

חומרי עזר שהוכנו ע"י משתתפי קורס מורים מובילים תשע"א
ניתן להשתמש בחומרים לצורך הוראה בלבד. אסור לפרסם את החומרים או לעשות בהם שימוש מסחרי כלשהו ללא קבלת אישור מראש מצוות הפיתוח

חומרים שהוכנו על-ידי משתתפי קורס מורים מובילים תשע"א

ניתן להשתמש בחומרים לצורך הוראה בלבד.

לא ניתן לפרסם את החומרים או לעשות בהם כל שימוש מסחרי

ללא קבלת אישור מראש מצוות הפיתוח

אוסף שאלות שמימושן מפה

כתיבה ועריכה:

אביטל גרינולד

דורית כהן

מיכל עמירה

ניתוח, חשיבה ורעיונות: גרינולד אביטל (EVI), כהן דורית, עמירה מיכל
כתיבת השאלות ופתרון בשפת ג'אווה מימוש ע"י EVI ומיכל
פתרונות בסי שארפ, מימוש ע"י דורית כהן

לפניכם מספר סוגים של שאלות אשר משתמשות במפה לפתרון.

מספר	סוג	פירוט והסבר	מטרה ומיומנות נדרשת
1	שאלת מעקב	נתון קטע קוד. - יש לעקוב אחר קטע הקוד ולרשום את הפלט. - הוספת שורות קוד - מימוש פעולה נוספת	לבדוק הבנה בסיסית של הוראות המשתמשות במפה. הן על-ידי מעקב אחר הקוד והן ע"י הוספת שורות לקוד.
2	בניית מחלקה על-פי ממשק	ישנו תיאור קצר לבעיה. למעשה בחרנו שאלה שנהגנו לפתור בעזרת מערך מונים וכאן בחרנו להשתמש במפה. - על פי הייצוג והממשק יש לממש את המחלקה. - יש לזמן את פעולות הממשק	כתיבת מחלקה על-פי ממשק וכאשר נתון הייצוג שלה. לבדוק שימוש בפעולות הממשק של המחלקה.
3	בחירת ייצוג פיתוח	- ייצוג מחלקה על פי ממשק. - שימוש בפעולות הממשק. - מימוש פעולת מחלקה - ניתוח יעילות.	הבנת השימוש במפה כאשר הערך הוא עצמו אוסף (רשימה)
4	ייצוג לאוסף מורכב	הצגת אוסף מורכב ודרישה לכתיבת ייצוגים שונים והשוואה ביניהם.	הבנה מעמיקה של טיפול באוספי נתונים.

ממשק המחלקה מפה <Map>:

המחלקה מגדירה אוסף דינמי הממפה מפתחות וערכים. המפתחות במפה יהיו מטיפוס מחרוזת בעוד הערכים יהיו גנריים.

Map()	הפעולה בונה מפה ריקה
V getValue (String key)	הפעולה מחזירה את הערך הקשור למפתח key. הפעולה מחזירה null אם המפתח אינו קיים במפה
void insert (String key, V value)	הפעולה מוסיפה למפה הנוכחית את המפתח key ואת הערך value הקשור אליו. אם המפתח key קיים במפה, הפעולה מעדכנת את הערך הקשור אליו בערך value שהתקבל
V remove (String key)	הפעולה מוציאה מהמפה הנוכחית את המפתח key ואת הערך הקשור אליו. הפעולה מחזירה את הערך הקשור למפתח שהוצא מהמפה. אם המפתח אינו קיים במפה היא מחזירה null
String[] getAllKeys()	הפעולה מחזירה את אוסף המפתחות שקיימים במפה הנוכחית ממוין בסדר אלפביתי עולה. אם המפה ריקה, יוחזר מערך בגודל אפס
String toString()	הפעולה מחזירה מחרוזת המתארת את המפה כך : [key1:value1, key2:value2, key3:value3,...]

שאלה 1 : שאלת מעקב על מפה

נתון קטע קוד אשר מטפל בתלמידים וציוניהם.

```
Scanner in = new Scanner(System.in);
Map<Integer> grades = new Map<Integer>();
Integer grdValue;
String name;
int grade;
System.out.println("Enter name");
name = in.next();
while ( !name.equals("end"))
{
    System.out.println("Enter grade");
    grade = in.nextInt();
    grdValue = grades.getValue(name);
    if (grdValue!=null)
    {
        if ( grade > grdValue)
            grades.insert(name, grade); // update grade
    }
    else // (grdValue==null)
        grades.insert(name, grade);
    System.out.println("Enter name");
    name = in.next();
}
System.out.println(grades);
```

א- עקוב אחר קטע הקוד בעזרת תרשים או טבלת מעקב ורשום את הפלט עבור הקלט משמאל לימין:
dani , 80 , reut , 70 , dani , 68 , avi , 90 , reut , 72 , end

ב- הוסף לכל היותר שתי שורות לקוד, כדי להציג את מספר התלמידים באוסף.
חובה להשתמש בפעולות הממשק של מפה.

ג- הוסף את שורות הקוד:

```
String[] arr = grades.getAllKeys();
String sName = arr[0];
Integer g = grades.remove(sName);
System.out.println("grade is: " + g);
System.out.println(grades);
```

1- צייר את תוכן המערך arr
2- רשום את הפלט.

ד- לפניך כותרת פעולה אשר מקבלת מפה של שלמים ומחזירה את הערך הגבוה ביותר
public static int getMaxGrade(Map<Integer> m)
השלם את גוף הפעולה.

פתרון שאלה מספר 1

א- [avi 90 , dani 80 , reut 72]

ב- `String[] arr = grades.getAllKeys();`
`System.out.println("there are " + arr.length + " students");`

ג- `arr [avi , dani, reut]`

grade is: 90

[dani 80 , reut 72]

ד `public static int getMaxGrade(Map<Integer> m)`
`{`
`Integer maxGrade=0;`
`int grade;`
`String[] arr = m.getAllKeys();`
`for (int i = 0; i < arr.length; i++)`
`{`
`grade = m.getValue(arr[i]);`
`if (grade>maxGrade)`
`maxGrade = grade;`
`}`
`return maxGrade;`
`}`

שאלה 2: שכיחות תווי המקלדת בטקסט

לבניית סידור התווים על מקשי המקלדת, יש לבדוק את שכיחות השימוש בכל אחד מהם.

כדי לסייע במציאת שכיחות תווי המקלדת, נגדיר טיפוס `SignCounter`,
אשר מכיל תכונה אחת בלבד: `Map<Integer> counter`

המפתחות יהיו התווים שהוקלדו והערכים יהיו מספר הפעמים שהופיע כל תו.
הערה: מאחר והמפתח הוא מטיפוס מחרוזת, נמיר את התו למחרוזת.

לפניך ממשק המחלקה `SignCounter`

תיאור הפעולה	הפעולה
הפעולה בונה <code>SignCounter</code> ריק	<code>SignCounter ()</code>
הפעולה מקבלת תו שהוקלד ומוסיפה אותו ל <code>SignCounter</code>	<code>void addSign(String sign)</code>
הפעולה מחזירה את אוסף התווים והשכיחות שלהם	<code>Map<Integer> getCounter()</code>
הפעולה מחזירה רשימה של התווים ששכיחותם היא <code>f</code>	<code>List<Character> signXtimes(int f)</code>
הפעולה מחזירה את השכיחות הגבוהה ביותר	<code>int maxFreq()</code>
הפעולה מחזירה מחרוזת המתארת את <code>SignCounter</code> כך: <code>Sign 1 → frequency1</code> <code>Sign2 → frequency2</code> . . התווים יופיעו בסדר עולה על פי הקוד ה <code>asci</code>	<code>String toString()</code>

א- ממש את המחלקה `SignCounter` על פי הממשק.

- ב- כתוב פעולה ראשית אשר מבצעת את הפעולות הבאות:
- קולטת תווים מהמקלדת ומציגה את השכיחות של כל תו שהוקלד.
- מציגה את השכיחות המקסימלית.
- מציגה את כל התווים ששכיחותם היא המקסימלית.

ג- כתוב פעולה חיצונית אשר מקבלת משתנה מטיפוס `SignCounter` ומציגה כפלט את כל האותיות שלא הופיעו מבין האותיות 'a' עד 'z' כולל.

הסבר הרעיון לשאלה 2

מטרה השאלה: מציאת שכיחויות של תווים שהוקלדו
רעיון הפתרון: שימוש במפה.

המפתח – התו מומר למחרזת
הערך - מספר שלם המהווה שכיחות

לקחנו תרגיל שבד"כ פתרנו בעזרת מערך מונים.
יתרון הפתרון בעזרת מפה על פני שימוש במערך מונים:

- נשמרים אך ורק התווים שהוקלדו – חסכון בזיכרון
- אין צורך לדעת מראש מי התווים הנבדקים
- לתלמידים יש קושי להבין את הקשר בין הערך הנקלט/הנבדק לבין האינדקס של המערך.
- במערך יש להתאים בין אינדקס המערך לערך הנבדק
- פתרון בעזרת מפה עוקף את הבעיות שצוינו
- הפתרון בעזרת מפה מכליל אוסף פתרונות.
- המימוש הרבה יותר פשוט.

דרך הפתרון:

במקום מערך השתמשנו במפה וניצלנו את העובדה שהפעולה getAllKeys מחזירה את המפתחות ממוינים לפי סדר עולה של התווים על פי קוד ה- ascii שלהם. כך שפלט לשכיחויות הוא למעשה הפעולה toString .

פתרון שאלה 2 - שכיחות תוי הקלט – פתרון בעזרת מפה

- א -

```
// מחלקה אשר שומרת כמה פעמים הופיע כל תו שהוקלד במקלדת
package SignCounter;
```

```
import unit4.collectionsLib.List;
import unit4.collectionsLib.Map;
import unit4.collectionsLib.Node;
```

```
/**
```

```
 * @author EVI + Michal
```

```
 * @version 6/6/11
```

```
 */
```

```
public class SignCounter
```

```
{
```

```
    private Map<Integer> counter;    // Map to save appearance of sign
```

```
/**
```

```
 * create SignCounter with no letters
```

```
 */
```

```
public SignCounter()
```

```
{
```

```
    this.counter = new Map<Integer>();
```

```
}
```

```
/**
```

```
 * @return counter
```

```
 */
```

חומרי עזר שהוכנו ע"י משתתפי קורס מורים מובילים תשע"א
ניתן להשתמש בחומרים לצורך הוראה בלבד. אסור לפרסם את החומרים או לעשות בהם שימוש מסחרי כלשהו ללא קבלת אישור מראש מצוות הפיתוח

```
public Map<Integer> getCounter()
{
    return this.counter;
}
/**
 * add 1 to sign appearance or makes it 1 if did not appear before
 * @param sign: String ( a letter form the sign )
 */
public void addsign(String sign)
{
    if (this.counter.getValue(sign) == null)
        this.counter.insert(sign, 1);
    else
        { // sign exists
            this.counter.insert(sign, this.counter.getValue(sign)+1);
        }
}
/**
 * @return String representing sign
 */
public String toString()
{
    String str="";
    String[] allsign = this.counter.getAllKeys();
    for (int i = 0; i < allsign.length; i++)
    {
        str += String.format("%3s --> %3s", allsign[i],
this.counter.getValue(allsign[i])+"\n");
    }
    return str;
}
/**
 * @param f: int , frequency
 * @return List<Character> with signs appears f times
 */
public List<Character> signXtimes(int f)
{
    List<Character> lstSigns = new List<Character>();
    Node<Character> pos = lstSigns.getFirst();
    String[] allSigns = this.counter.getAllKeys();
    for (int i = 0; i < allSigns.length; i++)
    {
        if ( this.counter.getValue(allSigns[i]) == f )
            pos = lstSigns.insert(pos, allSigns[i].charAt(0));
    }
    return lstSigns;
}
}
```

```
public int maxFreq()
{
    int max = 0;
    Integer f;
    String[] allSigns = this.counter.getAllKeys();
    for (int i = 0; i < allSigns.length; i++)
    {
        f = this.counter.getValue( allSigns[i] );
        if ( f>max )
            max = f;
    }
    return max;
}
} //=====
```

- ב -

```
/*
 * Test SignCounter : counts how many letters sign
 */
package SignCounter;
import java.util.Scanner;
/**
 * @version 6/6/11
 * @author EVI + Michal
 */
public class Main
{
    static Scanner in = new Scanner(System.in);

    //-----      MAIN      -----
    public static void main(String[] args)
    {
        SignCounter countersign = new SignCounter();
        // get letters
        in.useDelimiter("");
        System.out.println("Type characters");
        char tav;
        String sTav;
        tav = in.next().charAt(0);
        while (tav != 10)
        { // read characters till Enter is typed
            sTav = ""+tav;
            countersign.addsign(sTav);
            tav = in.next().charAt(0);
        }
        System.out.println();
        // output
        System.out.println("--- frequencies ---");
        System.out.println(countersign);
        // find common sign
        int max = countersign.maxFreuq();
    }
}
```


חומרי עזר שהוכנו ע"י משתתפי קורס מורים מובילים תשע"א
ניתן להשתמש בחומרים לצורך הוראה בלבד. אסור לפרסם את החומרים או לעשות בהם שימוש מסחרי כלשהו ללא קבלת אישור מראש מצוות הפיתוח

```
System.out.println("the signs that appear the most");  
System.out.println(countersign.signXtimes(max));  
}  
}
```

- ג -

```
/**  
 * prints all letters between a to z that not appeared  
 * @param sc: SignCounter  
 */  
public static void notAppear(SignCounter sc)  
{  
    System.out.println("letters that did not appear");  
    for (char c = 'a'; c<='z'; c++)  
    {  
        if ( sc.getCount().getValue(""+c) ==null)  
            System.out.print(c + " ,");  
    }  
    System.out.println();  
}
```

שאלה 3: מספרי שורות של מילה בטקסט

לשם חיפוש מידע בטקסט, מעוניינים לשמור עבור כל מילה את מספרי השורות בהן היא מופיעה.

שאלות אשר עשויות לעניין אותנו:

- 1) מהן מספרי השורות בהן מופיעה מילה מסוימת?
- 2) איזה מילים מופיעות בשורה מסוימת?
- 3) בכמה שורות מופיעה מילה מסוימת?
- 4) מה מספר השורות בטקסט?
- 5) עבור כל מילה מה השורה הראשונה בה היא מופיעה?
- 6) איזה מילה מופיעה במספר שורות מקסימלי?
- 7) בהינתן שתי מילים, מהם מספרי השורות בהן מופיעות שתי המילים גם יחד?
- 8) האם כל המילים מופיעות במספר זהה של שורות?

לסיוע במתן תשובות לשאלות הנ"ל נבנה מחלקה **מילים-בשורות**, `WordLines`
לפניך ממשק חלקי של המחלקה **מילים-בשורות**, `WordLines`

תיאור הפעולה	הפעולה
יצירת מילים-בשורות ריק	<code>WordLines()</code>
הפעולה מוסיפה מילה ומספר השורה בה היא מופיעה. אם המילה כבר קיימת אז יתווסף מספר השורה לרשימת מספרי השורות בהם מופיעה המילה. אם המילה כבר הופיעה באותה שורה, לא יתבצע שינוי. מספרי השורות ממוינים בסדר עולה. הנחה: מספר שורה חוקי (מספר טבעי)	<code>void addWord(String word, int lineNo)</code>
הפעולה מחזירה אוסף כל המילים שהופיעו ב מילים-בשורות ממוינות בסדר אלפא-ביתי עולה.	<code>String[] getAllWords()</code>
הפעולה מחזירה את מספרי השורות עבור מילה <code>word</code> . אם המילה לא קיימת הפעולה תחזיר <code>null</code>	<code>List<Integer> getLines(String word)</code>
הפעולה מחזירה את כל המילים בשורה <code>line</code> . אם אין אף מילה הפעולה תחזיר <code>null</code> .	<code>List<String> getWordsInLine(int line)</code>
הפעולה מחזירה את מספר השורות ב מילים-בשורות	<code>int getNumOfLines()</code>
הפעולה מחזירה את מספר השורה הראשונה בה מופיעה המילה <code>word</code> , <code>0</code> – אם לא מופיעה.	<code>int getFirstLine(String word)</code>
הפעולה מחזירה את מספר השורות בהן מופיעה המילה <code>word</code>	<code>int getNumOfLines(String word)</code>
הפעולה מחזירה רשימת מילים אשר הופיעו במספר שורות הרב ביותר	<code>List<String> wordsInMaximumLines()</code>
הפעולה מחזירה מחרוזת המתארת את מילים-בשורות כך: [מספרי השורות] → מילה	<code>String toString()</code>

(1) הצע ייצוג למחלקה **WordLines**

- (2) נתון עצם מטיפוס **WordLines** בשם **wl** לא ריק.
כתוב קטע קוד אשר יבצע את המשימות הבאות:
א- יציג כפלט את מספר המילים באוסף **wl** .
ב- יציג את המילים באוסף ממוינות בסדר אלפא-ביתי עולה.
ג- עבור כל מילה באוסף **wl** יציג את מספרי השורות בהן היא הופיעה.
ד- יבדוק אם המילה "ball" קיימת , אם כן יציג בכמה שורות היא מופיעה אם לא יוסיף אותה בשורה האחרונה.

(3) ממש את הפעולה **wordsInMaximumLines**

- (4) נתח את יעילות הפעולה שמימשת בשאלה (3).
הנח שיעילות הפעולה הבונה של המחלקה מפה היא $O(1)$ וכל שאר הפעולות שלה הן $O(n)$.
(5) כתוב פעולה חיצונית אשר מקבלת שתי מילים ומחזירה רשימה של מספרים שלמים שמהווים מספרי השורות בהן מופיעות שתי המילים גם יחד. מספרי השורות ממוינים בסדר עולה.

הערות:

- במימוש הפעולות ניתן:
- להוסיף פעולות עזר פרטיות.
- ניתן להשתמש בכל אחת מהפעולות של מפה, רשימה, חוליה או כל פעולה שמופיעה בממשק המחלקה **WordLines** מבלי לממשן.

הצעה לפתרון שאלה 3

(1) ייצוג אפשרי הוא מפה של רשימות של מספרים שלמים, כאשר:
המפתח: מילה בטקסט.

הערך: רשימה ממוינת של מספרים שלמים שמהווים מספרי השורות בהם מופיעה כל מילה.

```
private Map<List<Integer>> lineNumbers;
```

ניתן לחשוב על ייצוגים נוספים, למשל:

הערך – קבוצה (set) של מספרים שלמים מספרי השורות בהן מופיעה המילה.

(2)

```
String[] words = wl.getAllWords();
int numOfWords = words.length;
System.out.println("1) There are " + numOfWords + " words");
System.out.println("2) The words");
for (int k=0; k<words.length; k++)
    System.out.print(words[k] + " ,");
System.out.println();
System.out.println("3) word and lines");
System.out.println(wl);
int numberOfLines = wl.getNumOfLines();
System.out.println("4) check if ball exists");
List<Integer> linesBall = wl.getLines("ball");
if (linesBall!=null)
    System.out.println("ball exists in: " + linesBall);
else
    { //ball does not exists, add it to last line
        int numOfLines = wl.getNumOfLines();
        wl.addWord("ball", numOfLines);
    }
```

(3) הפעולה מחזירה את אוסף המילים שמופיעות במספר שורות מירבי

```
public List<String> wordsInMaximumLines()
{
    String[] allWords = this.wordAndLines.getAllKeys();
    List<Integer> lines;
    int max = this.maxLinesPerWord();
    int len;
    List<String> words = new List<String>();
    Node<String> pos = words.getFirst();
    for (int k=0; k<allWords.length; k++)
    {
        lines = this.wordAndLines.getValue(allWords[k]);
        len = this.lengthList(lines);
        if (len==max)
            pos = words.insert(pos, allWords[k]);
    }
    return words;
}
```

חומרי עזר שהוכנו ע"י משתתפי קורס מורים מובילים תשע"א
ניתן להשתמש בחומרים לצורך הוראה בלבד. אסור לפרסם את החומרים או לעשות בהם שימוש מסחרי כלשהו ללא קבלת אישור מראש מצוות הפיתוח

הפעולה מחזירה את מספר השורות הגדול ביותר עבור מילה באוסף.

```
private int maxLinesPerWord()
{
    String[] allWords = this.wordAndLines.getAllKeys();
    int max = 0; // maximum lines for a word
    int len;
    for (int k=0; k<allWords.length; k++)    [ O(n) ]
    {
        len = this.getNumOfLines(allWords[k]);    [ O(m) ]
        if (len>max)
            max = len;
    }
    return max;
}
```

$O(n \cdot m)$

הפעולה מקבלת רשימה של שלמים ומחזירה את אורכה.

```
private int lengthList(List<Integer> lst)
{
    int count = 0;
    Node<Integer> pos = lst.getFirst();
    while (pos!=null)
    {
        count++;
        pos = pos.getNext();
    }
    return count;
}
```

4) יעילות הפעולה **wordsInMaximumLines** היא $O(n \cdot m)$, כאשר n מספר המילים באוסף ו- m מספר השורות באוסף.

הסבר: במקרה הגרוע עבור כל מילה באוסף יש לעבור על כל השורות. הפעולה עושה שימוש בפעולה **maxLinesPerWord** שהיא בעלת סיבוכיות זמן ריצה $O(m)$ והיא מתבצעת n פעמים.

(5) הפעולה מחזירה רשימה של מספרי השורות בהם מילים word1,word2 מופיעות ביחד.

```
public static List<Integer> appearSameLine(WordLines wl, String word1, String word2)
{
    List<Integer> lBoth = new List<Integer>();
    List<Integer> lines1 = wl.getLines(word1);
    List<Integer> lines2 = wl.getLines(word2);
    if (lines1!=null && lines2!=null)
        lBoth = Main.cutList(lines1, lines2);
    return lBoth;
}
```

הפעולה מקבלת 2 רשימות ממוינות בסדר עולה של מספרים שלמים ומחזירה את רשימת החיתוך שלהן.

```
private static List<Integer> cutList(List<Integer> lst1, List<Integer> lst2)
{
    List<Integer> cutL = new List<Integer>();
    Node<Integer> p1 = lst1.getFirst();
    Node<Integer> p2 = lst2.getFirst();
    Node<Integer> p3 = cutL.getFirst();
    int x,y;
    while (p1!=null && p2!=null)
    {
        x = p1.getInfo();
        y = p2.getInfo();
        if (x==y)
        {
            p3 = cutL.insert(p3, x);
            p1 = p1.getNext();
            p2 = p2.getNext();
        }
        else
        { // x!= y
            if (x<y)
                p1 = p1.getNext();
            else // x>y
                p2 = p2.getNext();
        }
    }
    return cutL;
}
```

שאלה מספר 4: ייצוג שפת תגיות HTML

שפת HTML היא שפת תגיות.

לכל תגית יש אוסף תכונות אפשריות ולכל תכונה יש אוסף ערכים אפשריים.

לדוגמא

שם התגית	התכונות	הערכים
<div>	dir	rtl , ltr
	align	left, center, right

כדי להקל על הכותב בשפת HTML נרצה לבנות אוסף שיכיל את כל התגיות האפשריות עם התכונות והערכים שהן יכולות לקבל.

הצע לפחות שתי דרכים שונות לייצוג המידע הנ"ל.

א- כתוב תיאור מילולי לכל אחד מהייצוגים שבחרת.

ב- כתוב את כותרת המחלקה והתכונות המתאימות לכל אחד מהייצוגים שבחרת.

ג- לכל אחד מהייצוגים שהצעת, הסבר כיצד תמומשנה הפעולות הבאות:

- בניית אוסף תגיות ריק

- הוספת תגית, תכונה וערך לאוסף

- הצגת כל האוסף

ד- נתח והסבר את היתרונות והחסרונות של כל אחד מהייצוגים שהצעת.

הצעה לפתרון שאלה 4

ייצוגים שונים לתגיות HTML

אפשרות ראשונה לייצוג:

א- אוסף התגיות ייוצג על ידי מפה כאשר:

המפתח - שם התגית

הערך - מפה ש-

המפתח שלה הוא – שם התכונה

והערך הוא רשימה של מחרוזות אשר מהוות את הערכים האפשריים לתכונה זו.

אפשר היה לפצל לטיפוס מורכב כאשר הערך הוא טיפוס **Property** אשר בנוי גם הוא כמפה אשר

המפתח הוא שם התכונה והערך – רשימת ערכים אפשריים.

ב- כותרת המחלקה: HTML

תכונת המחלקה: `private Map<Map<List<String>>> htmlTags;`

ג- **בניית אוסף תגיות ריק**

זו פעולת הבנייה:

```
public HTML()
{
    this.htmlTags = new Map<Map<List<String>>>();
}
```

הוספת תגית, תכונה וערך לאוסף

אם התגית לא קיימת באוסף אז :

- ניצור רשימה של מחרוזות ונוסיף אליה את הערך

- ניצור מפה ונוסיף לה את התכונה

אחרת, התגית קיימת האוסף

- אם התכונה לא קיימת

- ניצור רשימה של ערכים

- נוסיף את הערך לרשימה

- נוסיף את הערך למפת התכונות

- אחרת, התכונה קיימת

- אם הערך לא קיים

- נוסיף אותו לרשימת הערכים

- נוסיף את התכונה עם הערכים לתגית


```
/**
 * add tag name with property and value
 * @param tag: tag's name
 * @param property: property's name
 * @param value: String
 */
public void add(String tag, String property, String value)
{
    // step1 : search if tag exists
    Map properties = this.htmlTags.getValue(tag);
    if ( properties == null )
        { // tag is not exist
            // create list of values and add the value to the list of values
            List values = new List<String>();
            values.insert(null, value);
            // create Map for the property and add value-list to property
            properties = new Map<List<String>>();
            properties.insert(property, values);
        }
    else // tag exists
        {
            List<String> lstValues = (List<String>) properties.getValue(property);
            if (lstValues == null)
                { // no such property --> create List for values , add value
                    lstValues = new List<String>();
                    lstValues.insert(null, value);
                    // add new property to existed tag
                    // insert values-list to all properties
                    properties.insert(property, lstValues);
                }
            else
                { // property exists, check if values exists
                    boolean valueExist = false;
                    Node<String> pos = lstValues.getFirst();
                    while (pos!=null )
                        {
                            if (pos.getInfo().equals(value))
                                valueExist = true;
                            else
                                pos = pos.getNext();
                        }
                    // end while
                    if ( ! valueExist)
                        lstValues.insert(null, value);
                }
            // end if-else property exists
        }
    // end if-else tag exists
    // add tag to HTML, if exists , update the properties of the tag
    this.htmlTags.insert(tag, properties);
} // end add
```

חומרי עזר שהוכנו ע"י משתפי קורס מורים מובילים תשע"א
ניתן להשתמש בחומרים לצורך הוראה בלבד. אסור לפרסם את החומרים או לעשות בהם שימוש מסחרי כלשהו ללא קבלת אישור מראש מצוות הפיתוח

הצגת כל האוסף פעולת ה toString

```
/**
 * @return Strign representing HTML
 */
public String toString()
{
    String str="";
    String[] allTags = this.htmlTags.getAllKeys();
    String tag;
    Map<List<String>> properties;
    String[] allProperties;
    String property;
    for (int i = 0; i < allTags.length; i++)
    {
        tag = allTags[i];
        str += tag+ "\n";
        properties = this.htmlTags.getValue(tag);
        allProperties = properties.getAllKeys();
        for (int j = 0; j < allProperties.length; j++)
        {
            property = allProperties[j];
            str += " " + property + ": ";
            str+= properties.getValue(property);
            str+="\n";
        }
        // str += "\n";
    }
    return str;
}
} // end class HTML
```

אפשרות שנייה לייצוג:

- נגדיר מחלקה בשם **Property** שמכילה שתי תכונות:
- שם התכונה – name מטיפוס מחרוזת.
 - אוסף ערכים – values שהוא רשימה של מחרוזות. כל מחרוזת מייצגת ערך אפשרי.
- אוסף תגיות HTML ייוצג ע"י מחלקה בשם **HTML** אשר מכילה שתי תכונות:
- שם התגית – tagName מטיפוס מחרוזת.
 - אוסף תכונות tagProperties הוא רשימה של Properties .

המחלקה **Property**

שם התכונה	String name
רשימת ערכים	List<String> values

המחלקה **HTML**

שם תגית	String tagName
רשימת תכונות	List<Property> properties

דוגמה לתוכנית ראשית

```
/*
 * Main for HTML: add tag , property and value to HTML
 */

package HTMLtag;

/**
 * @author EVI & Michal
 * @version 11/6/11
 */
public class MainHTML
{
    public static void main (String[] args)
    {
        HTML html = new HTML();
        html.add("body", "dir", "rtl");
        html.add("body", "dir", "ltr");
        html.add("body", "align", "center");
        html.add("table", "border", "1");
        html.add("table", "border", "2");
        html.add("table", "border", "3");

        System.out.println(html);
    }
}
/* output
body
-----
align: [center]
dir: [ltr,rtl]
table
-----
border: [3,2,1]
 */
```