תכנות מונחה עצמים מתקדם

עבודת הגשה מס' 3

להגשה עד ה 23/05/2022 ב-23:55

דגשים להגשה

- 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 – ג'ודי מנדלבוים במייל judymand70@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 2 – עליכם לעדכן/להרחיב את המחלקות הקיימות במידת הצורך ולהשתמש בהן. כל הדרישות מעבודה 2 רלוונטיות לעבודה זו, אלא אם נכתב אחרת.**
- **במידה ולא נאמר בתרגיל כיצד לבצע דבר מסוים, כל פתרון נכון ויעיל שאינו פוגע בעקרונות התכנות הנכון יתקבל.**

1. מבוא

זהו התרגיל השלישי בקורס ,בתרגיל זה נתרגל תכנות Threads עם שימוש ב-GUI.

מטרת התרגיל:

- בנייה והרצה **Threads** ב-Java.
- פיתוח מנגנון **סנכרון** בין Threads.

יש לבצע את העבודה לפי הדרישות המצוינות במסמך הנתון. בעבודה זאת יש נשתמש בטריידים בכדי לבצע פעולות שונות בגן החיות.

2. הגדרות כלליות:

אתם מייבאים את כל הממשקים והמחלקות ישר מן התרגיל הקודם.

בממשק **IAnimalBehavior** צריך להוסיף את המתודות הבאות:

public void setSuspended(); - החיה נכנסת למצב של המתנה.

public void setResumed(); - החיה יוצאת ממצב המתנה.

בתרגיל זה כל חיה הופכת לתהליכון.

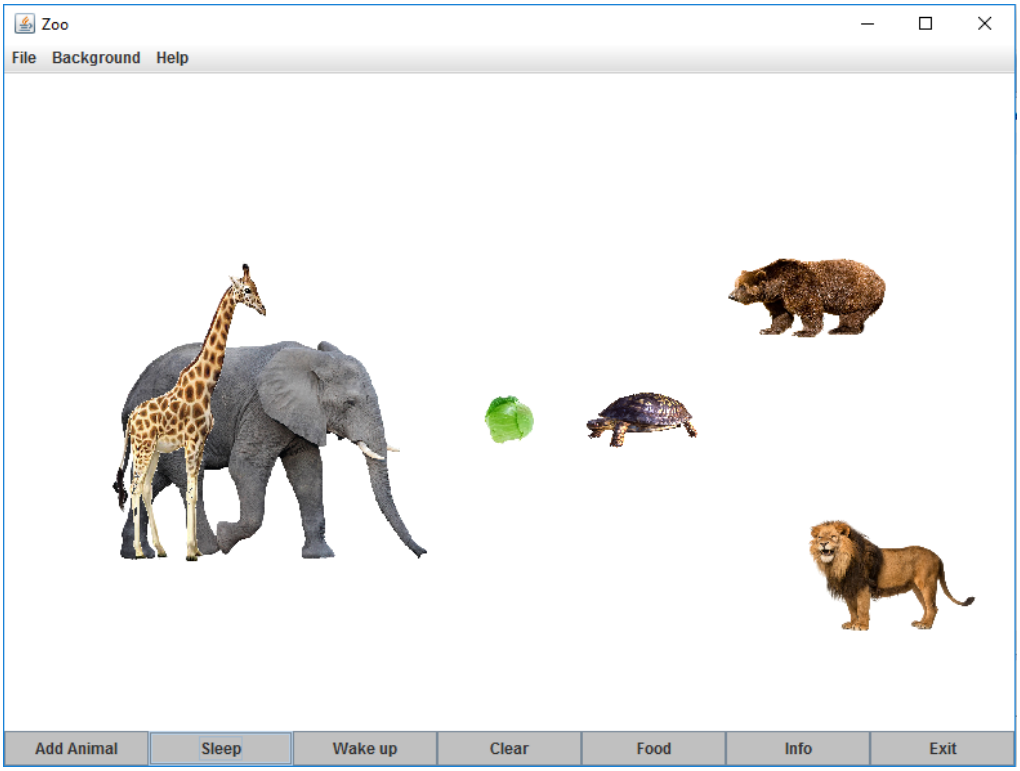
בנוסף להגדרה של **Animal** מתרגיל הקודם: בתרגיל הזה **Animal** מממש גם את ממשק **Runnable:**
במחלקה **Animal** יש להגדיר את השדות נוספים:

```
protected Thread thread;
protected boolean threadSuspended;
```

3. **הכפתורים** שנמצאים בתחתית ה-**ZooPanel** והתנהלות עם אוכל:

תוספות לפונקציונליות של הכפתורים שיצרתם:

לאחר יצירת חיה ע"י **Add Animal** שיצרתם בעבודה הקודמת, החייה החדשה מתחילה לנוע עם המהירות שהגדרנו בעת יצירתה.
את האופציה של Move Animal יש למחוק.
Sleep - עצירה זמנית של כל החיות (כל החיות נשארות במקומן ללא תזוזה).
Wake up – חזרה לתנועה של כל החיות.
Clear All – מחיקת כל החיות מהפאנל והריגת כל התהליכונים של החיות.
Food - אוכל מופיע במרכז ה-**ZooPanel** (יש לקרוא לפונקציה ***repaint***). אם סוג של אוכל מתאים לבעל חיים, הוא משנה את כיוון התנועה שלו לכיוון מרכז ה-**ZooPanel** כמטרתם הוא תפיסת האוכל. אם סוג האוכל לא מתאים לבעל חיים, הוא לא יגיב לאוכל וימשיך לנוע ללא שינוי.
Info – כמו בעבודה קודמת.
Exit –הריגת כל התהליכונים וסגירה של התוכנית.



4. הגדרות ה-Controller עבור התרגיל:

שימו לב, ה- ZooPanel עצמו הופך לתהליכון בעבודה זו. בנוסף להגדרות הקודמות של מחלקת ZooPanel, מחלקה זו תממש את ממשק Runnable.
נוסיף שדה למחלקה זו:

private Thread controller;

בדומה לעבודה 2, ה- **ZooPanel** צריך "לנהל" את גן החיות ולהיות במעקב אחרי כל שינוי אפשרי.

- חיפוש שינוים אפשריים של מיקום החיות. אם יש איזשהו שינוי יש לקרוא לפונקציית **ZooPanel.paintComponent()** (**repaint()**).
- מכיוון שמחלקה **Animal** מממשת ממשק **IEdible** , אז קיימת האפשרות שחיה אחת יכולה לאכול חיה אחרת. **התהליכון 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 חיות **ולהרוג את תהליכון שלה.**

את הפונקציה manageZoo ניתן למחוק ולהעביר את האחריות שלה לפונקציית run() או לחילופין לעדכן את manageZoo בהתאם לדרישות העבודה, ולקרוא לה מה-run. נתון להחלטתכם.

5. הגדרות ה-Model עבור התרגיל:

- כל חיה בתרגיל היא ה- **Thread** שמתחיל לרוץ כשהחיה נוצרת, החל מהמיקום הדיפולטי שלה בתנועה ימינה.
- החיה נעה בגן החיות על פי המהירות שלה בכל אחד מהצירים. ברגע שחיה מגיעה לאחד מגבולות המסך, היא צריכה לשנות את כיוונה בהתאם. אם החיה הגיעה לגבול ימני או שמאלי, יש להפוך את כיוונה ואת כיוון הפנים שלה (מבחינת תמונה). אם החיה הגיעה לגבול תחתון או עליון, יש לשנות את כיוונה בלבד מבלי לשנות את התמונה.
- כל עוד החיה לא ישנה או מושהית, היא **נמצאת בתנועה כל הזמן**, ללא דרישה או בקשה מהמשתמש.
- במידה ושני חיות אשר לא אוכלות אחת את השנייה מתנגשות, הן ימשיכו לנוע.

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 שיעשה שימוש בשני הכיוונים.

- **location.x** ו-**location.y** - קואורדינטות של "פה" החיה (**protected Point location** הוא שדה של מחלקת **Animal**), הקואורדינטות האלה משתנים בפונקציית **run()** בהתאם למהירות של חיה על פי הנוסחה הבאה:
  - new_x = old_x + hor_speed * x_dir

- new_y = old_y + hor_speed * y_dir

- **x_dir**– מגדיר כיווני תנועה של חיות (**x_dir=1** – תנועה ימינה, **x_dir=-1** – תנועה שמאלה), ובצורה דומה עם **y_dir** (למעלה ולמטה). כיוון החיה על ציר x, הוא זה שיקבע באיזו תמונה של החיה להשתמש (עם הפנים שמאלה או ימינה). גם משתנה בפונקציית **()run.**

- כדי לעצור ולהעיר את החיות יש להשתמש בפונקציות **()wait** ו-**()notify**.
- כל פניות למשתנים משותפים בין תהליכונים צריכות להיות **מסונכרנות**, לדוגמא בכל פנייה למשתנה המשותף threadSuspended שנמצא במחלקת animal, כאשר נרצה לשנות את הערך של שדה זה, נרצה להגביל את הגישה לשדה זה, מכיוון שהוא שדה משותף של כל החיות, וכל חיה היא למעשה thread.

חשבו היטב היכן יש צורך בסנכרון ומה צריך להיות האובייקט שעליו יש לבצע את הנעילה.

שימו לב! צריך לבנות את העבודה בצורת thread safe כפי שלמדתם בהרצאות ובתרגולים.

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

עבודה נעימה