תרגיל 5

תכנות מונחה עצמים

תאריך הגשה: במודל.

קבצי הבדיקה:

- לשאלה 1: IteratorTest.java
- לשאלה 2: MinesTester.java

ובנוסף, כמו קודם, שימו את Tester.java, ואת ExDetails.java בחבילה main.

שאלה 1 (package iterator)

נבנה שתי מחלקות שיש להן איטרטור.

TwoArrays

למחלקה זו יהיה בנאי:

```
public TwoArrays(int[] a1, int[] a2)
```

והיא תממש את הממשק Iterable<Integer> על כל הכרוך בכך. האיטרטור שלה יחזיר בקריאה ראשונה ל-next את האבר הראשון של a1, אחר כך את הראשון של a2, ואז את הבא של a1 וכך הלאה, עד ששני המערכים נגמרים. למשל:

```
int[] a1 = { 1, 2, 3, 4 };
int[] a2 = { 100, 101, 102, 103, 104, 105, 106 };

TwoArrays aa = new TwoArrays(a1, a2);
for (int i : aa)
	System.out.print(i + " ");
```

ידפיס:

```
1 100 2 101 3 102 4 103 104 105 106
```

מי שלא זוכר על מה מדובר מוזמן להסתכל בהרצאה (וגם בתרגול כיתה) - יש שם כמה דוגמאות לאיטרטורים ואיך לגרום למחלקה להיות Iterable.

Combined<E>

זוהי הכללה של המחלקה הקודמת. יש לה בנאי שמקבל שני Iterable<E> - כלומר היא תוכל לקבל כל דבר שמממש Iterable<E>, כמו List או Set וכו':

```
public Combined(Iterable<E> first, Iterable<E> second)
```

היא בעצמה גם מממשת Iterable<E>, והרעיון הוא כמו קודם, שהיא תחזיר קודם מה-Iterable הראשון שקיבלה, אז מהשני וכך הלאה, עד ששניהם יגמרו. כך למשל:

```
public static void main(String[] args) {
	List<String> list1 = Arrays.asList("one", "two", "three");
	List<String> list2 = Arrays.asList("A", "B", "C", "D", "E");

	Combined<String> c = new Combined<>(list2, list1);
	for (String s : c)
		System.out.print(s + " ");
}
```

ידפיס:

```
A one B two C three D E
```

שאלה 2 (package mines)

המטרה שלנו היא לכתוב את משחק שולה המוקשים הידוע. מי שלא מכיר מוזמן לחפש minesweeper בגוגל, ולשחק. בשלב הראשון נכתוב

Mines

זוהי מחלקה שנועדה לנהל את המשחק עצמו, בנפרד מהעובדה שהוא יהיה חלק מאפליקציה גרפית.

public Mines(int height, int width, int numMines)	בנאי לאתחול משחק ברוחב ובגובה הנתונים, ובו בדיוק numMines מוקשים המונחים באקראי.
public boolean addMine(int i, int j)	מוסיף מוקש במקום במיקום הנתון. ניתן להניח ששיטה זו נקראת רק מיד לאחר הבנאי, ולפני כל

השיטות האחרות (למעשה יש אותה כדי שאוכל לעשות בדיקות)	
מסמן שהמשתמש פותח את המיקום הזה. מחזיר true אם זהו לא מוקש. בנוסף, אם ליד מיקום זה אין אף מוקש, אז פותח את כל המיקומים השכנים - וממשיך כך רקורסיבית.	public boolean open(int i, int j)
שם דגל במיקום הנתון, או מסיר אותו אם כבר יש שם דגל.	public void toggleFlag(int x, int y)
מחזיר אמת אם כל המיקומים שאינם מוקשים פתוחים.	public boolean isDone()
מחזיר ייצוג כמחרוזת של המיקום: אם המיקום סגור יחזיר "F" אם יש עליו דגל, ואחרת "." אם המיקום פתוח, יחזיר "X" אם זהו מוקש, אחרת את מספר המוקשים שליד ואם מספר זה הוא 0 אז " ".	public String get(int i, int j)
יקבע את ערך השדה showAll. שדה זה מאותחל ל-false, אבל כשהוא true, אז ב-get וגם ב-toString, ערך החזרה הוא כאילו כל התאים פתוחים (אבל הוא לא באמת פותח אותם)	public void setShowAll(boolean showAll)
מחזיר תיאור של הלוח כמחרוזת לפי get לכל מיקום.	public String toString()

למשל:

```
Mines m = new Mines(3, 4, 0);
m.addMine(0, 1);
m.addMine(2, 3);
m.open(2, 0);
System.out.println(m);
m.toggleFlag(0, 1);
System.out.println(m);
```

ידפיס:

```
....
112.
  1.

.F..
```

112.
1.

כמה הערות:

1. אפשר להניח שכל הקלטים חוקיים.
2. כן צריך לשים לב למקרים הגיוניים - כמו שמישהו מנסה לפתוח מיקום שכבר פתוח.
3. אין צורך להתייחס לניצחון או להפסד פה. כלומר אם נפתח מוקש או שאין עוד מה לפתוח אז אתם יכולים להתנהג איך שמתאים לכם להמשך הריצה.
4. מומלץ לכתוב מחלקה פנימית שתתאר מה קורה במיקום אחד:
   - האם יש שם מוקש, האם פתוח, האם יש שם דגל, כמה מוקשים יש ליד - כולל שיטת toString.
   - ואז המחלקה הראשית יהיה מערך דו מימדי של כאלו.
5. עוד המלצה מעניינת, היא לכתוב מחלקה פנימית שמתארת מיקום בודד. כלומר יש לה פשוט שני שדות i ו-j. אז תוכלו להשתמש בה בכל מני דברים פנימיים.
   - אחד הדברים היפים שאפשר לעשות עם זה, הוא להוסיף לה שיטה שמחזירה את אוסף מיקומי השכנים שלה (כרשימה או כל אוסף אחר - או אפילו כאיטרטור אם ממש בא לכם). עקרונית יש שמונה שכנים, אבל אולי חלק מחוץ ללוח. יש לפחות שני מקומות שונים בהם שיטה כזאת יכולה לעזור.

MinesFX

מחלקה זו תהיה הממשק הגרפי של המשחק שלנו. היא תממש את כל ענייני הממשק, ותחזיק מופע של המחלקה Mines, בו תשתמש כדי לעשות כל דבר שקשור למשחק עצמו. בסופו של דבר המשחק אומר להתנהל כמו בוידאו הקצר פה.

כמה דרישות והמלצות:

1. כמו שנאמר, הלוגיקה של המשחק צריכה להיות דרך המחלקה Mines ולא משוכפלת פה.
2. את המסך הראשי, מלבד הגריד עצמו יש לעשות ב- Scene Builder.
   - בתוכו יהיה רכיב כלשהו, למשל StackPane, שיהיה ריק, ולתוכו תכניסו את הגריד.
3. הגריד יווצר בקוד רגיל ולא ב-Scene Builder, ולכן גם הטיפול בארועים בו קורה בקוד רגיל (כמו בהרצאה).
4. כמובן שתהיה לכם גם מחלקת controller של ה-Scene Builder - תצטרכו לתקשר איתה כמובן. תזכרו שאתם יכולים לשנות את הקוד שלה איך שבא לכם.
5. שווה מאוד להסתכל על המסמך שימוש והתקנה של Scene Builder, יש שם פרטים קטנים וחשובים על העבודה איתו.
6. הסימון של דגל (F) נעשה ע"י לחיצה על הכפתור הימני. תמיכה בו היא רשות, ותוסיף לכם 5 נקודות לציון התרגיל.
7. בהינתן לחיצה על כפתור תצטרכו למצוא מיהם ה-x וה-y שלו. יש כמה פתרונות לזה.

- אולי הכי פשוט הוא להגדיר תת מחלקה של כפתור שבה יש גם שדות x ו-y, ואלו יהיו הכפתורים שלכם - לכן ברגע שילחצו על כפתור, אז בעזרת getSource תוכלו לקבל את הכפתור שלכם וממנו לקבל את x ו-y.
- עוד דרך היא שה-EventHandler של לחיצה על כפתור תהיה מחלקה שיכולה לקבל בבנאי x ו-y, ואז לכל כפתור תצרו אוביקט חדש מהסוג הזה שמאותחל עם x ו-y משלו.
- בקיצור, כשתגיעו לשם, תפתרו את הבעיה.