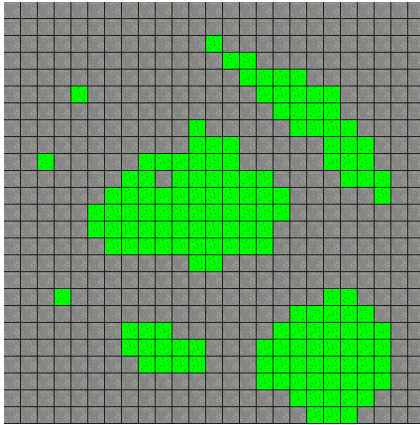


תוכנה 1 תרגיל 8 עולם פראי!

בתרגיל הזה נפתח סימולציה של עולם קטן עם חיות ממינים שונים ועם צמחיה. אתם צריכים לכתוב את תוכנית הסימולציה בעצמכם. 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.

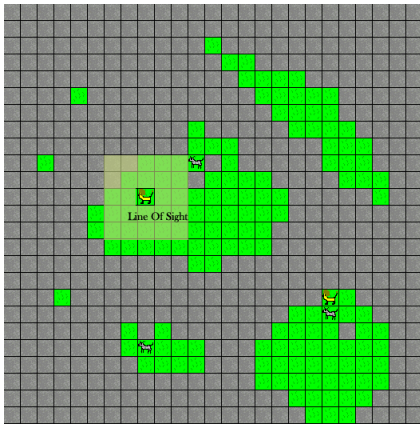
חוקי הסימולציה הם אלה:



- העולם הוא עצם מהמחלקה World שמייצגת לוח בגודל n על n של משבצות. בכל רגע נתון, במשבצת יש צמחייה (צבע ירוק באיור) או שהצמחייה נאכלה (אפור).
- העולם הוא ציקלי (מעגלי). את המשבצות שלו נסמן בקואורדינטות $(0,0)$ (שמאלית תחתונה) עד $(n-1, n-1)$ (ימנית עליונה). מימין למשבצת $(i, n-1)$ נמצאת שוב משבצת $(i, 0)$ ומעל משבצת $(j, n-1)$ נמצאת משבצת $(j, 0)$. משמאל למשבצת $(i, 0)$ נמצאת שוב משבצת $(i, n-1)$ ומתחת למשבצת $(j, 0)$ נמצאת משבצת $(j, n-1)$.

- הסימולציה מורכבת ממהלכים, שבכל אחד מהם כל חיה יכולה לפעול פעם אחת, לפי סדר שיקבע בהמשך. אם חיה אכלה את הצמחייה במשבצת מסוימת, המשבצת תישאר ללא עשב לפחות עד לסוף המהלך. בסוף כל מהלך, עשב צומח באופן אקראי בכל המשבצות הקרחות. ההסתברות שעשב יצמח במשבצת קרחת היא 20%.

- בכל משבצת יכולה גם להיות חיה. משבצת לא יכולה להכיל יותר מחיה אחת.
- לכל חיה יש כמות אנרגיה מסוימת. פעולות שהיא עושה צורכות אנרגיה, ואכילה מספקת לה אנרגיה. כאשר האנרגיה של חיה נגמרת, החיה מתה ויוצאת מהסימולציה.
- 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 ראייה r רואה חלק בגודל $2r+1$ על $2r+1$ של העולם, חלק שהיא במרכזו. לגבי כל משבצת בטווח הראיה, החיה רואה האם יש בה צמחיה, האם יש בה חיה, ואם כן מאיזה מין.

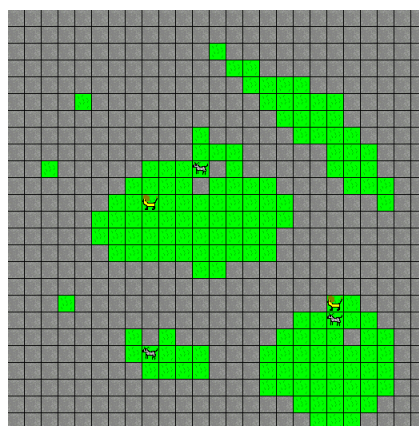
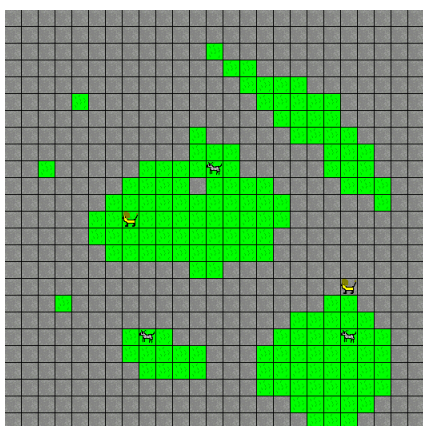
- כאשר מגיע תורה של חיה לפעול, מופעל השירות $act()$ של החיה. שירות זה צריך להחזיר עצם שמתאר את ההחלטה של החיה כיצד לפעול. ההחלטה כוללת האם להשריץ ($spawn$), להתפצל לשתי חיות, אחת בגודל של צאצא של המין

והשנייה בגודל שנותר לאחר הקצאת האנרגיה לצאצא) והאם לנוע למשבצת סמוכה. אם החיה מחליטה לנוע, היא קובעת כמובן לאן היא רוצה לנוע. התנועה מוגבלת למרחק של

משבצת אחת, כשמותרת תנועה באלכסון. כלומר מותר לזוז לאחת משמונה המשבצות המקיפות את החיה (זכרו שהעולם מעגלי).

○ הבהרה: כשחיה משריצה, היא נותנת חלק מהאנרגיה שלה לצאצא, זו האנרגיה איתה מתחילה החיה החדשה

- חיה אוכלת עשב יכולה לנוע רק למשבצת ריקה. אם במשבצת יש עשב, החיה אוכלת אותו, מקבלת 0.5 יחידות אנרגיה, והעשב נעלם מהמשבצת. הוא אולי יגדל בהמשך, באופן אקראי.
- חיה טורפת יכולה לנוע גם למשבצת שיש בה חיה אוכלת עשב (אבל לא חיה טורפת). בתנועה למשבצת מאוכלסת הטורף אוכל את החיה שבמשבצת, מקבל 0.5 יחידות אנרגיה, והחיה שאכלסה את המשבצת מתה.
- אם חיה מחליטה להשריץ ויש לה מספיק אנרגיה לשם כך, נוצרת חיה חדשה מאותו מין באותה משבצת. אם החיה המקורית לא נעה בתור הזה, היא מתה (מכיוון שהמשבצת יכולה להכיל חיה אחת לכל היותר). המיקום של הצאצא בסדר הפעולות שחיות עושות הוא מייד לאחר ההורה. כלומר לאחר שהורה משריץ (ובדרך כלל נע), הצאצא פועל מייד אחריו.
- כאשר מגיע תורה של חיה לנוע, העולם בודק קודם כל האם היא עדיין חיה. עצם ההישרדות של חיה עד לתורה עולה לה 0.05 יחידות אנרגיה. אם יש לה פחות, היא מתה מייד כאשר מגיע תורה. אם יש לה יותר מכך, העולם קורא לשירות act של החיה. הקריאה עצמה עולה לחיה 0.005 יחידות אנרגיה לכל משבצת שהיא רואה פרט למשבצת שלה עצמה. זה מחיר חוש הראיה של החיה. אם החיה החליטה לנוע ויש לה יותר מ-0.1 יחידות אנרגיה, היא נעה וכמות האנרגיה שלה מופחתת ב-0.1. אם היא גם החליטה להשריץ ויש לה עדיין מספיק אנרגיה לשם כך (האנרגיה של צאצא מאותו מין), היא משריצה (הצאצא נשאר במשבצת המקורית שלה) וכמות האנרגיה שלה מופחתת בהתאם. אם החיה נעה למשבצת שבה היא יכולה לאכול, היא אוכלת (ומקבלת אנרגיה), והאוכל נעלם (עשב או חיה אוכלת עשב שנטרפה). שני האיורים הבאים (משמאל לימין) מדגימים מהלך שלם שבו כל החיות פעלו:



- בתחילת מהלך, לפני שכל החיות פעלו, עשב צומח מחדש בהסתברות של 20% בכל משבצת קרחת.
- החיות הראשונות (בתחילת המשחק) נוצרות עם 0.5 יחידות אנרגיה.
- הסימולציה מסתיימת כאשר כל החיות נכחדות, או אחרי 1000 מהלכים.

הנה מספר מנשקים ומחלקות שאתם צריכים לממש.

- המנשק Species מתאר תכונות קבועות המשותפות לכל החיות מאותו מין. לדוגמא, האם החיה אוכלת עשב או טורפת, מה טווח הראיה שלה ומה היא כמות האנרגיה הראשונית שלה. בנוסף מוגדרת פונקציה היוצרת חיה חדשה ממין זה.

```
/**
 * Represents a specific species (e.g. CowSpecies, LionSpecies).
 * There should only be one instance of each class implementing this
 * interface.
 */
public interface Species {

    /**
     * Create a new animal that belongs to this species.
     * There should be no other way to create an animal other than
     * this method.
     * @return A new animal object. The return value is never null.
     * @see Animal
     */
    public Animal newAnimal();

    /**
     * Retrieve the name of the species
     * @return The name of the species (e.g "cow", "lion")
     */
    public String getName();

    /**
     * Determine whether this species is herbivore.
     * A species cannot be both herbivore and carnivore.
     */
    public boolean isHerbivore();

    /**
     * Determine whether this species is carnivore.
     * A species cannot be both herbivore and carnivore.
     */
    public boolean isCarnivore();

    /**
     * Retrieve the range of sight of an animal of this species.
     * All animals belong to this species have the same range of
     * sight.
     * @return An animal's range of sight.
     */
    public int getRangeOfVision();

    /**
     * Retrieve the size of an offspring. The size of an offspring
     * is the initial energy an animal of this species has.
     * When an animal gives birth this amount of energy will be decreased
     * from it.
     * @return a value in the range 0..1
     */
    public double getSizeOfOffspring();
}
```

- המנשק Animal מתאר חיה בודדת. לכל חיה הפניה אל המין שלה וכן לכל חיה שומרת את רמת האנרגיה הנוכחית שלה ומאפשרת שינויים בה. חל איסור מוחלט על החיה לשנות את רמת האנרגיה בעצמה.
לכל חיה יש צוהר דרכו היא רואה את העולם. ראו הסבר מפורט למטה.
השירות act() הוא נקרא בכל פעם שעל החיה לפעול. הערך המוחזר משקף את ההחלטה שלה כיצד לפעול.

```
public interface Animal {

    /**
     * Get the species this animal belongs to.
     * @return - the species object. the return value is never null.
     */
    public Species getSpecies();

    /**
     * Set the animal's view of the world.
     */
    public void setView(World.View view);

    /**
     * Plan the animal's next action.
     * @return - An Action object specifying the outcome of
     * the decision process.
     * @see Action
     */
    public Action act();

    /**
     * get the animal's energy level
     * @return - the animal's current energy level.
     */
    public double getEnergy();

    /**
     * Decrease the animal's energy level by a specified amount
     * @param amount - the amount of energy to decrease. this value
     *                 should be positive
     */
    public void decreaseEnergy(double amount);

    /**
     * Increase the animal's energy level by a specified amount
     * @param amount - the amount of energy to increase. this value
     *                 should be positive
     */
    public void increaseEnergy(double amount);
}
```

- המנשק Tile מתאר משבצת אחת בעולם. המנשק מאפשר קריאה בלבד של מצב המשבצת.

```

/**
 * 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 any)
     * @return The animal occupying this tile, or null if none exists.
     */
    public Animal getAnimal() {
        // TODO your code goes here
    }

    /**
     * Check if there is vegetation on this tile.
     */
    public boolean hasVegetation() {
        // TODO your code goes here
    }
}

```

- המחלקה Action מתארת את הפעולה אותה מבקשת החיה להוציא לפועל. החיה יכולה לציין באם ברצונה להשריץ וכן את תנועתה. חיה יכולה לנוע משבצת אחת בלבד בכל תור (ימינה, שמאלה, למעלה, למטה או משבצת אחת באלכסון).

```

public class Action {

    public boolean spawn() {
        // TODO your code goes here
    }

    public void setSpawn(boolean spawn) {
        // TODO your code goes here
    }

    public int getHorizontalMove() {
        // TODO your code goes here
    }

    public int getVerticalMove() {
        // TODO your code goes here
    }

    public void setMove(int horizontal, int vertical) {
        // TODO your code goes here
    }
}

```

- המחלקה World אחראית על מצב העולם ושינויו במהלך הרצת הסימולציה. לכל חיה קיים צוהר (View) דרכו היא יכולה להסתכל על העולם. חלון זה מוגבל בגודלו בהתאם ליכולת הראיה של החיה. חיה יכולה להתבונן במשבצת בעולם ע"י קריאה לשיחות getTileAt של ה-View. הקואורדינטות הן במונחי החיה, כלומר משבצת (0,0) היא המשבצת שבה היא נמצאת. המשבצת לשמאלה היא (-1, 0) וכו'. תפקידו של ה View לתרגם קואורדינטות יחסיות אלו לקואורדינטות מוחלטות של העולם ולהחזיר את ה Tile המתאים. ה View מוגדר כמחלקה פנימית בעולם כדי לאפשר את התרגום הנ"ל ביתר קלות ללא צורך בחשיפת המבנה הפנימי של העולם.

```

public class World {

    /**
     * create a new world with size n * n.
     * initially, the world is empty - no animals and all squares
     * are barren.
     */
    public World(int n) {
        // TODO your code goes here
    }

    /**
     * Create a new animal of the specified species at
     * coordinate (x, y).
     * @param s - The species the new animal belongs to
     * @param x - the x coordinate
     * @param y - the y coordinate
     * @return - Returns 'true' if the animal was created as
     * requested and 'false' otherwise (in case another
     * animal occupies this tile).
     */
    public boolean createAnimalAt(Species s, int x, int y) {
        // TODO your code goes here
    }

    /**
     * Run a simulation of the world.
     * See more details in the assignment description
     */
    public void simulate() {
        // TODO your code goes here
    }

    public class View {
        public View(...) {
            // TODO your code goes here
        }

        public Tile getTileAt(int h, int v) {
            // TODO your code goes here
        }

        // TODO choose the internal representation of the view
    }

    // TODO choose the internal representation of the world
}

```

שימו לב, **אסור (!)** לשנות את המנשקים והמחלקות הנ"ל, כלומר לא לשנות שמות של פונקציות, להוסיף/להוריד פרמטרים וכו'. מותר לכם במחלקות שאתם מממשים להוסיף שירותי עזר פרטיים במידת הצורך.

המטלה

1. ממשו את המחלקה World, מחלקה שמייצגת אוכל עשב (למשל Cow או Zebra וכדומה), מחלקה שמייצגת טורף (Tiger או Lion וכדומה), ואת המחלקות שמייצגות את המינים שלהם (CowSpecies ו-LionSpecies, למשל). בעת המימוש הקפידו לא לשכפל קוד ע"י שימוש מושכל בירושה.
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 Play 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?

הדרכה

אחד הקשיים בתוכנית הזו הוא למצוא את מבני הנתונים מתאימים עבור המחלקה World. עשו שימוש נרחב כל האפשר במחלקות קיימות (Set, List, Map 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 נתוני זמינים לכם במחלקות והממשקים שהוגדרו ולאילו תצטרכו להגדיר מבני נתונים בעצמכם.

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 שידאגתם שהמערכת עובדת כראוי ליצור אלגוריתמים מסובכים יותר. דוגמא לאלגוריתם פשוט יכולה להיות, בדוק אם קיים מזון באחת המשבצות הקרובות, אם כן נוע למשבצת זו, אם לא זוז בצורה רנדומאלית.

הוראות הגשה

- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה ע"י המערכת VirtualTAU (<http://virtual.tau.ac.il>).
- הגשת התרגיל תתבצע ע"י יצירת קובץ zip בנושא 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 התכניות שהתבקשתם לכתוב.
- קובץ טקסט עם העתק של כל קבצי ה Java.