

האוניברסיטה העברית בירושלים
ביה"ס להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא לתכנות מונחה עצמים (67125)

(מועד ב')

מרצה: יואב קן-תור

מתרגלים: זיו בן-אהרון, נגה רוטמן, ג'ונה הריס

תעודת זהות של הסטודנט/ית:

הוראות:

- משך המבחן הינו 3 שעות.
- המבחן הינו עם חומר סגור. אין להשתמש בכל חומר עזר או אמצעי חישוב שהוא, לרבות מחשבון.
- למבחן שני חלקים:
 - חלק א' – כולל 6 שאלות, מהן עליכם לענות על 5. כל שאלה שווה 10 נקודות.
 - חלק ב' – כולל 2 שאלות קוד גדולות, ועליכם לענות על שתיהן. כל שאלה שווה 25 נקודות. החלוקה לסעיפים היינה לנוחות קריאה בלבד.

יש לענות על כל שאלות המבחן במחברת הבחינה.

שימו לב לנקודות הבאות:

- "Less is more" – רשמו תשובה קצרה ומדויקת, שלא תשאיר מקום לספק שאכן הבנת את החומר.
- ייתכן שהקוד המוגש יעבוד, אבל למבחן בתכנות מונחה עצמים זה לא מספיק. הקוד צריך להיות קריא ומתוכנן היטב. תעד את הקוד (אפשר בעברית) רק אם יש חשש שמישהו לא יבין אותו.
- זכור כי בעיצוב אין תשובה אחת נכונה, אבל בהחלט יש תשובות שהן לא נכונות. נקודות מלאות יתקבלו רק באם השתמשת באחת מהדרכים המתאימות ביותר.
- באם תענה על יותר שאלות מן הנדרש, רק הראשונות ייבדקו.
- אנו ממליצים לקרוא את כל המבחן מתחילתו עד סופו לפני תחילת הפתרון.
- המבחן כתוב בלשון זכר, אך מיועד לכלל הסטודנטיות והסטודנטים הלוקחים את הקורס במידה שווה.

בהצלחה!

חלק א'

- כולל 6 שאלות, מהן עליכם לענות על 5. כל שאלה שווה 10 נקודות. הקפידו לנמק כאשר התבקשתם.
- חלק מן השאלות כוללות מספר סעיפים, אשר מופרדים לתיבות שונות במודל ומסומנים בהתאם (למשל, אם שאלה 1 כוללת שני סעיפים הם יסומנו כ"שאלה 1.1" ו"שאלה 1.2"). הקפידו לענות על כל חלקי השאלות.
- בשאלות הכוללות כתיבת קוד, הקפידו לכתוב קוד קצר, פשוט ומובן. רק מימושים כאלו יקבלו ניקוד מלא.

שאלה 1

בחרו את האפשרות הנכונה עבור כל היגד בפסקה הבאה (כתבו את מספר המשפט ואת המילה הנכונה במחברת):

1. פונקציה סטטית (static method) במחלקה גנרית `class MyClass<T>` יכולה/לא יכולה לקבל ארגומנט מהטיפוס `T`.
2. לאובייקט מסוג `enum` יכול/לא יכול להיות בנאי (constructor).
3. ממשק פונקציונלי (functional interface) חייב להגדיר תשובה לפחות/בדיוק/עד אבסטרקטית (abstract method) אחת.
4. מהרשימה הבאה: `List<? extends Number> foo1 = new ArrayList<Number>();` [Number/null/Number.Integer.Double] ניתן לקרוא אובייקטים מטיפוס `[Number/Integer.Double/Number.Integer.Double]`.
5. הרשימה הבאה: `List<? extends Number> foo2;` יכולה להיות רפרנס לרשימה של [Number/Integer.Double/Number.Integer.Double].

שאלה 2

תארו בקצרה מהו מנגנון reflection בג'אווה. תנו דוגמא למקרה בו כדאי להשתמש ב reflections והסבירו מדוע באופן כללי לא כדאי להשתמש במנגנון זה.

שאלה 3

(א) מה תהיה תוצאת הקוד הבא:

```
public class PalindromeAnalyzer {
    String s;
    PalindromeAnalyzer(String s) {
        this.s = s;
    }

    boolean analyze() {
        int n = s.length();
        for (int i = 0; i <= n / 2; i++) {
            if (s.charAt(i) != s.charAt(n - i - 1))
                return false;
        }
        return true;
    }

    public static void main(String[] args) {
        String s = "1234567";
        PalindromeAnalyzer p = new PalindromeAnalyzer(s);
        System.out.print(p.analyze());
        s = "1234321";
        System.out.print(p.analyze());
    }
}
```

- A. true, true
- B. true, false
- C. false, true
- D. false, false

(ב) נמקו בקצרה את תשובתכם.

שאלה 4

(א) נתון הקוד הבא:

```
package snbl;
```

```
class Ben {
```

```
    A. _____ int myL;
```

```
        Ben(int l){
```

```
            myL = l;
```

```
        }
```

```
    B. _____ int getAndSetMyL(int addToL){
```

```
        return myL += addToL;
```

```
    }
```

```
}
```

```
package snbl;
```

```
public class SNBL {
```

```
    C. _____ Ben l = new Ben(2);
```

```
        public static void main(String[] args) {
```

```
            SNBL mySnbl = new SNBL();
```

```
            mySnbl.kal();
```

```
            SNBL mySnbl2 = new SNBL();
```

```
            mySnbl2.kal();
```

```
        }
```

```
        D. _____ void kal(){
```

```
            int newl = l.getAndSetMyL(2);
```

```
            System.out.println(l.getAndSetMyL(0));
```

```
            System.out.println(newl);
```

```
            Ben l2 = new Ben(newl + l.getAndSetMyL(newl));
```

```
            System.out.println(l2.getAndSetMyL(0));
```

```
            System.out.println(l2.getAndSetMyL(newl+l.getAndSetMyL(0)));
```

```
        }
```

```
}
```

השלימו את המודיפייירים (modifiers) הכי מגבילים על מנת שיודפס:

4

4

12

28

30

30

90

210

(ב) נמקו את בחירתכם בקצרה.

שאלה 5

כתבו מתודה המקבלת שתי מחרוזות ומחזירה true אם מדובר באנגרמות ואחרת מחזירה false. על המתודה להתייחס לאותיות רישיות (capital letters) ולא רשיות כאותה האות (כלומר, מופע של "A" זהה למופע של "a"). על החתימת המתודה להיות:

```
public static boolean isAnagram(String a, String b);
```

אנגרמות הן מילים בהן יש את אותו מספר האותיות (מספר המופעים של כל אות זהה בכל המילים)

דוגמאות:

- קלט: "Note", "Tone" פלט: true
- קלט: "Tone", "Nott" פלט: false
- קלט: "Aabbcc", "abacbc" פלט: true
- קלט: "aabbcc", "abacac" פלט: false

הנחות מקלות:

1. אורך כל מחרוזת אינו עולה על 50 תווים
2. התווים האפשריים הם רק האותיות [A-Z a-z]

שאלה 6

נתון הקוד הבא:

```
interface Advanceable<T> {
    void advance(T units);
}

abstract class Clock implements Advanceable<Integer> {
    protected int hour;
    Clock(int initialHour) {
        this.hour = initialHour;
    }
    public int getHour() {
        return this.hour;
    }
}
```

עליכם לממש מתודה עם החתימה הבאה:

```
public static Clock buildClock(int initialHour, int totalHours)
```

הפונקציה צריכה להחזיר שעון (Clock) עם שעה התחלתית initialHour ועם כמות שעות כוללת totalHours. כל שעון תומך בפעולת קידום (advance) אשר מתנהגת באופן הבא: כאשר שעון עם שעה נוכחית hour וכמות שעות כוללת totalHours מקודם ב-X שעות, השעה לאחר הקידום תהיה

$$(\text{hour} + X) \% \text{totalHours}$$

שימו לב: עליכם לממש קוד אך ורק בתוך המתודה buildClock.

ראו את קטע הקוד הבא אשר משתמש במתודה buildClock ואת הפלט הדרוש:

```
public static void main(String[] args) {
    Clock clock12 = buildClock(10, 12);
    Clock clock24 = buildClock(10, 24);
    System.out.println(clock12.getHour()); // prints 10
    System.out.println(clock24.getHour()); // prints 10
    clock12.advance(5);
    clock24.advance(5);
    System.out.println(clock12.getHour()); // prints 3
    System.out.println(clock24.getHour()); // prints 15
}
```

חלק ב'

- קראו בעיון את שתי השאלות הבאות. קראו כל שאלה עד הסוף לפני תחילת הפתרון.
- לשאלות יכולות להיות מספר סעיפים ותתי-סעיפים. החלוקה לסעיפים נועדה לנוחות הפתרון בלבד.
- שאלה ללא נימוק תאבד נקודות.
- כאשר אתם כותבים נימוק הקפידו לפרט באילו עקרונות ותבניות עיצוב (design patterns) בחרתם להשתמש ומדוע.
- הקפידו על עיצוב יעיל. העיצוב ייבחן על בסיס:
 - שימוש בכלי המתאים ביותר מהכלים שנלמדו בקורס.
 - שימוש במינימום ההכרחי של מחלקות וממשקים.
- בכל פעם שהוגדרה מתודה למימושכם, אם חתימתה חסרה, עליכם להשלים אותה לפי שיקולכם.
- ניתן להשתמש בקוד Java של מחלקות שלמדנו עליהן (לדוגמה הפונקציה sort של Collections).

שאלה 1

סעיף א'

רקע:

בעקבות מתקפת פתע של כוחות סילונים שהותירו את קפטן ג'ונה אדאמה בפיקוד על שתי משחתות מסוג "כוכב-קרב" החליט יוהן קו טוב, נשיא 12 המושבות, לקדם את אדאמה לדרגת אדמירל. בצעד ראשון החליט אדמירל אדאמה שמעתה ואילך לכל ספינה בצי יהיה נשק יחיד וסוג מגן יחיד.

מימוש:

ממשו מחלקת Ship לה נשק ומגן וכן שדה health המציין את הבריאות של הספינה. על המחלקה לממש את המתודות הבאות:

המתודה בודקת כמה נזק נספג ע"י המגן, את שארית הנזק היא מחסירה משדה הבריאות של הספינה ומחזירה כמה נזק הוסב לספינה עצמה.	____ getHit (Weapon weapon)
מתודה זו מחזירה את הבריאות היחסית של הספינה	____ getHealth ()
מתודה זו מקבלת מטרה להתקפה, ותוקפת אותה באמצעות הנשק של היחידה. מחזירה את כמות הנזק שנגרמה למטרה.	____ attack (____ target)
מתודה זו מתקנת את הספינה - עליה להוסיף את כמות הבריאות שהתקבלה לבריאות הנוכחית של הספינה. על ערך הבריאות לא לעבור את 100%.	getFixed (int addedHealth)

שימו לב שחלק מהמימוש מצריך השלמת טיפוס ההחזרה של המתודות (היכן שמופיע קו תחתון "_") לפי רצונכם. ודאו כי ה-access modifiers של המתודות מתאימים למימוש.

יש לממש את המחלקה כך שיצירת ספינה אפשרית רק באמצעות נשק מסויים ושריון מסויים. לפניכם הממשקים של Weapon ו-Armor, ודאו כי המתודות שאתם מממשים מתאימות להם. יש להשלים את הטיפוס במתודה attack של הממשק Weapon בהתאם למימוש שלכם.

```
public interface Weapon {  
    /**  
     * How much damage does this weapon deliver?  
     * @return int value representing damage delivered  
     */  
    double getBaseDamage ();  
    /**  
     * deliver "getDamage" points onto target.  
     * @param target The object to be targeted.  
     * @return How much damage was delivered to target.  
     */  
    double attack ( ____ target );  
}
```

```

public interface Armour {
    /**
     * calculate how much damage this armor can absorb from a given weapon.
     * @param weapon The weapon attacking the unit wearing this armor.
     * @return The amount of damage passing through the armor (weapon base damage
     minus amount absorbed)
     */
    double absorbDamage(Weapon weapon);

    /**
     * Set new durability percentage for this armour
     * @param newPercent Durability (in percentage points) to be set for this
     armour.
     */
    void setDurabilty(int newPercent);

    /**
     * Get the current durability of the armour.
     * @return Current durability
     */
    int getDurability();
}

```

הערות:

- מותר לכם להוסיף טיפוסים
- שמרו על עקרונות מימוש מינימלי - ניקוד מלא ינתן לפתרונות קצרים המראים עקרונות עיצוב OOP טובים
- בכל מקום בשאלה המופיע קו תחתון "_" הינכם נדרשים להשלים את הקוד על פי שיקולכם
- כאשר ספינה נוצרת פרמטר ה-health שלה מאותחל ל-100 ובשום שלב לא יכול לרדת מתחת ל-0

סעיף ב'

רקע:

לאחר מספר התכתשויות עם להקים של ספינות סילוניות, החליט אדמירל אדאמה כי חשוב שהטייסות יהיו הומוגניות. כלומר בכל טייסת יהיו ספינות בעלות אותו הנשק ואותו המיגון.

מימוש:

ממשו מחלקת Squadron הכוללת את הפונקציות הבאות אשר מכילה רק חלליות בעלי אותו מגן ואותו נשק. לכל טייסת יש גודל מקסימלי הניתן לה בעת יצירת הטייסת ואינו ניתן לשינוי.

מתודה זו מוסיפה ספינה לטייסת כל עוד הטייסת לא חרגה מגודלה המקסימלי. מחזירה אמת אם הספינה הוספה או שקר אם לא ניתן להוסיף את הספינה	boolean add(____)
מתודה זו מחזירה את הספינה ה-i בטייסת אם קיימת כזו ספינה	____ get(int i)
מתודה זו מחזירה אמת אם ספינה מסויימת קיימת בטייסת או שקר אחרת	boolean contains(____)
מתודה זו מסירה ספינה מסויימת מהטייסת. היא מחזירה אמת אם הספינה הייתה קיימת בטייסת והוסרה בהצלחה או שקר אחרת.	boolean remove(____)

סעיף ג'

רקע:

אחרי שהצי' בהנהגת אדמירל אדאמה הגיע למעגן ראגנאר להצטיידות מחדש, גילו הכוחות מאגר של כוורות רקטות אשר יכולות לפגוע בטייסות שלמות במקביל.

מימוש:

עדכנו את תכניתכם כך שגם טייסות יוכלו להפגע על ידי כלי נשק, כלומר ממשו לטייסת את השיטה:

מתודה זו מעבירה את ההתקפה ע"י הנשק <code>weapon</code> אל כל אחת מהספינות בטייסת. על המתודה להחזיר את סכום הנזק שפעל על כל אחת מהספינות בסה"כ.	<code>_____ getHit(Weapon weapon)</code>
--	--

ודאו כי כל רכיבי התוכנה מסוגלים לעבוד יחדיו. לשם כך נספק פסודו-קוד של תכנית, אותו עליכם לתרגם לקוד ג'אוה שיוכל לרוץ בהנתן המימוש שלכם:

פונקציית main:

צור טייסת אחת בשם `redSquadron` בגודל 5
הוסף לטייסת 5 ווייפרים (ספינות קרב) בעלי תותח (Cannon) ושיריון מתכת (MetalPlates)
צור טייסת נוספת בשם `blueSquadron` בגודל 2
הוסף לטייסת 2 ראפטורים (ספינות תמיכה) בעלי כוורת רקטות (Rocket Pod) ושיריון מתכת (Metal Plate)

בצע התקפה של הטייסת האדומה ע"י הטייסת הכחולה. כלומר - על כל אחת מהספינות בטייסת הכחולה לתקוף את הטייסת האדומה.

הערות:

הניחו כי המחלקות הבאות קיימות באופן התואם את העיצוב שלכם:

- `RocketPod implements Weapon`
- `Cannon implements Weapon`
- `MetalPlates implements Armor`

שאלה 2

רקע

לאחר הצלחתו המסחררת של קים באג אין בשוק הצפון קוריאני, החליט לפתוח עסק חדש בכוכב Risa - כוכב הפנאי המפורסם ביותר בגלקסיה. קים באג אין החליט לפתוח קזינו וירטואלי חדש, בו משחקי הקלפים הם אוטומטיים. כל שחקן מתחיל עם אסטרטגיה קבועה ומקווה לנצח במשחק הקלפים בקזינו. התבקשתם לעזור לקים לממש את תהליך המשחק.

למזלנו המתכנת הקודם (לפני שהתפטר ממשרתו) מימש וסיפק לכם את מחלקת Deck המוגדרת ע"י ה-API הבא:

הבנאי של המחלקה Deck. ניתן להגדיר את מספר הקלפים בחבילה ואת הערך של הקלף החזק ביותר בחבילה	<pre>public Deck() public Deck(int numCards) public Deck(int numCards, int maxVal)</pre>
מתודה המחזירה קלף אקראי מהחבילה ומוציאה אותו מהחבילה (כאשר קלפים מיוצגים על ידי int)	<pre>public int pickRandomCard()</pre>
מתודה שמחזירה true אם החבילה ריקה (כלומר, לא נשארו עוד קלפים בחבילה)	<pre>public boolean isEmpty()</pre>

(מחלקה זו נתונה לכם. כלומר, לא צריך לממש את מחלקת Deck)

בשאלה זו נתמקד בשתי מחלקות: Dealer ו-Player.

Dealer מוגדר ע"י ה-API הבא:

הבנאי של מחלקת Dealer. הדילר מאותחל עם סט של חוקי משחק	<pre>public Dealer(Comparator<Integer> gameRules)</pre>
מתודה המחזירה את המנצח של המשחק בו players שיחקו עם החבילה deck תחת החוקים שהוגדרו בבנאי של האובייקט dealer	<pre>public Player dealGame(Player[] players, Deck deck)</pre>

Player מוגדר ע"י ה-API הבא:

הבנאי של מחלקת Player. השחקן מאותחל עם סט של חוקי משחק ואסטרטגיית משחק (הסבר בסעיף ב)	<pre>public Player(_____, Comparator<Integer> rules)</pre>
מתודה המחזירה קלף מהיד של השחקן לפי האסטרטגיה שלו. לאחר שהשחקן שיחק את אותו הקלף, הקלף יוצא מידו	<pre>public int playCard()</pre>
מתודה שמוסיפה קלף לידו של השחקן	<pre>public void addCard(int card)</pre>

בנוסף, נגדיר את משחק השלום והשגשוג 7:

כללי המשחק הם כדלהלן: עבור שני קלפים c1, c2, אם רק קלף אחד מתחלק ב-7, זה הקלף המנצח. אם שניהם מתחלקים ב-7 אז הקלף הקטן יותר מנצח. אם אין קלף המתחלק ב-7, הקלף בעל הערך הנמוך ביותר מנצח.

לכל אורך השאלה, אתם מוזמנים להוסיף מתודות\מחלקות לפי שיקולכם להכל חוץ מל-Deck

סעיף א'

משחק בקזינו מתנהל באופן הבא:

1. בתחילת המשחק, הדילר מחלק 5 קלפים לכל שחקן. סדר החלוקה הוא 5 קלפים לשחקן הראשון ואז 5 לשני וכן הלאה
 2. בכל סיבוב הדילר מחלק קלף לכל שחקן על פי הסדר
 3. כל סיבוב הדילר אוסף קלף אחד בלבד מכל שחקן על פי סדר השחקנים
 4. לאחר שאסף את הקלפים, הדילר מחליט איזה קלף מנצח בהתאם לחוקי המשחק
 5. הדילר נותן נקודה אחת למנצח
 6. המשחק נמשך עד שאין קלפים בחבילה ובידיהם של השחקנים
 7. המנצח של כל המשחק הוא השחקן עם הכי הרבה נקודות
- המנצח בכל סיבוב הוא השחקן ששיחק את הקלף החזק ביותר. הקלף החזק ביותר תלוי בחוקי המשחק. לכל משחק יש Comparator שמגדיר סדר על הקלפים לפיו נקבע המנצח.

א.1: ממשו את מחלקת Dealer על פי ה-API המופיע למעלה. נמקו את מימושכם.

א.2: ממשו את מתודת main תחת מחלקת Dealer. ממשו בה Comparator עבור משחק השלום והשגשוג 7 ואתחלו דילר עם חוקי משחק זה.

הערות, הבהרות והנחות מקלות:

- לדילר לא חשוב איך השחקן בוחר את הקלף כל עוד הוא מקבל אותו ממתודת addCard
- ניתן לקבל קלף משחקן באמצעות מתודת playCard
- הקלפים ב-Deck מיוצגים כ-int
- מחלקת Dealer יכולה להריץ כל משחק שחוקיו ניתנים להצגה על ידי Comparator של Integers
- אם שני שחקנים שיחקו קלף בעל אותו ערך, השחקן שינצח הוא הראשון ששיחק את הקלף
- אם יש פחות קלפים מאשר שחקנים, הדילר מחלק את הקלפים הנותרים לשחקנים לפי החוקים למעלה על פי סדר השחקנים. כלומר, לדילר לא איכפת כמה קלפים נותרו בחבילה (ספציפית, נניח שיש 3 שחקנים ו-12 קלפים בחבילה בתחילת המשחק, הראשון והשני יקבלו 5 קלפים והשלישי יקבל 2)
- כל תור שחקן ללא קלפים יחזיר את המספר "0" כקלף

סעיף ב'

בתוכנית של קים באג אין, כל אורח בקזינו יוצר לעצמו שחקן בעל אסטרטגיה. הוחלט שאסטרטגיה תלויה ב-3 דברים: הקלפים שיש לשחקן ביד, כללי המשחק (המוגדרים על ידי Comparator) ומספר התור הנוכחי.

כל שחקן בקזינו מביא בעצמו instance של מחלקת Player עם האסטרטגיה הספציפית שהוא בחר. עליכם להחליט איך השחקנים יגדירו את האסטרטגיות שלהם, כאשר מותר להם להשתמש רק בשלושת הנתונים שהוזכרו.

ב.1: ממשו את מחלקת Player המסוגלת לפעול עם מספר אסטרטגיות שונות. נמקו בקצרה איך החלטתם להוסיף לשחקנים אסטרטגיות ולמה מימשתם בדרך הזאת וכיצד מימוש זה מבטיח חשיפה מינימלית של אסטרטגיית השחקן.

ב.2: ממשו אחת משלוש האסטרטגיות הבאות:

1. **BestFirst** : השחקן הזה תמיד שם את הקלף הכי חזק שלו בכל תור
2. **FoxTrot** : השחקן הזה שם קלפים חזקים וחלשים בתורות לסירוגין (כלומר, בתור הראשון את הכי חזק שלו, בשני הכי חלש, שלישי חזק וכו')
3. **WorstFirst** : השחקן הזה תמיד שם את הקלף הכי חלש שלו בכל תור

ב.3: אתחלו 2 שחקנים עם האסטרטגיה שמימשתם והשלימו את שאר הקוד הדרוש להרצת משחק של ה-Dealer מ-2.2 ואותם השחקנים

הבהרות:

- שחקן צריך להיות מסוגל להשתמש בכל אסטרטגיה המשתמשת בקלפים ביד השחקן, כללי המשחק ומספר התור הנוכחי. חשבו באיזה אופן יהיה הכי קל לממש את זה והגדירו את הבנאי של Player כדי לתמוך במימוש שלכם
- שחקן לא יכול לשנות אסטרטגיה במהלך המשחק
- מותר להריץ משחק עם חבילה שמאותחלת ללא ארגומנטים (כלומר; Deck d = new Deck;)

סעיף ג'

כתבו 2 טסטים (unit tests) למימוש של מתודת dealGame במחלקת Dealer בעזרת JUnit. לאחר מימוש הטסטים, כתבו נימוק קצר המסביר מה בדקתם ולמה.

שימו לב: קיימים טסטים יותר ופחות מרכזיים - עליכם להחליט מה כדאי לבדוק.

תזכורת:

ב-JUnit משתמשים באנוטציות (annotations) הבאות:

אנוטציה למתודה שמריצים לפני שכל הטסטים רצים	@BeforeClass
אנוטציה למתודה שמריצים לפני כל אחד מהטסטים	@Before
אנוטציה למתודה של unit test	@Test
אנוטציה למתודה שמריצים אחרי כל אחד מהטסטים	@After
אנוטציה למתודה שמריצים אחרי שכל הטסטים הסתיימו	@AfterClass

Class HashMap<K,V>	
V get(Object key)	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key
Set<K> keySet()	Returns a Set view of the keys contained in this map
V put(K key, V value)	Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.
V remove(Object key)	Removes the mapping for the specified key from this map if present
Collection<V> values()	Returns a Collection of the values contained in this map

Class HashSet<E>	
boolean add(E e)	Adds the specified element to this set if it is not already present
boolean contains(Object o)	Returns true if this set contains the specified element
boolean remove(Object o)	Removes the specified element from this set if it is present
Iterator<E> iterator()	Returns an iterator over the elements in this set

Class LinkedList<E>	
boolean add(E e)	Appends the specified element to the end of this list
void clear()	Removes all of the elements from this list
E poll()	Retrieves and removes the head (first element) of this list
E get(int index)	Returns the element at the specified position in this list

הרשימה נועדה לעזור ואינה מחייבת. ייתכן שקיימות פונקציות או סוגי אובייקטים שלא נמצאים ברשימה אבל כן צריך להשתמש בהם על מנת לפתור את המבחן