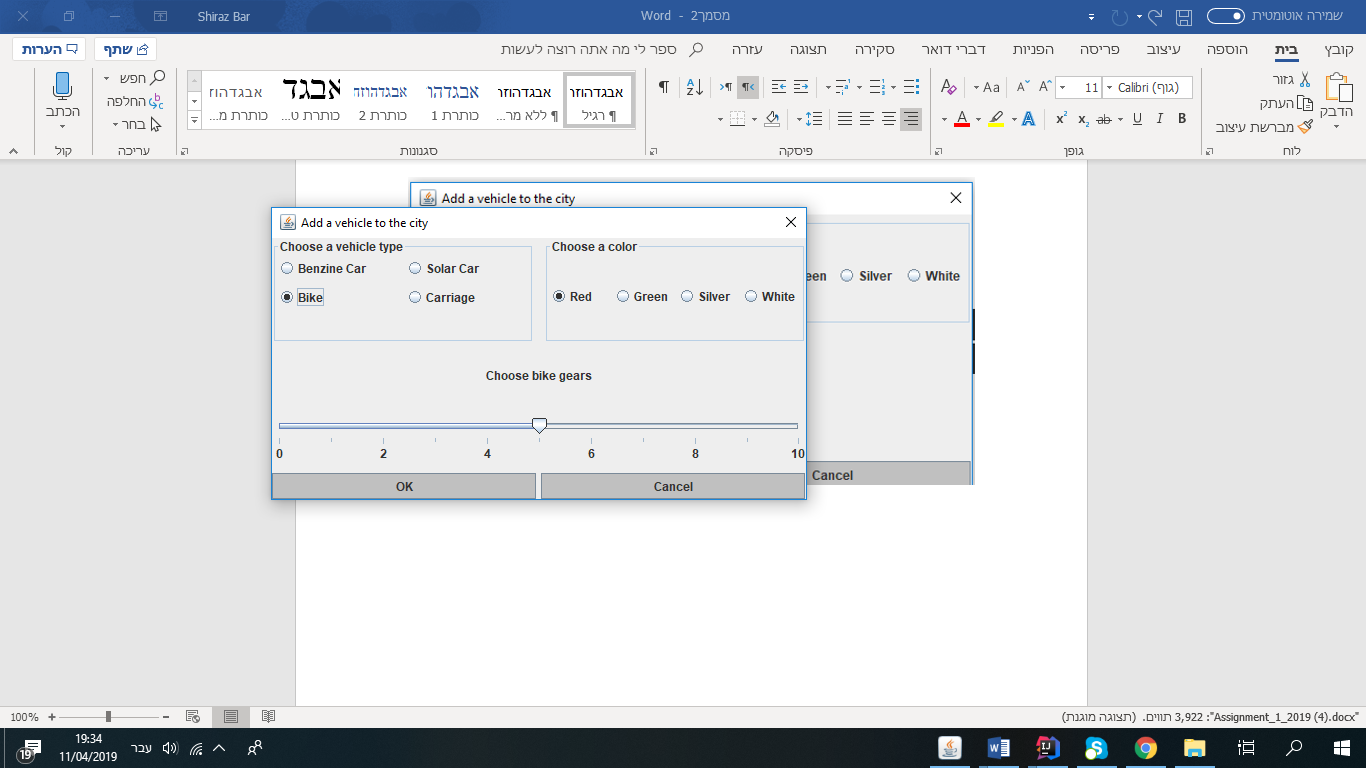
**תכנות מונחה עצמים מתקדם עבודה מס' 2 – הבהרות**

* עליכם להגדיר את הCityFrame שאתם בונים להיות ברוחב 800 ובגובה 600 כך:

this.setSize(800,600);

* אין חובה לממש את ה-Gui בצורה שמפורסמת, אבל מומלץ אם תרצו. מי שכבר מימש בדרך אחרת, יכול להשאיר את זה כך אך עליו לוודא שהוא מקבל מהשתמש אך ורק את הנתונים הבאים.
* הנתונים המופיעים במסך אלו הם הנתונים היחידים שנקלטים מהמשתמש בתוך AddVehicleDialog:



* + במידה ומשתמש בחר להוסיף אופנים יש לתת לו את האפשרות לבחור כמה הילוכים הוא רוצה ולפיכך לאתחל את אובייקט האופניים. ההילוך הוא מספר שלם בין 1 ל-10.
* לאחר שהמשתמש לוחץ על OK, עליכם לבנות אובייקט של כלי הרכב בתוך מחלקת AddVehicleDialog ולאחר מכן לעדכן את הCityPanel שזה אובייקט הרכב שלו, ולהפעיל את פונקציית הmove של כלי הרכב.
* על כל שאר הפרמטרים שנמצאים במחלקת Vehicle להיות מאותחלים באופן דיפולטי הבא:
  + השדה lights מאותחל בבנאי להיות false.
  + שדה הקילומטראז' מאותחל בבנאי להיות שווה ל-0 ומשתנה במהלך התכנית בהתאם לנסיעת הרכב.
  + מס' המושבים מוגדר עבור כל מחלקה כמשתנה סטטי קבוע של המחלקה:
    - למכונית יש 5 מושבים
    - לאופניים יש מושב אחד
    - לכרכרה יש 2 מושבים
  + גיל מינימלי מוגדר עבור כל כלי הרכב כמשתנה סטטי קבוע של המחלקה Vehicle להיות גיל 18.
  + המשתנה fuelConsumption מאותחל להיות שווה ל-0 עבור כל כלי הרכב. בכל תדלוק מוסיפים את כמות הדלק בליטרים שהרכב קיבל. אצל האופניים זה תמיד נשאר 0.
  + שם כלי הרכב, וכמו כן שם של חית משא, מאותחל עבור כל מחלקה להיות על פי שם המחלקה. אצל כלי רכב גם יוחזר סוג המנוע ביחד עם השם, כדלקמן:   
    "Solar Car", ""Benzene Car, "Bike", "Carriege", "Pack Animal".
  + מחלקת Vehicle תכיל משתנה סטטי שמאותחל להיות 1000. כל כלי רכב שייבנה יבנה את ה-ID שלו לפי המשתנה הסטטי הזה, ולאחר מכן יגדיל אותו ב-1. כלומר כלי הרכב הראשון יהיה עם תז 1000, השני יהיה עם תז 1001, וכן הלאה.
* כרגע מס' הרכבים המקסימלי על המסך הוא 1(וזה ישתנה בהמשך....). ברגע שמשתמש מנסה להוסיף יותר מרכב 1, צריך להופיע showMessageDialog שמתריע לו על כך שלא ניתן לעשות זאת.
* הפונק' loadImages מקבלת כפרמטר מחרוזת. שימו לב- אפשר לוותר על כך במידה ואתם רוצים(יש מימוש אפשרי ללא הארגומנט הזה), אך לא חייב. במידה והחלטתם לוותר על הפרמטר עליכם להגדיר מחדש את החתימה של המתודה בתוך הממשק IDrawable.
* בתוך מחלקת Vehicle, בתוך פונקציית move, עליכם לבדוק שהרכב אכן יכול לנסוע (יש דלק וכו'). אם כן, מפעילים על הרכב sleep ל100 מילישניות, מזיזים את הרכב לנקודה הבאה על ידי קריאה לפונקציה drive(לפי המהירות שלו כפי שתואר בקובץ העבודה), וכמו כן קוראת ל-pan.repaint() – כלומר "מציירת מחדש" את הפאנל שכלי הרכב מכיל על מנת שנוכל לראות את הרכב זז.
* הפונקציה repaint() של הפאנל בעצם מפעילה את המתודה paintComponent של הפאנל (באופן אוטומוטי- ואין צורך לממש את repaint כך שתעשה זאת!!). הפונקציה paintComponent(Graphics g) קוראת ל-drewObject של הרכב שנמצא בו במידה והוא קיים והיא מציירת אותו בהתאם לתמונה שתיבחר ב-drewObject.
* **מימוש (משופר) של drewObject ב-Vehicle:**

**public void** drawObject(Graphics g)  
{  
 **if**(**loc**.getOrien() == Location.Orientation.***NORTH***) *//drives to right side* g.drawImage(**img1**, **loc**.getP().getX(), **loc**.getP().getY(), **size**, **size**\*2, **pan**);  
 **else if** (**loc**.getOrien() == Location.Orientation.***SOUTH***)*//drives to the south side* g.drawImage(**img2**, **loc**.getP().getX(), **loc**.getP().getY(), **size**, **size**\*2, **pan**);  
 **else if**(**loc**.getOrien() == Location.Orientation.***EAST***){ *//drives to the east side* g.drawImage(**img3**, **loc**.getP().getX(), **loc**.getP().getY(), **size**\*2, **size**, **pan**);}  
 **else if**(**loc**.getOrien() == Location.Orientation.***WEST***) *//drives to the west side* g.drawImage(**img4**, **loc**.getP().getX(), **loc**.getP().getY(), **size**\*2, **size**, **pan**);  
}

* **מימוש של פונקציית move במחלקת Vehicle:**

**public void** move(Point p){  
 **if**(**canMove**){  
 **try** { Thread.*sleep*(100); }  
 **catch** (InterruptedException e) { e.printStackTrace(); }  
 **this**.drive(p);  
 }  
 **pan**.repaint();  
}

* **מימוש של פונקציית paintComponent בCityPanel:**

**protected void** paintComponent(Graphics g)  
{  
 **super**.paintComponent(g);  
 g.drawImage(**backgroundImage**,0, 0, getWidth(), getHeight(), **this**);  
 **if** (**v**!=**null**){ //if the vehicle object exists  
 **v**.drawObject(g);  
 **v**.move(**v**.nextLocation());}  
}

* הפונקצייה v.nextLocation() זו בעצם פונקציה שמגדירה מה היא הנקודה הבאה של שאליה צריך להגיע הרכב ומחזירה את הנקודה הזאת (שימו לב- בפונקציה הזו לא משנים שום דבר באובייקט אלא רק מחזירים את הנקודה הבאה שהוא מגיע אליה):
  + - אם יש צורך לשנות את כיוון הרכב כי הוא הגיע לצומת כלשהי במסך – החישוב מתבצע שם ובהתאם לפי הצורך משתנה הכיוון של כלי הרכב.
    - הפונקציה בודקת מה הוא כיוון הרכב נוכחי ולפיכך יודעת להגדיר לו מה היא הנקודה הבאה שלו על המסך:  
      אם צפונה – אז צריך לקדם אותו רק על ציר y – משנים את הy להיות הy הנוכחי **פלוס** המהירות של הרכב

אם דרומה – אז צריך לקדם אותו רק על ציר y – משנים את הy להיות הy הנוכחי **מינוס** המהירות של הרכב (כי הוא בעצם נוסע לכיוון השלילי של ציר y)

וכן הלאה.....

* מה שקורה זה בעצם שהפונקציה move של Vehicle והפונקציה paintComponent של CityPanel קוראות אחת לשניה ללא הפסקה ובכך מתאפשרת התקדמות תמידית של הרכב וציור מחדש של המסך בהתאם.
* **אין לולאת while(true)- זה לא רלוונטי לעבודה הנוכחית כיוון שלא מדובר ב-threads! המימוש של הנסיעה הרציפה והמתמשכת היא על ידי המימוש הנ"ל.**
* לגבי ICloneable- מי שרוצה יכול לממש ישירות את הממשק Clonable שקיים בג'אווה במקום. מי שנשאר עם ICloneable זה פשוט לעטוף את Clonable הקים על ידי שימוש בפונקצייה clone() שקיימת בג'אווה.
* שימו לב- כל מחלקה שממשת את ממשק Clonable (או IClonable) צריכה בסופו של דבר לממש דריסה של פונקציית clone שקוראת לsuper.clone() כך:

@Override  
**protected** Object clone() **throws** CloneNotSupportedException {  
 **return super**.clone();  
}

**בהצלחה!**