



**מבחן מסכם מועד א – הנדסת תוכנה 094219**

**מרצה: ד"ר אחמד ג'בארה**

**מתרגל אחראי: מר דביר טויטו**

**תאריך: 15.02.2022**

**תעודת זהות:**

\_\_\_\_\_

**חדר בחינה:**

\_\_\_\_\_

הנחיות לנבחנת:

- ☐ מומלץ לקרוא את כל המבחן לפני תחילת הפתרון.
- ☐ חלק מן השאלות מכילות בסופן הערות והכוונות. מומלץ לא להתחיל לפתור שאלה לפני שקראת את כל סעיפיה.
- ☐ משך המבחן: 3 שעות.
- ☐ במבחן יש 7 שאלות. עלייך לענות על כולן.
- ☐ משקל כל שאלה רשום לידה.
- ☐ אין חומר עזר אשר מותר לשימוש.
- ☐ יש לפתור את המבחן בעט כחול או שחור. ניתן להשתמש בעפרון לצורך סרטוט התרשימים בלבד.
- ☐ יש לענות כל השאלות במחברת הבחינה פרט לשאלת ה-UML עליה יש לענות על גבי טופס הבחינה.
- ☐ יש להקפיד על כתב יד ברור.
- ☐ בשאלות בהן נדרש לסרטט תרשים, יש להקפיד על בהירותו ועל כיווני החיצים.
- ☐ חלק מקטעי הקוד עושים שימוש במחלקות סטנדרטיות של Java. ניתן להניח כי בוצע ייבוא כנדרש.
- ☐ ההנחיות במבחן מנוסחות בלשון נקבה מטעמי נוחות בלבד, אך מתייחסות לנבחנים ולנבחנות כאחד.

**בהצלחה!**



**שאלה 1 (8 נקודות)**

בהרצאה דנו בבעיית מגדלי האנוי: נתונים שלושה עמודים A, B, C, כאשר על עמוד A ישנן דיסקיות אשר מסודרות אחת על גבי השנייה מן הדיסקית הגדולה ביותר ועד הקטנה ביותר. בבעיה, עלינו להעביר את כל הדיסקיות מעמוד A לעמוד C בעזרת עמוד B ולבנות את מגדל הדיסקיות מחדש, כאשר לא ניתן להניח דיסקית כלשהי על גבי דיסקית אשר קטנה ממנה ובכל צעד ניתן להעביר דיסקית אחת בדיוק.

שאלה זו עוסקת בגרסה של בעיית מגדלי האנוי אשר מכונה Adjacent Pegs (עמודים סמוכים). בגרסה זו מתווסף לבעיה אילוץ נוסף: מותר להעביר דיסקית לעמוד סמוך בלבד. לדוגמה, ניתן להעביר דיסקית מעמוד B לעמוד C או לעמוד A בתנאי שהעברה זו עומדת בתנאי הבעיה המקורית, אך לא ניתן להעביר דיסקית ישירות מעמוד A לעמוד C או ההפך.

כתבי ב-Java פעולה סטטית בעלת הכותרת הבאה:

```
public static void adjacentHanoi(int n, String A, String B, String C)
```

הפעולה מקבלת את מספר הדיסקיות ואת שלושת העמודים ומדפיסה למסך את רצף ההוראות המינימלי אשר נדרש על מנת להעביר את כל הדיסקיות מעמוד A לעמוד C בעזרת עמוד B.



שאלה 2 (11 נקודות)

נתונה המחלקה Node והפעולה הסטטית hasSomething:

```
class Node {
    private int info;
    private Node next;

    public Node(int info, Node next) {
        this.info = info;
        this.next = next;
    }

    public Node getNext() {
        return next;
    }
}

public static boolean hasSomething(Node list) {
    Node first = list;
    Node second = list;

    while (first != null && second != null && second.getNext() != null) {
        first = first.getNext();
        second = second.getNext().getNext();
        if (first == second) {
            return true;
        }
    }
    return false;
}
```

סעיף א (6 נקודות)

תני דוגמה לרשימה מקושרת עבורה הפעולה hasSomething תחזיר שקר (false) ודוגמה לרשימה מקושרת עבורה הפעולה hasSomething תחזיר אמת (true). על כל אחת משתי הרשימות להכיל לפחות שלושה איברים.

סעיף ב (5 נקודות)

הסבירי במילים מה הפעולה hasSomething מבצעת עבור רשימה מקושרת כלשהי.



### שאלה 3 (28 נקודות)

לפנייך קוד התחלתי של המחלקה Set אשר מייצגת קבוצה של מספרים שלמים. כזכור, בקבוצה כל איבר מופיע לכל היותר פעם אחת, ואין חשיבות לסדר בין האיברים.

```
public class Set {  
    private int[] values;  
    private int index; // Next index for inserting an element  
  
    public Set(int maxSize) {  
        values = new int[maxSize];  
        index = 0;  
    }  
}
```

### סעיף א (7 נקודות)

הגדירי במחלקה Set פעולה בשם addElement אשר מקבלת מספר שלם ומוסיפה אותו לקבוצה במידה וניתן. אם האיבר מופיע כבר בקבוצה תיזרק חריגה בלתי מסומנת בשם ElementAlreadyExists. במידה והקבוצה הגיעה לגודלה המקסימלי (לפני הוספת האיבר) תיזרק חריגה בלתי מסומנת בשם SetOverflowException. שימי לב: חריגות אלו אינן מוגדרות ב-Java ועלייך להגדירן באופן עצמאי כך שניתן יהיה לתפוס את שתיהן על ידי שימוש בבלוק catch יחיד.

### סעיף ב (6 נקודות)

ממשי במחלקה Set את הממשק Cloneable ודרסי את הפעולה clone כדי לאפשר העתקה עמוקה של עצמים מן המחלקה. עלייך לדרוס את הפעולה כך שהפעולה הבאה תעבור הידור:

```
public static void doSomething() {  
    Set mySet = new Set(17);  
    Set myClonedSet = mySet.clone();  
}
```

שימי לב כי לא נדרשת המרה בשורה השנייה, וכי שורות הקוד אינן מוקפות בבלוק try.

### סעיף ג (5 נקודות)

דרסי במחלקה Set את הפעולה equals כך ששתי קבוצות יהיו שוות זו לזו אם ורק אם הן מכילות את אותם האיברים, ללא חשיבות לסדר. יש להקפיד על שאר הדרישות אשר קיימות עבור יחס השוויון ופעולת ה-equals.

הערה: בסעיף זה אין צורך לדרוס את הפעולה hashCode.



סעיף ד (10 נקודות)

נרצה לאפשר מעבר על איברי הקבוצה על ידי לולאת `foreach`. לשם כך נגדיר שני סוגי סריקה אפשריים: מעבר על איברי הקבוצה לפי הסדר בו הם מופיעים במערך, ומעבר על האיברים בסדר הפוך מן הסדר בו הם מופיעים במערך.

לשם כך, תחילה עלייך להגדיר במחלקה `Set` פעולה בשם `setScanningType` אשר מקבלת מספר שלם. המספר אשר מועבר לפעולה קובע את סוג הסריקה הנוכחי של האיברים עד לקריאה הבאה לפעולה, כאשר הערך 0 מציין סריקה לפי הסדר והערך 1 מציין סריקה בסדר הפוך. ניתן להניח שלא יועבר לפעולה פרמטר שאינו תקין.

לאחר מכן, עלייך ליצור מחלקה בשם `SetIterator` אשר מממשת את הממשק `Iterator<Integer>`. יש לממש במחלקה זו את הפעולות `hasNext` ו-`next` בהתאם לסוג הסריקה.

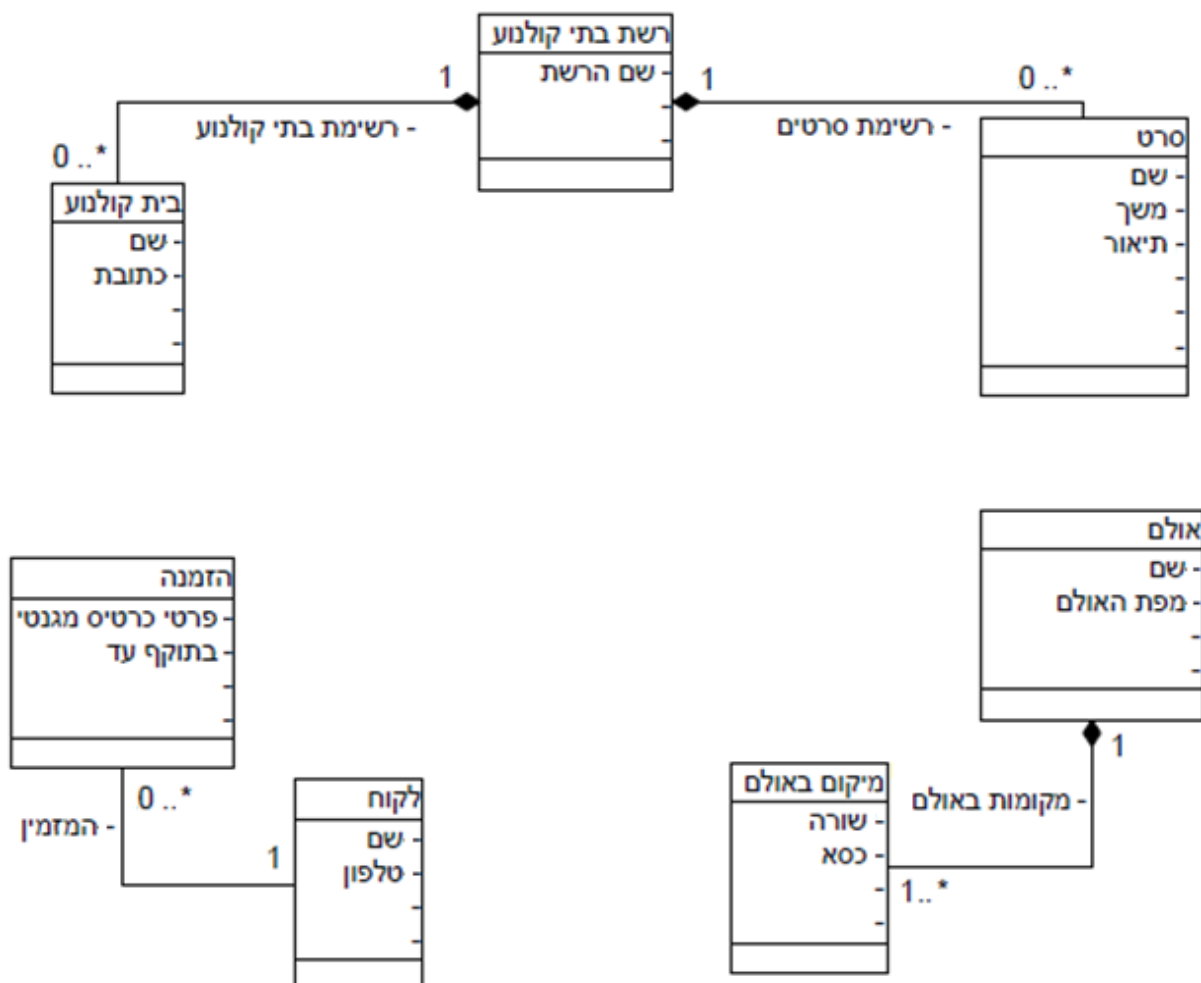
לבסוף, עלייך לממש במחלקה `Set` את הממשק `Iterable<Integer>` ואת הפעולה `iterator` אשר מוגדרת בממשק.



**שאלה 4 (15 נקודות)**

בבעלותה של רשת בתי הקולנוע Technion Planet בתי קולנוע בכל רחבי הארץ. הרשת מקימה מערכת ממוחשבת דרכה ימכרו כרטיסים לכל בתי הקולנוע השייכים לרשת. בכל בית קולנוע יש מספר אולמות קולנוע. הסרטים המוקרנים יכולים להיות שונים מבית קולנוע אחד למשנהו, אך לפעמים אותו סרט מוקרן בבתי קולנוע שונים. עלות צפייה בסרט הנה קבועה בכל בתי הקולנוע של הרשת ולכל הסרטים. לרשת יש מנויים המקבלים הנחה קבועה של X אחוזים על צפייה בכל אחד מהסרטים.

להלן מידול התחלתי של המערכת המוצגת על ידי תרשים UML:





עלייך להוסיף לתרשים **אך ורק** את המידע הבא באמצעות יחסים, ריבויים (Multiplicity), תפקידים (Roles), מחלקות ומאפיינים:

- באילו בתי קולנוע מוקרן סרט מסוים, באיזה אולם בבית הקולנוע המסוים הסרט מוקרן, באילו תאריכים ובאילו שעות.
- מי הזמין כרטיס(ים), לאיזו הקרנה של הסרט הוזמן הכרטיס והאם מדובר במנוי או בלקוח מזדמן (לקוח שאינו מנוי).
- עלות צפיה בסרט ללקוח מזדמן ולמנוי.
- מה שמו ומספרו הסידורי של כל מנוי.

#### שימי לב:

- הוספת מחלקה / מאפיין / יחס מיותרים או לא נכונים עשויה להוריד נקודות.
- לא בהכרח נדרש להוסיף יחס הורשה לשרטוט.
- אין חובה לרשום מאפיינים אשר מייצגים יחס אך הוספתם לא תוריד ניקוד.
- אל תמהרי להוסיף מחלקות. עדיף להוסיף יחס או מאפיין במידה והם עונים על הדרישה.
- יחס יכול להיות יחס פשוט (Association), הכלה חזקה (Composition), הכלה חלשה (Aggregation) והורשה (Inheritance).
- התבליטים הריקים אשר מופיעים בכל מחלקה אינם מעידים על כך שעלייך להוסיף תכונות למחלקה וכן הם אינם מעידים על מספר התכונות של המחלקה.
- יש לענות על שאלה זאת **על גבי טופס הבחינה**.
- לנוחיותך, בדף האחרון של הבחינה מופיע עותק נוסף של התרשים. ניתן להיעזר בעותק זה לצורך ביצוע סקיצות של התרשים. על התשובה הסופית להופיע בשאלה זו ולא על גבי העותק.



### שאלה 5 (18 נקודות)

בשווקים רבים מוכרים יכולים להציע מוצר למכירה על ידי ביצוע מכרז. כל מכרז מורכב ממוכר יחיד וממספר לא מוגבל של משתתפים. בכל מכרז המשתתפים מציעים מחיר בעבור המוצר באופן שאינו גלוי לשאר המשתתפים. בעבור המוכר שומרים את שמו, את מספר הטלפון שלו ואת הפריט אשר הוא מציע למכירה. בעבור כל משתתף במכרז שומרים את שמו, את מספר הטלפון שלו ואת הכיד (Bid) שלו (הסכום אשר הוא הציע במכרז).

ישנם שני סוגים של מכרזים – מכרז מחיר ראשון ומכרז מחיר שני.

במכרז מחיר ראשון המשתתף אשר הציע את המחיר הגבוה ביותר מקבל את המוצר ומשלם את המחיר שהציע.

במכרז מחיר שני המשתתף אשר הציע את המחיר הגבוה ביותר מקבל את המוצר אך משלם את הסכום השני בגובהו מבין הסכומים אשר הוצעו.

### סעיף א (6 נקודות)

כתבי מחלקות אשר מייצגות מוכר במכרז ומשתתף במכרז. במחלקות יש לכלול תכונות מתאימות ובנאים.

### סעיף ב (6 נקודות)

כתבי מחלקה מופשטת בשם Auction אשר מייצגת מכרז כלשהו. במחלקה יש לכלול תכונות מתאימות ואת הפעולות הבאות:

- בנאי אשר מקבל עצם המייצג את המוכר במכרז.
- פעולה אשר מוסיפה משתתף למכרז.
- פעולה בשם getWinner אשר מחזירה את המשתתף שניצח במכרז. ניתן להניח כי יש רק משתתף אחד כזה.
- פעולה מופשטת בשם getWinningBid אשר מחזירה את הסכום אותו המנצח במכרז צריך לשלם. ניתן להניח כי בעת הקריאה לפעולה זו יש במכרז לפחות שני משתתפים.

### סעיף ג (6 נקודות)

כתבי מחלקות בשם FirstPriceAuction ו-SecondPriceAuction אשר מייצגות מכרז מחיר ראשון ומכרז מחיר שני בהתאמה. על המחלקות לרשת מן המחלקה Auction ולממש את הפעולה המופשטת.

הערות לכל סעיפי השאלה:

- יש לתכנן את המחלקות על פי עקרונות תכנות מונחה עצמים נכונים.
- ניתן להשתמש בפעולות get ו-set לתכונות המחלקות מבלי לממש אותן.
- אין צורך להקפיד על ביצוע import.





**שאלה 6 (15 נקודות)**

לפניך קטע קוד אשר מכיל הגדרה של המחלקה A ומספר שורות קוד נוספות. מיד לאחריו מופיע קוד המקור של הפעולות put ו-get של המחלקה הגנרית HashMap.

```
1 class A {
2     private int a;
3     public A(int a){
4         this.a = a;
5     }
6     public int hashCode(){
7         return a;
8     }
9 }
10 HashMap<A, Integer> m = new HashMap<A, Integer>();
11 A obj;
12 System.out.println(m.put(obj = new A(5), 1));
13 System.out.println(m.put(obj = new A(21), 2));
14 System.out.println(m.put(obj = new A(5), 3));
15 System.out.println(m.put(obj, 4));
16 System.out.println(m.get(new A(5)));
```

קוד המקור:

```
1 public class HashMap<K,V> extends AbstractMap<K,V> implements Map<K,V>, Cloneable, Serializable {
2     static final int TREEIFY_THRESHOLD = 8;
3     Node<K,V>[] table;
4     public V put(K key, V value) {
5         return putVal(hash(key), key, value, false, true);
6     }
7     final V putVal(int hash, K key, V value, boolean onlyIfAbsent, boolean evict) {
8         Node<K,V>[] tab; Node<K,V> p; int n, i;
9         if ((tab = table) == null || (n = tab.length) == 0)
10             n = (tab = resize()).length;
11         if ((p = tab[i = (n - 1) & hash]) == null)
12             tab[i] = new Node<>(hash, key, value, null);
13         else {
14             Node<K,V> e; K k;
15             if (p.hash == hash && ((k = p.key) == key || (key != null && key.equals(k)))) e = p;
16             else if (p instanceof TreeNode) e = ((TreeNode<K,V>)p).putTreeVal(this, tab, hash, key, value);
17             else {
18                 for (int binCount = 0; ; ++binCount) {
19                     if ((e = p.next) == null) {
20                         p.next = new Node<>(hash, key, value, next);
21                         if (binCount >= TREEIFY_THRESHOLD - 1) treeifyBin(tab, hash);
22                         break;
23                     }
24                     if (e.hash == hash && ((k = e.key) == key || (key != null && key.equals(k))))
25                         break;
26                     p = e;
27                 }
28             }
29             if (e != null) {
30                 V oldValue = e.value;
31                 if (!onlyIfAbsent || oldValue == null) e.value = value;
32                 return oldValue;
33             }
34         }
35         ++modCount;
36         if (++size > threshold) resize();
37         return null;
38     }
```



```
39 public V get(Object key) {
40     Node<K,V> e;
41     return (e = getNode(hash(key), key)) == null ? null : e.value;
42 }
43 final Node<K,V> getNode(int hash, Object key) {
44     Node<K,V>[] tab; Node<K,V> first, e; int n; K k;
45     if ((tab = table) != null && (n = tab.length) > 0 &&
46         (first = tab[(n - 1) & hash]) != null) {
47         if (first.hash == hash &&
48             ((k = first.key) == key || (key != null && key.equals(k))))
49             return first;
50         if ((e = first.next) != null) {
51             if (first instanceof TreeNode)
52                 return ((TreeNode<K,V>)first).getTreeNode(hash, key);
53             do {
54                 if (e.hash == hash && ((k = e.key) == key || (key != null && key.equals(k))))
55                     return e;
56             } while ((e = e.next) != null);
57         }
58     }
59     return null;
60 }
61 static class Node<K,V> implements Map.Entry<K,V> {
62     final int hash;
63     final K key;
64     V value;
65     Node<K,V> next;
66     Node(int hash, K key, V value, Node<K,V> next) {
67         this.hash = hash; this.key = key;
68         this.value = value; this.next = next;
69     }
70 }
71 }
```

#### סעיף א (9 נקודות)

ציירי תרשים עצמים עבור התכונה table של המחלקה HashMap לאחר ביצוע שורות 10-15 בקטע הקוד הנתון. עליך לכלול בתרשים את ערכי התכונות של כל העצמים. במידה והתכונות מתעדכנות במהלך ביצוע קטע הקוד, יש למחוק את הערך הישן של התכונה על ידי קו (באופן שעדיין יהיה ניתן לקרוא אותו) ולרשום ליד את הערך החדש. בנוסף, עליך להסביר את התרשים ולציין באופן מפורש את מספרי השורות של פעולות ה-put וה-get אשר התבצעו והובילו לתרשים.

#### סעיף ב (6 נקודות)

מה יהיה הפלט לאחר ביצוע קטע הקוד הנתון?

יש להסביר את תשובתך ולציין באופן מפורש את מספרי השורות של פעולות ה-put וה-get אשר התבצעו והובילו לפלט זה.

תזכורות:

- פעולת ה-hash של המחלקה HashMap משתמשת בפעולה hashCode של המפתח אותו מוסיפים.
- בקריאה הראשונה לפעולה resize של המחלקה HashMap חוזר הערך 16.
- לכל שני מספרים שלמים  $m, n$  כאשר  $n$  הוא חזקה שלמה של 2, הפעולה  $m \& (n - 1)$  שקולה לפעולה  $m \% n$  כאשר  $m \% n$  הוא אופרטור המודולו.



**שאלה 7 (5 נקודות)**

אחת מתבניות העיצוב אשר נלמדו בקורס הינה סינגלטון (Singleton). בתרגול ראינו כי ניתן לממש את תבנית העיצוב בעזרת שתי גישות שונות: אתחול עצל (Lazy Initialization) ואתחול להוט (Eager Initialization). בתרגול ראינו כי המימוש העצל אינו בטוח לעבודה עם מספר תהליכונים (Thread Safe) ומטרתה של השאלה היא לפתור את בעיה זו.

**סעיף א (2 נקודות)**

תארי במילים דרך בה ניתן להפוך את שיטת האתחול העצל לבטוחה לעבודה עם מספר תהליכונים.

**סעיף ב (3 נקודות)**

ממשי את הצעתך מסעיף ב'. לצורך כך, עלייך ליצור מחלקה בשם Singleton אשר מממשת את תבנית העיצוב סינגלטון ונעזרת באתחול עצל ובהצעתך מן הסעיף הקודם.



עותק נוסף של תרשימים המחלקות:

