בתרגיל הזה תפתחו סימולציה של עולם המכיל חיות ממינים שונים וצמחייה. החיות השונות נמצאות באינטראקציה בינן ובין עצמן ובינן ובין העולם לפי חוקי העולם שיתוארו בהמשך. התרגיל מתאר את חוקי הסימולציה, מנשקים מסוימים שאתם חייבים להשתמש בהם (כדי שתוכלו לשתף קוד שמגדיר התנהגות של מיני חיות), ומספק רמזים לגבי מבני נתונים אפשריים, אבל את רוב עבודת התכנון והתכנות תעשו בעצמכם.

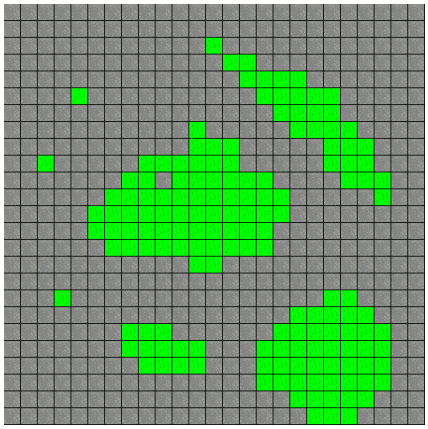
כחלק מהתרגיל תצטרכו לקבל החלטות מימוש. עליכם לשקול את האופציות העומדות בפניכם ולבחור בזו הנראית לכם טובה ביותר. בכל מקום בו החלטת המימוש שקבלתם אינה טריוויאלית (שתי אופציות שקולות, אלגוריתם מורכב וכו') דאגו לתעד את השיקולים שהובילו להחלטה.

קראו את כל התרגיל וכן את הקבצים המצורפים (קוד והערות) לפני שאתם מתחילים במלאכת המימוש. חישבו כיצד המערכת צריכה לפעול בהינתן הדרישות הפונקציונאליות ומגבלות הקוד. הריצו כמה תסריטים על "יבש" (הוספת חיה לעולם, עדכון העולם כתוצאה מפעולה של חיה, ...) כדי להבין טוב יותר את פעולת המערכת לפני כתיבת הקוד. תכננו את הקוד שלכם כך שייתן מענה לכל התרחישים המתוארים.

מהן המחלקות שבחרתם לממש? מהם היחסים בין המחלקות? מה מייצגת כל מחלקה ומה תחם האחריות שלה? מהי האחריות של כל שירות במחלקות אלו? האם אתם צריכים עוד מחלקות עזר? התייחסו לשאלות הללו בזמן התכנון של הפתרון. השקיעו זמן ומחשבה במתן תשובות לשאלות אלו, זמן זה יחסוך לכם שינויים בקוד מאוחר יותר.

העולם

העולם הוא לוח משחק ריבועי בגודל n על n. כל משבצת בלוח יכולה להכיל צמחייה (צבע ירוק באיור) וחיה אחת בלבד לכל היותר.



העולם הוא מעגלי. את המשבצות שלו נסמן בקואורדינאטות 0,0 (שמאלית תחתונה) עד n-1,n-1 (ימנית עליונה). מימין למשבצת i,n-1 נמצאת שוב משבצת i,0 ומעל משבצת n-1,j נמצאת משבצת 0,j.

כל משבצת בלוח יכולה להכיל צמחייה וחיה אחת לכל היותר (למעט אירוע מיוחד שיתואר בהמשך)

חיות

חיה היא יצור בעולם שיכול לנוע, לאכול ולהשריץ.

לכל חיה יש כמות אנרגיה מסוימת. פעולות שהיא עושה צורכות אנרגיה, ואכילה מספקת לה אנרגיה. כאשר האנרגיה של חיה נגמרת, החיה מתה.

כל חיה שייכת למין מסוים. המינים מתחלקים לשני סוגים: אוכלי עשב וטורפים.

המין קובע שלושה דברים לגבי החיה: האם היא אוכלת עשב או טורפת, מה גודל הצאצאים שלה (כמות האנרגיה שלהם בלידה), וטווח הראיה שלה. חיה עם טווח ראיה r רואה ריבוע של העולם בגודל 2r+1 על 2r+1 ($r << n$), כשהיא במרכזו. לגבי כל משבצת בטווח הראיה, החיה רואה האם יש בה צמחיה, האם יש בה חיה ואם כן מאיזה מין.

הסימולציה

הסימולציה מורכבת ממהלכים. בכל אחד מהמהלכים כל חיה יכולה לפעול פעם אחת בלבד.

סדר הפעולה של החיות הוא כסדר הוספתן לעולם למעט צאצאים. המיקום של הצאצא בסדר הפעולות הוא מייד לאחר ההורה. כלומר לאחר שההורה משריץ (ובדרך כלל נע), הצאצא פועל מייד אחריו.

חיה יכולה לנוע משבצת אחת בכל תור. כלומר, בכל תור היא יכולה לנוע לאחת משמונה המשבצות המקיפות אותה (זכרו העולם מעגלי) תחת המגבלה שכל משבצת בעולם מכילה חיה אחת לכל היותר. מכאן נובע שחיה אוכלת עשב יכולה לנוע רק למשבצת שאינה מכילה חיה אחרת (אוכלת עשב או טורפת) וחיה טורפת יכולה לנוע למשבצת שיש בה חיה אוכלת עשב (אבל לא חיה טורפת).

האכילה בעולם מבוצעת באופן אוטומטי, כלומר לאחר שחיה נעה (או בוחרת להישאר במקום) אם המשבצת בה היא נמצאת מכילה מזון המתאים למין החיה אזי המזון יאכל. במקרה של חיה אוכלת עשב, הצמחייה הנמצאת באותה המשבצת תעלם ואילו במקרה של חיה טורפת החיה שקודם לכן הייתה במשבצת תאכל ותחדל להתקיים.

אם חיה מחליטה להשריץ ויש לה מספיק אנרגיה לשם כך (אם לא – היא מתה), נוצרת חיה חדשה מאותו מין באותה המשבצת. אם החיה המקורית לא נעה בתור זה היא מתה (מכיוון שהמשבצת יכולה להכיל חיה אחת לכל היותר). הצאצא מתחיל את חייו עם אנרגיה מינימלית לאותו המין וכמות אנרגיה זו מופחתת מההורה.

כאשר מגיע תורה של חיה לנוע, העולם בודק קודם כל האם היא עדיין בחיים. עצם ההישרדות של חיה עד לתורה עולה לה 0.03 יחידות אנרגיה. אם יש לה פחות, היא מתה מייד כאשר מגיע תורה. אם יש לה יותר מכך, הסימולציה קוראת לשירות act של החיה.

כאשר מגיע תורה של חיה לפעול, מופעל השירות ()act של החיה. שירות זה צריך להחזיר עצם שמתאר את ההחלטה של החיה כיצד לפעול. ההחלטה כוללת שני פרמטרים, האם להשריץ (spawn, להתפצל לשתי חיות, אחת בגודל של צאצא של המין והשנייה בגודל שנותר לאחר הקצאת האנרגיה לצאצא) והאם לנוע למשבצת סמוכה (ולאיזו מהן).

הסימולציה מוציאה לפועל את ההחלטה של החיה (כפי שהתקבלה בעצם המוחזר) היא מוודאת כמובן שהחיה לא חורגת מתנאי הסימולציה (לדוגמא, תזוזה למשבצת חוקית עבורה). במקרה שחיה מקבלת החלטה שאינה חוקית, החיה מתה.

ראשית, במידה והחיה בחרה להשריץ ויש לה מספיק אנרגיה לשם כך תיווצר חיה חדשה מאותו מין ובעלת אנרגיה התחלתית מתאימה באותה המשבצת של ההורה. האנרגיה הראשונית של הוולד תופחת ממאזן האנרגיה של ההורה. רק כעת תתבצע התנועה של החיה (באם בחרה לנוע) ועלות התנועה (0.01 יחידות אנרגיה) תופחת ממאזן האנרגיה שלה. זה המצב היחיד בסימולציה בו (רגעית) נמצאת יותר מחיה אחת במשבצת.

חיה מגדילה את מאזן האנרגיה שלה ע"י אכילה. כאשר חיה אוכלת עשב נעה למשבצת שיש בה צמחייה היא אוכלת את הצמחייה באותה המשבצת (באופן אוטומטי) ומוסיפה למאזן האנרגיה שלה 0.25 יחידות אנרגיה.

כאשר חיה טורפת נעה למשבצת שיש בה חיה אוכלת עשב החיה הראשונה טורפת את השניה (באופן אוטומטי) ומוסיפה למאזן האנרגיה שלה את מספר יחידות האנרגיה של החיה שנטרפה. החיה הנטרפת כמובן מתה.

אם חיה אוכלת עשב אכלה את הצמחייה במשבצת מסוימת, המשבצת תישאר ללא עשב לפחות עד לתחילת הסיבוב הבא (לאחר שכל החיות יסיימו את פעולתן).

בתחילת סיבוב, לפני שהחיה הראשונה בסבב פעלה, עשב צומח מחדש בהסתברות של 10% בכל משבצת קרחת. (יש לבצע את החישוב עבור כל משבצת קרחת בנפרד).

החיות הראשונות (בתחילת המשחק) נוצרות עם כמות אנרגיה זהה לצאצא מהמין המתאים.

הסימולציה מסתיימת לאחר 1000 סיבובים או כאשר כל החיות נכחדו.

מחלקות ומנשקים

נתונים מספר מנשקים ומחלקות שעליכם לממש. תוכלו להוריד את הקבצים מאתר הקורס.

- המנשק Tile

המנשק מתאר משבצת אחת בעולם. המנשק מאפשר קריאה בלבד(read only) של מצב המשבצת.

```
package il.ac.tau.cs.sw1.wildworld;


/**
 * A single tile in the world. A tile may be occupied by a single
 * animal at a time, and may contain vegetation
 */
public interface Tile {

	/**
	 * Get the animal occupying this tile (if exists)
	 *
	 * @return The animal at this tile, or null if none exists.
	 */
	public Animal getAnimal();

	/**
	 * Check if there is vegetation in this tile.
	 */
	public boolean hasVegetation();

}
```

המנשק Animal

המנשק Animal מתאר סוג של חיה. כל אובייקט ממחלקה המממשת מנשק זה הוא חיה בעולם.

חיה פועלת בעזרת השירות ()act. למעשה בשירות זה החיה מחליטה מה ברצונה לעשות והערך המוחזר של השרות מייצג החלטה זו (ראו למטה). הסימולציה היא שמוציאה לפועל את ההחלטה.

תיעוד מפורט יותר (כולל חוזה) עבור המתודות השונות תוכלו למצוא בקובץ הנמצא באתר.

```
package il.ac.tau.cs.sw1.wildworld;

/**
 * Represents an animal in the system
 */
public interface Animal {
    public Action act();

    public void decreaseEnergy(double amount);

    public double getEnergy();

    public String getName();

    public int getRangeOfVision();

    public double getSizeOfOffspring();

    public double getVisionEnergy();

    public World getWorld();

    public void increaseEnergy(double amount);

    public boolean isCarnivore();

    public boolean isHerbivore();

    public Animal spawn();

}
```

המנשק Action

המנשק Action מתאר את הפעולה אותה מבקשת החיה להוציא לפועל, זוהי התוצאה של שירות ה-act. החיה יכולה לציין באם ברצונה להשריץ וכן האם ולהיכן ברצונה לנוע. חיה יכולה לנוע משבצת אחת בלבד בכל תור.

תיעוד מפורט יותר עבור המתודות השונות תוכלו למצוא בקובץ הנמצא באתר.

```
package il.ac.tau.cs.sw1.wildworld;

/**
 * The Action interface represents an animal's action
 */
public interface Action {

    /**
     * Constants representing the movement options for an animal.
     */
    static final int STAY_PUT = 0;
    static final int NORTH = 0x1;
    static final int SOUTH = 0x2;
    static final int EAST = 0x4;
    static final int WEST = 0x8;
    static final int NORTH_EAST = NORTH | EAST;
    static final int NORTH_WEST = NORTH | WEST;
    static final int SOUTH_EAST = SOUTH | EAST;
    static final int SOUTH_WEST = SOUTH | WEST;

    public Animal getAnimal();

    public int getMovement();

    public void setMovement(int movement);

    public boolean getSpawn();

    public void setSpawn(boolean spawn);
}
```

המחלקה World

המחלקה World אחראית על הרצת הסימולציה ועל מצב העולם ושינויו במהלך ביצוע הסימולציה. במחלקה זו תממשו את השירות simulate האחראי על ביצוע הסימולציה. כדי לממש שירות זה תזדקקו להחזיק מידע על מצב העולם והחיות בו. בחרו את מבני הנתונים שלכם בקפידה.

המחלקה Location ותיעוד מפורט יותר עבור המתודות השונות תוכלו למצוא בקובץ הנמצא באתר.

```
package il.ac.tau.cs.sw1.wildworld;

import il.ac.tau.cs.sw1.wildworld.util.Location;

public class World {

    public World(int size) {

    }

    public void addAnimalAt(Animal animal, Location location) {

    }

    public int getSize() {

    }

    public Tile getTileAt(Location location) {

    }

    public Tile getTileAt(Animal animal, int dx, int dy) {

    }

    public void simulate() {

    }

}
```

המטלה

1. ממשו את המחלקה World, וכן מחלקות המתארות שני סוגים של חיות. מחלקה אחת שמייצגת אוכל עשב (למשל Cow או Zebra וכדומה) ומחלקה שמייצגת טורף (Tiger או Lion וכדומה). בעת המימוש הקפידו לא לשכפל קוד ע"י שימוש מושכל בהורשה ובמחלקות אבסטרקטיות.
2. השירות simulate של World מבצע סימולציה של העולם. בתחילת כל סיבוב בסימולציה השירות ידפיס הודעה,וכן הודעה לאחר שכל חיה פועלת, למשל:

```
***  Round 93 Starting  ***
A cow at [24,24] spawned, moved to [25,24], and ate there
A cow at [24,24] moved to [23,24] and ate there
A lion at [30,31 moved to [30,32]
A lion at [90,99 moved to [90,100] and ate a cow
```

3. המטרה של החיות והמינים שלהם היא שהמין עצמו ישרוד זמן ארוך ככל האפשר ועם מספר פרטים גדול ככל האפשר. פתחו אסטרטגיות הישרדות חכמות. אפשר להשתמש באסטרטגיות עם מרכיבים רנדומאליים (ראו java.util.Random, ו Math.Random)
4. ממשו מחלקה בשם WordApplication ששירות ה-main שלה יוצר עולם בגודל 100 על 100 עם שלושים אוכלי עשב (מהמין שיצרתם) במשבצות כלשהן, ועם שני טורפים ב-24,74 ו-74,24, ומסמלצת את העולם. מה קורה (מי שורד הכי הרבה זמן, או לאיזה מין יש יותר חיות בסוף הסימולציה?)
שימו לב שלא צריך ליצור ממשק גראפי, רק להדפיס את הפעולות באמצעות פונקציות כדוגמת System.out.println
5. [לא חובה] נסו להחליף את החיות שאתם מימשתם בחיות שפיתחו סטודנטים אחרים, איזו חיה שורדת יותר טוב?

<u>הערה</u>
בכל הקשור למחלקות ולמנשקים שפורטו בתרגיל חל **איסור מוחלט!** על שינוי המנשקים ושינוי הממשק הפומבי של המחלקות. אין להוסיף או לגרוע פונקציות, לשנות את שמותיהן, לשנות פרמטרים וכדומה. מותר לכם במחלקות אלו להוסיף מימוש (שדות) ושירותי עזר פרטיים בהתאם למימוש שבחרתם.

הדרכה

אחד הקשיים בתוכנית הזו הוא למצוא את מבני הנתונים מתאימים עבור המחלקה World. עשו שימוש נרחב כל האפשר במחלקות קיימות (Map, List, Set וכדומה). קראו בעיון את התיעוד של

המנשקים והמחלקות המממשות אותם ובחרו במימוש המתאים ביותר לכם. הקפידו לשים לב לשימוש באיטרטור בכל אחת מהמחלקות ובפרט מה קורה לאיטרטור קיים כאשר משנים את מבנה הנתונים שהוא פועל עליו(!).

הגדירו תחילה מהם הנתונים להם אתם זקוקים כדי לממש את הסימולציה (מצב העולם, מיקום החיות בו, סדר הפעולה של חיות וכו'). בדקו אילו נתונים זמינים לכם במחלקות והמנשקים שהוגדרו ולאילו תדרשו להגדיר מבני נתונים בעצמכם.

מימוש איטרטיבי: מומלץ לממש את העולם תחילה ורק אחר כך, כאשר כבר ברור לכם לאילו נתונים אתם זקוקים לצורך הרצת הסימולציה, את שאר המחלקות. בנוסף מומלץ להתחיל במימוש אלגוריתם פשוט עבור החיות ורק אחרי שוידאתם שהמערכת עובדת כראוי ליצור אלגוריתמים מסובכים יותר.
אלגוריתם פשוט במיוחד הוא זוז משבצת אחת קדימה. אלגוריתם מעט מסובך יותר יהיה בדוק אם קיים מזון באחת המשבצות הקרובות, אם כן נוע למשבצת זו, אם לא זוז בצורה רנדומאלית.

דוגמא אפשרית ל main:

```
public static void main(String[] args) {
    // create a new world
    World world = new World(WORLD_SIZE);

    // add the lions to the world
    world.addAnimalAt(new Lion(world), new Location(24, 74));
    world.addAnimalAt(new Lion(world), new Location(74, 24));

    int i = 0;
    while (i < NUMBER_OF_COWS) {
        Location location = getRandomLocation();

        // add cow to the world; ensure location isn't occupied
        if (world.getTileAt(location).getAnimal() == null) {
            world.addAnimalAt(new Cow(world), location);
            i++;
        }
    }

    // run the simulation
    world.simulate();
}
```

הוראות הגשה

- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה ע"י המערכת VirtualTAU (http://virtua2002l.tau.ac.il).
- הגשת התרגיל תתבצע ע"י יצירת קובץ zip שנושא את שם המשתמש. לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip.
  קובץ ה zip יכיל:
  - קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
  - קבצי ה-java. של התוכנית (אין להגיש קבצי class.)
  - קובץ טקסט עם העתק של כל קבצי ה Java.