תכנות מונחה עצמים מתקדם
עבודת הגשה מס' 2

להגשה עד ה 02/05 ב-23:55

דגשים להגשה

- קראו היטב את הוראות העבודה מההתחלה ועד הסוף לפני שתתחילו – זה חשוב והכרחי לצורך הצלחתכם.
- ניתן להגיש עבודה זו בזוגות – רק אחד מהסטודנטים יגיש את העבודה במודל. בתיעוד של קובץ יש לציין שם, ות"ז בתוך תיעוד ה-javadoc
- לכל שאלה אנא עברו על הפורום באתר הקורס במודל על מנת לראות אם היא כבר נענתה.
  במידה ולא, ניתן להוסיף שאלה בפורום או לפנות למתרגלת האחראית – שירז נווה במייל shirazbarsce@gmail.com
  על כל פניה להכיל את פרטי הסטודנט המלאים כולל ת.ז ושם מלא.
- חובה לתעד כל קובץ, מחלקה ופונקציה ע"י javadoc
  ניתן להיעזר בתיעוד באתר oracle 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 שימוש ב-GUI.

מטרת התרגיל:

- שימוש בעקרונות **MVC** לבנית סביבת **GUI**.
- **MVC (Model – View – Controller)** – כפי שנלמד בקצרה בתרגול, זוהי תבנית עיצוב מבנית אשר מחלקת את היישום לשלושה חלקים לוגיים – "מודל", "תצוגה", "בקר". כל אחד מהרכיבים הללו נוצר על מנת לטפל בחלקי פיתוח ספציפיים של יישום. בקצרה:
  - **רכיב המודל** אחראי לכל הלוגיקה שקשורה בנתונים שאיתם המשתמש עובד.
  - **רכיב התצוגה** אחראי לכל הלוגיקה הקשורה בממשק המשתמש ביישום.
  - **רכיב הבקר** משמש כממשק בין רכיבי המודל והתצוגה – עיבוד הבקשות הנכנסות, תפעול הנתונים באמצעות רכיב המודל, ויצירת אינטראקציה עם רכיב התצוגה על מנת לעדכן את הפלט המוצג.

MVC היא אחת מתבניות הפיתוח הסטנדרטיות והנפוצות בתעשייה לצורך פיתוח פרויקטים (web בפרט) הניתנים להרחבה (scalable).

לקריאה נוספת:

https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

https://he.wikipedia.org/wiki/Model_View_Controller

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

אינטואיציה כללית לעבודה זו כהכנה לעבודה 3:

בעבודה זו תצטרכו להגדיר שדות מסוימים אשר לא ייעשה בהם שימוש בעבודה הנוכחית, אך אנו מגדירים אותם כבר מעכשיו לצורך העבודה הבאה. הרעיון הוא שלקראת עבודה הבאה תלמדו כבר על תהליכונים (Threads) ואז החיות יתחילו לנוע לבד בגן החיות. לכן אם יש משתנים שאין בהם שימוש כרגע – תדעו שזה בסדר. נא להימנע מלשאול על זה שאלות מיותרות.

שימו לב – בעבודה מצורפות תמונות להמחשה. חלק מהתמונות נוצרו על מערכת ההפעלה macOS. דעו שאם זה נראה שונה אצלכם זה בסדר.

2. הגדרות כלליות:

אתם מייבאים את כל הממשקים והמחלקות ישר מן התרגיל הקודם למעט מטלה 7 (ה- main).

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

- public class **ZooFrame** extends JFrame – המחלקה תכלול פונקציית ***main***
- public class **ZooPanel** extends JPanel implements Runnable
- public class **AddAnimalDialog** extends JDialog
- public interface **IDrawable**
- public interface **IAnimalBehavior**

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();
}
```

ממשק **IAnimalBehavior** צריך להגדיר בצורה הבאה:

```
public interface IAnimalBehavior {
    public String getAnimalName();
    public int getSize();
    public void eatInc();
    public int getEatCount();
    public boolean getChanges ();
    public void setChanges (boolean state);
}
```

בנוסף להגדרה של **Animal** מתרגיל הקודם:
בתרגיל הזה **Animal** מממש גם את הממשקים **IDrawable** ו-**IAnimalBehavior**

```
public abstract class Animal extends Mobile implements IEdible, IDrawable,
IAnimalBehavior.
```

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

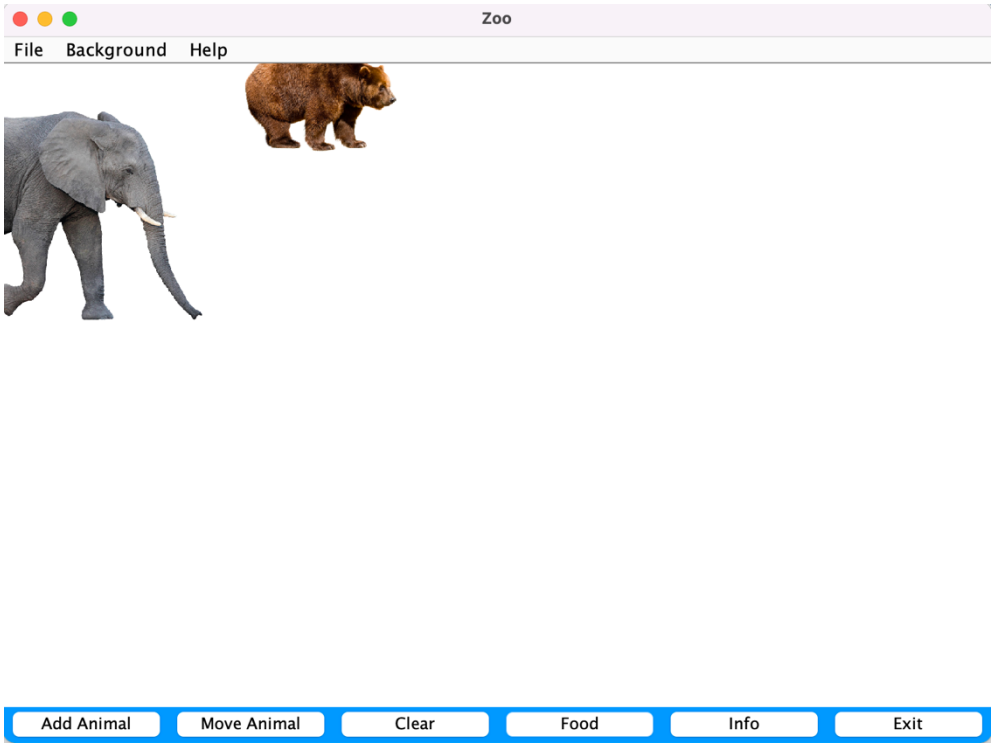
```
    private final int EAT_DISTANCE = 5;
    private int size;
    private Color col;
    private int horSpeed;
    private int verSpeed;
    private boolean coordChanged;
    private Thread thread;
    private int x_dir;
    private int y_dir;
    private int eatCount;
    private ZooPanel pan;
    private BufferedImage img1, img2;
```

במחלקת Animal ובכל המחלקות היורשות יש להגדיר בנאים נוספים לאתחול שדות וגם להוסיף פונקציות לפי צורך. **ניתן להרחיב גם מחלקות וממשקים אחרים.**

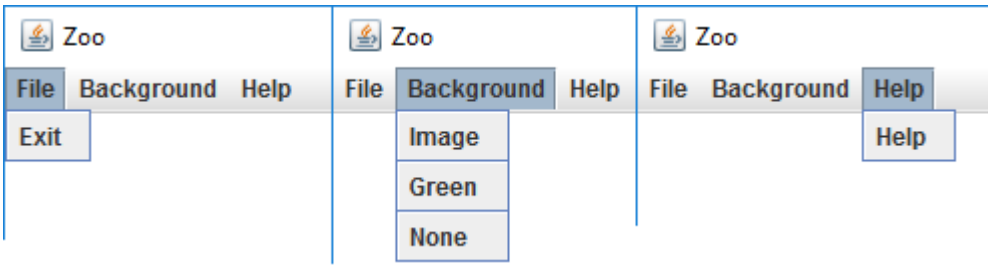
בנוסף להגדרה של **Plant** מתרגיל הקודם: בתרגיל הזה **Plant** מממש גם את **IDrawable .**

3. הגדרות ה- View עבור התרגיל :

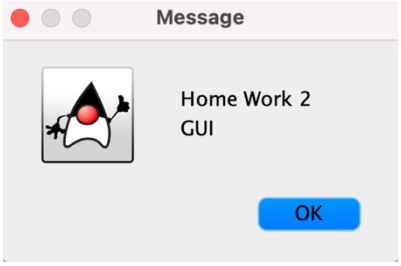
- ה- **ZooFrame** כולל menu ו **ZooPanel** עם מספר כפתורים הנמצאים בתחתית הפאנל.
- כל השטח ה-**ZooPanel** מיועד לציורי חיות:



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



- **Exit** – סגירה של התכנית
- **Image** – אפשרות להשתמש בתמונה לבניית רקע של גן חיות
- **Green** – בניית רקע עם צבע ירוק
- **None** – ללא רקע
- **Help** – לפתיחת *showMessageDialog* בצורה הבאה:



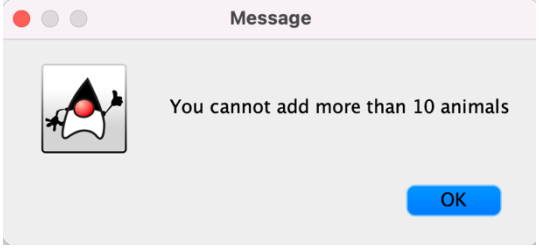
- **הכפתורים** שנמצאים בתחתית ה-**ZooPanel** מבצעים את הפעולות הבאות:
  - **Add Animal** – פותח חלון של **AddAnimalDialog** (מחלקה שאתם צריכים להגדיר, תחשבו ממי היא יורשת ומה היא צריכה לממש), שמאפשר להגדיר חיה חדשה עם פרמטרים הבאים:
    1. בחירת סוג של חיה: פיל, אריה, ג'ירפה, צב או דוב.
    2. גודל של חיה (בפיקסלים) – מ-50 עד 300
    3. מהירות אופקית (פיקסלים ביחידת זמן) – מ-1 עד 10 **(יעשה שימוש בעבודה 3)**
    4. מהירות אנכית (פיקסלים ביחידת זמן) – מ-1 עד 10 **(יעשה שימוש בעבודה 3)**

    הערה: יש לוודא שמספרים מיועדים למהירות וגודל רק שלמים. אחרת יש להודיע שנתון לא נכון באמצעות *showMessageDialog*.
    5. הצבע של החיה (יש לבחור אחת מכמה אופציות, למשל: אדום, כחול וטבעי).

משקל החיה יחושב לפי גודלה. יש להשתמש בנוסחאות הבאות:

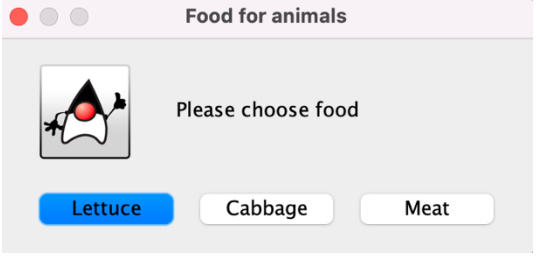
- בשביל פיל: משקל = גודל * 10
- בשביל ג'ירפה: משקל = גודל * 2.2
- בשביל דוב: משקל = גודל * 1.5
- בשביל אריה: משקל = גודל * 0.8
- בשביל צב: משקל = גודל * 0.5

גן החיות לא יכול להכיל יותר מ-10 חיות ברגע נתון. במידה ויש כבר 10 חיות והמשתמש מנסה להגדיר חיה נוספת יש להודיע שאין אפשרות כזו:

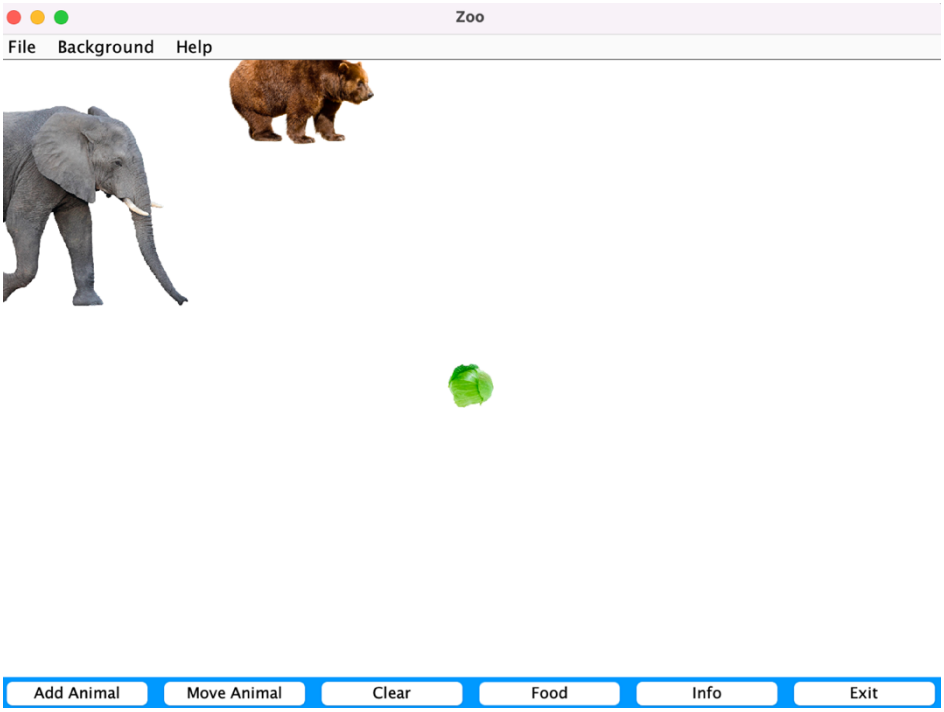


- **Move Animal** – פותח חלון של **MoveAnimalDialog** (מחלקה שאתם צריכים להגדיר, תחשבו ממי היא יורשת ומה היא צריכה לממש), שמאפשר לבחור מיקום חדש לחיה ספציפית (מתוך רשימה של כל החיות הקיימות) באמצעות הפרמטרים הבאים:
  1. בחירת חיה מתוך רשימת החיות הקיימות בגן החיות ברגע הנתון
  2. קורדינאטת x חדשה (מ-0 עד 800)
  3. קורדינאטת y חדשה (מ-0 עד 600)

  כאן כאשר מציגים את רשימת החיות הקיימות יש לחשוב כיצד ניתן להשתמש בייצוג של אובייקט באופן שיהיה מובן על ידי האדם.
- **Clear All** – מחיקת כל החיות מהפאנל.
- **Food** – חלוקת אוכל. יש להוסיף אפשרות לבחור סוג של אוכל: "Meat" (שמתאים לבעלי חיים טורפים) או "Cabbage" או "Lettuce" (שמתאים לבעלי חיים אוכלי עשב)



אוכל מופיע במרכז ה-**ZooPanel** (יש לקרוא לפונקציה ***repaint***). החיה הראשונה שמגיעה (ע"י הזזתה) לאוכל למרחק פחות מ-10 פיקסלים גם ב-X וגם ב-Y , אוכל אותו ומעדכנת את "***Eat counter***" שלה.



- **Info** – מופיע כל המידע על החיות בצורת ה-JTable (בחלון נפרד או ב-ZooPanel):

Animal	Color	Weight	Hor. speed	Ver. speed	Eat counter
Elephant	Natural	2750	5	5	4
Lion	Natural	185	7	5	3
Giraffe	Natural	650	4	4	3
Turtle	Natural	50	4	1	3
Bear	Natural	280	8	7	4
Total					17

- **Exit** – סגירה של התכנית.

הגדרות ה-Controller עבור התרגיל:

Controller – יש להגדיר אותו ב-**ZooPanel** . יש להגדיר בתוך **ZooPanel** פונקציה בשם **manageZoo** והיא תהווה את ה- controller שלנו. יש לקרוא לפונקציה manageZoo לאחר סיום הקמת וחשיפת ה-**ZooFrame**. ה-controller רץ תמיד ומבצע את שני הדברים הבאים:

- חיפוש שינוים אפשריים של מיקום החיות. אם יש איזשהו שינוי יש לקרוא לפונקציית

ZooPanel.paintComponent(Graphics g) (**repaint()**). כלומר צריך לממש פונקציה בשם

paintComponent(Graphics g) (שתפקידה לצייר מחדש את כל האובייקטים) בתוך ZooPanel כאשר

הקריאה אליה נעשית באמצעות repaint(). (ראו דוגמה לחתימה של paintComponent בהמשך)

למעשה ה- controller הוא זה שמנהל את הציור של החיות ומיקומן על המסך.

- מכיוון שמחלקה **Animal** מממשת ממשק **IEdible**, אז קיימת האפשרות שחיה אחת יכולה לאכול חיה אחרת. ה- **Controller** מבצע בדיקות של אפשרויות כאלה. חיה אחת יכולה לטרוף חיה אחרת רק אם מתקיימים שלושת התנאים הבאים:
  - ***פונקציה canEat() של שדה diet של טורף מחזירה "true"***
  - ***משקלו של הטורף הוא לפחות פי שניים יותר גדול ממשקלו של הטרף***
  - ***המרחק בין טורף לבין טרף קטן יותר מגודלו של טרף***

  במקרה שאוכלים חיה כלשהי יש למחוק אותה מאוסף החיות.

הבהרה: ה- controller מנהל את הדברים הנ"ל (שימו לב – לא מנהלים את זה דרך החיות עצמן!). לצורך העניין – אם חיה שינתה את מיקומה, אזי ערכי הקורדינאטות שלה ישונו וכמו כן גם ערך ה- coordChanged שלה. הקונטרולר הוא זה שצריך לבדוק כל הזמן אם יש שינוי אצל חיה כלשהי. אם כן – הוא יצייר מחדש את החיה על המסך בהתאם.

תבנית הפונקציה manageZoo:

```
public void manageZoo()
{
  while (true) {
        if (isChange())
                repaint();
        .
        . // need to check if some animal can eat another animal according to their locations
        .
}
```

כאשר הפונקציה isChange() היא פונקציה שבודקת עבור כל חיה האם קיים אצלה שינוי בהתאם לפרמטר coordChanged. הסבר נוסף בהמשך.

4. הגדרות ה-Model עבור התרגיל:

- כל חיה בתרגיל מופיעה על המסך עם סיום יצירתה בנקודה **הדיפולטית** שהוגדרה עבורה בעבודה 1.
- את כל החיות יש להכניס ל-**ArrayList<Animal>** , שיש להגדיר ב-**ZooPanel.**
- אחרי כל שינוי של מיקום חיה יש לעדכן את השדה בוליאני **coordChanged**.
- יש להגדיר ב-**ZooPanel** גם את שדה מסוג **Plant** (ערך האתחול שלו הוא null). במקרה שבוחרים את אוכל "Cabbage" או "Lettuce" יש ליצור עצם חדש **Cabbage** או **Lettuce** . אחרי שהאוכל נאכל ערך של שדה הזה שוב חוזר ל-null.
- לכל החיות והצמחים יש לממש פונקציית **drawObject()** (המוכרזת בממשק **IDrawable**) ולקרוא לה ב-**ZooPanel.paintComponent()**.

בשביל ציורי בעלי חיים יש להשתמש ב- **BufferedImage**. דוגמא של שימושו:

1) **הגדרת Graphics g: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*" .

3) **ציור של תמונה כרקע**

```
public void paintComponent(Graphics g)   {
  super.paintComponent(g) ;

  if(img!=null)
        g.drawImage(img,0,0,getWidth(),getHeight(), this);
  .....
}
```

4) **לכל חיה יש שני שדות מסוג BufferedImage**: **img1** לצורך תנועה לצד ימין ו-**img2** לצורך תנועה לצד שמאל.

שימו לב!!!!

בעבודה זו עליכם להגדיר את הפונקציה כך אך באופן דיפולטי x_dir וגם y_dir יהיה תמיד שווה ל-1.

כלומר בפועל יעשה שימוש רק באחד מהכיוונים. בעבודה 3 יעשה שימוש בשני הכיוונים.

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

```
public void drawObject (Graphics g)
{
  g.setColor(col);
  if(x_dir==1) // giraffe goes to the right side
        g.drawImage(img1, location.x-size/2, location.y-size/10, size/2, size, pan);
  else // giraffe goes to the left side
        g.drawImage(img2, location.x, location.y-size/10, size/2, size, pan);
}
```

בדוגמא הזו משתמשים במשתנים הבאים כאילו שהם protected, עליכם להשתמש בgetters ו-setters במקום:

- **col** – צבע של חייה (private data member של מחלקת **Animal**)
- **size** – גודל של חייה (private data member של מחלקת **Animal**)
- **location.x** ו-**location.y** - קואורדינטות של "פה" החיה (**private Point location** הוא שדה של מחלקת **Animal**), הקואורדינטות האלה משתנים ע"י ביצוע פעולה של move animal.
- **x_dir**– מגדיר כיווני תנועה של חיות (**x_dir=1** – תנועה ימינה, **x_dir=-1** – תנועה שמאלה). **בעבודה זו לא משתנה במהלך התכנית.**

- החייה הראשונה שמגיעה לאוכל (ע"י הזזתה) למרחק פחות מ-10 פיקסלים אוכלת אותו וקוראת ל-**callback** של **ZooPanel**, שמעדכן את "***Eat counter***" לחייה ו-"***Total Eat counter***".

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

עבודה נעימה