

פתרון המבחן בתוכנה 1

סמסטר א', מועד א', תשע"ד

04/02/2014

דן הלפרין, שחר מעוז, יעל אמסטרדמר, דביר נתנאלי

הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות**, חלקו את זמנכם ביעילות.
- אסור השימוש בחומר עזר כלשהו, כולל מחשבוניו או כל מכשיר אחר פרט לעט. בסוף הבחינה צורף לנוחותכם נספח ובו תיעוד מחלקות שימושיות.
- יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תיפסל. **תשובות במחברת הבחינה לא תיבדקנה**. במידת הצורך ניתן לכתוב בגב טופס הבחינה.
- יש למלא מספר סידורי (מס' מחברת) ומספר ת.ז על כל דף של טופס הבחינה.
- ניתן להניח שכל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import.
- במקומות בהם תתבקשו לכתוב מתודה (שירות), ניתן לכתוב גם מתודות עזר.
- ניתן להוסיף הנחות לגבי אופן השימוש בשירותים המופיעים בבחינה, ובלבד שאין הן סותרות את תנאי השאלה. יש לתעד הנחות אלו כחווה (תנאי קדם, תנאי בתר) בתחביר המקובל, שייכתב בתחילת השורות.

לשימוש הבודקים בלבד:

שאלה	משקל	א	ב	ג	ד	ה	סה"כ
1	25						
2	35						
3	15						
4	25						

בהצלחה!

© כל הזכויות שמורות

מבלי לפגוע באמור לעיל, אין להעתיק, לצלם, להקליט, לשדר, לאחסן במאגר מידע, בכל דרך שהיא, בין מכאנית ובין אלקטרונית או בכל דרך אחרת כל חלק שהוא מטופס הבחינה.

שאלה 1 (25 נקודות)

ממשו מתודה המקבלת שם של קובץ טקסט המכיל מילים, ומחזירה את רשימת המילים ממוינת ללא חזרות. המילים ברשימה המוחזרת תהיינה ממיינות קודם לפי אורך (מיון ראשי, בסדר יורד), ואח"כ לפי מספר המופעים שלהן בקובץ הקלט (מיון משני, בסדר יורד).

כלומר, במקום הראשון ברשימה תופיע המילה הארוכה ביותר בקובץ. במידה וקיימות מספר מילים באותו האורך, הסדר הפנימי שלהן יהיה לפי מספר המופעים בקובץ הקלט כך שהמילה הנפוצה ביותר תופיע ראשונה.

להלן חתימת המתודה:

```
public static List<String> sortByLengthAndFrequency (String inputFileName)
```

ניתן להיעזר בנספח המופיע בסוף הבחינה.

הערות:

- הקובץ inputFileName יכיל רשימת מילים כאשר כל מילה מופיעה בשורה נפרדת.
- ניתן להניח שקובץ הקלט נמצא בתיקייה בה תרוץ התוכנית (אין צורך לשנות נתיב).
- ניקוד מלא יינתן למימוש אלגנטי העושה שימוש באוספים (כגון Map, Set...) וכולל מעבר יחיד על הקובץ (פתיחה וקריאה של הקובץ רק פעם אחת).
- במקרה של בעיה כלשהי במהלך ריצת המתודה (חריגה), המתודה תחזיר null.

לדוגמא, עבור קובץ קלט המכיל את המילים הבאות:

```
ABC
ABC
ABB
ABCDE
A
B
B
B
```

תוחזר הרשימה: [ABCDE, ABC, ABB, B, A]

```
דוגמא ראשונה לפתרון (שימוש ב-Scanner ובקומפוטור על Map.Entry שמוגדר כמחלקה אנונימית):
public static List<String> sortByLengthAndFrequency (String inputFileName) {

    Scanner inputScanner = null;

    try {
        // Counting word occurrences in the input file using a HashMap
        Map<String,Integer> wordsFreq = new HashMap<>();

        inputScanner = new Scanner (new File(inputFileName));

        while (inputScanner.hasNext()) {
            String curWord = inputScanner.next();
            if (!wordsFreq.containsKey(curWord))
                wordsFreq.put(curWord, 1);
            else
                wordsFreq.put(curWord,wordsFreq.get(curWord)+1);
        }

        // Sorting the map's entries by value
        List<Map.Entry<String,Integer>> entries=new ArrayList<>( wordsFreq.entrySet());

        Collections.sort(entries, new Comparator<Map.Entry<String, Integer>>() {

            // Compares words based on length (descending) and then by Frequency (descending)
            @Override
            public int compare(Entry<String, Integer> o1, Entry<String, Integer> o2) {
                // If length is the same, secondary order is by frequency
                if (o1.getKey().length()==(o2.getKey().length()))
                    return ((Integer) o2.getValue()) - ((Integer) o1.getValue()); // descending order
                return o2.getKey().length() - o1.getKey().length();
            }
        });

        // Builds the result list composed of only keys
        List<String> result = new ArrayList<>();
        for (Map.Entry<String, Integer> entry: entries)
            result.add(entry.getKey());

        return result;
    } catch (Exception e) {
        return null;
    }
    finally {
        if (inputScanner!=null)
            inputScanner.close();
    }
}
```

דוגמא נוספת לפתרון (שימוש ב-BufferedReader ובקומפרטור חיצוני שממין מחרוזות):

```

public static List<String> sortByLengthAndFrequency (String inputFileName) {

    BufferedReader inputReader = null;

    try {

        // Counting word occurrences in the input file using a HashMap
        Map<String,Integer> wordsFreq = new HashMap<>();
        String curWord;

        inputReader = new BufferedReader (new FileReader(inputFileName));
        while ((curWord=inputReader.readLine())!=null) {
            if (!wordsFreq.containsKey(curWord))
                wordsFreq.put(curWord, 1);
            else
                wordsFreq.put(curWord,wordsFreq.get(curWord)+1);
        }
        inputReader.close();

        // Extracting the words without repetitions from the map keys
        List<String> wordList = new ArrayList<>(wordsFreq.keySet());

        // Sorting the word list using an external comparator which takes wordFreq as a parameter
        Collections.sort(wordList, new MyComparator(wordsFreq) );

        return wordList;

    } catch (Exception e) {
        return null;
    }
}

// Nested class defining a String comparator.
public static class MyComparator implements Comparator<String> {
    Map<String,Integer> wordFreqMap;

    // Constructor
    public MyComparator (Map<String,Integer> map) {
        this.wordFreqMap = map;
    }

    // Compares based on length (descending) and then by Frequency (descending)
    @Override
    public int compare(String s1, String s2) {
        if (s2.length() > s1.length())
            return 1;
        if (s2.length() < s1.length())
            return -1;
        if ( wordFreqMap.get(s2) > wordFreqMap.get(s1))
            return 1;
        else if ( wordFreqMap.get(s2) < wordFreqMap.get(s1))
            return -1;

        return 0;
    }
}

```

דרכי פיתרון אפשריות:

- קריאת הקובץ אל תוך מבנה נתונים מסוג `HashMap<String,Integer>` המחזיק את תדירויות המילים (המפתח הוא המילה והערך הוא מספר הפעמים בו המילה מופיעה בקובץ). שימו לב שהמפתחות של מפת התדירויות הם למעשה קבוצת המילים ללא חזרות.
- לצורך מיון המילים יש לשמור אותן במבנה נתונים ששומר על סדר. ישנן כמה אפשרויות לטיפול האיברים ברשימה אותה נמיון:

- רשימה של `Map.Entry<String,Integer>` (דוגמא ראשונה בקוד)
המרה של איברי המפה לרשימה של `Entries` המכילים את המילה כמפתח ואת תדירות המילה כערך, תבצע למשל ע"י

```
List< Map.Entry<String,Integer>> entries = new ArrayList<>(map.entrySet())
```

מיון זוגות הערכים יתבצע ע"י קריאה ל- `Collections.sort` תוך ציון קומפרטור שיועד למיון `Map.Entry`.

```
Collections.sort( entries, new Comparator<Map.Entry<String,Integer>>(){...})
```

לבסוף יש להמיר את רשימת ה-`entries` לרשימה של מילים ע"י שליפת המפתחות של כל איבר ברשימה.

שימו לב שכאן לא נדרשת גישה למפת תדירויות המילים כי כל המידע כבר נמצא בזוגות הערכים. הקומפרטור יכול להיות מוגדר כמחלקה מקוננת ולאו דווקא אנונימית.

- רשימה של מחרוזות (דוגמא שניה בקוד)
המרה של מפתחות המפה לרשימה של מחרוזות ללא חזרות תבצע למשל ע"י

```
List<String> words = new ArrayList<>(map.keySet());
```

מיון רשימת המילים יתבצע ע"י קריאה ל- `Collections.sort` תוך ציון קומפרטור שיועד למיון מחרוזות ואשר יש לו גישה למפת תדירויות המילים.

```
Collections.sort( words, new Comparator<String>(){...})
```

הקומפרטור יכול להיות מחלקה אנונימית (ואז יהיה ניתן לגשת מתוכו למפת תדירויות המילים שבמחלקה העוטפת, אם היא תוגדר כ-`final`). לחילופין ניתן להגדיר את הקומפרטור כמחלקה מקוננת בעלת שם ואז כדי לגשת למפת תדירויות המילים נשלח אותה כפרמטר לבנאי שלו. לבסוף, פשוט נחזיר את `words` בתור רשימת המילים הממוינת ללא חזרות.

- רשימה של טיפוס עזר המחזיק זוג ערכים
ניתן גם להגדיר מחלקת עזר המייצגת זוג ערכים (מילה ואת התדירות שלה), ואז למיון רשימה של אובייקטים מטיפוס זה. די מקביל לפתרון הקודם. במקרה זה מחלקת העזר עשויה לממש את הממשק `Comparable` ואז נוכל להפעיל את המיון מבלי לספק קומפרטור חיצוני.

טעויות נפוצות:IO

- אי סגירה של ה-Scanner
- שימוש לא תואם ב-`hasNext` ו-`nextLine` או ב-`hasNextLine` ו-`next` (לא הורדו נקודות)

Exceptions

- תפיסת החריגה צריכה להיעשות בסוף המתודה תוך שימוש ב-`catch (Exception e)`
- גישה לאובייקט ה-Scanner מבלוק ה-`catch` או מה-`finally` דרך רפרנס שהוגדר בתוך ה-`try` (ועל כן טווח ההכרה שלו יהיה רק בתוך בלוק ה-`try`).

Generics, Collections ומיון

- נסיון למיין Map ע"י `Collections.sort` (שמקבל List)
- שימוש ב-`TreeMap` במקום ב-`HashMap` מבלי להשתמש בתכונות המיון של `TreeMap` (פחות יעיל, לא הורדו נקודות).
- יצירת מופע של `TreeMap` תוך ציון קומפרטור שהינו גנרי על טיפוס השונה מטיפוס המפתחות של ה-`TreeMap`. למשל, לא ניתן להגדיר:

```
Map<String,Integer> map = new TreeMap<>(new
Comparator<Map.Entry<String,Integer>>(){...})
```
- גישה למבנה נתונים שאינו מוגדר כ-`final` מתוך מחלקה אנונימית של הקומפרטור.
- הגדרה שגויה של ה-`Comparator` כך שימיין בסדר עולה במקום בסדר יורד כפי שהוגדר בשאלה.
- לא ניתן לעדכן ערך ב-`Map` בצורה הבאה: `map.get(key)++` או בצורה הבאה: `map.put(key, map.get(key)++)`
- לא ניתן לרוץ בלולאת `for` על מפה! (`Map` אינו `Iterable`)
- בדיקה של קיום ערך באוסף עדיף שתיעשה על `HashSet` ולא על `List` (היו הרבה פתרונות שהחזיקו את שני מבני הנתונים ועדיין ביצעו את הבדיקה על ה-`List`) – לא הורדו נקודות.

שאלה 2 (35 נקודות)

שאלה זו עוסקת בתכנון טיפוס נתונים עבור מטריצה שאיבריה $M(i,j)$ הם מטיפוס `double`. התא הראשון במטריצה הוא $M(0,0)$.

נתון הממשק `IMatrix` ובו השירותים הבאים:

```
/** מטריצה שאיבריה double */
public interface IMatrix {

    /** השירות מחזיר את מספר השורות */
    public int nRows();

    /** השירות מחזיר את מספר הטורים */
    public int nColumns();

    /** M(i,j) האיבר האבר */
    public double element(int i, int j);

    /** השירות מחזיר מטריצה חדשה המייצגת את סכום המטריצות */
    public IMatrix add(IMatrix other);
}
```

לרשותנו מתודה `parseMatrixContents` סטטית אשר קוראת את תוכן המטריצה מקובץ ומחזירה `Map` שהמפתח בו הוא מטיפוס `IndexPair` והערך הוא `Double`. המבנה `Map` שומר רק ערכים של המטריצה השונים מאפס. לא נשתמש בה ישירות בשאלה, אלא רק ב `Map` שהיא מחזירה:

```
public static Map<IndexPair, Double> parseMatrixContents(
    String fileName)
```

המחלקה `IndexPair` מכילה זוג מספרים שלמים המייצגים את השורה והטור של האיבר במטריצה, ומוגדרת כך:

```
public class IndexPair {
    public final int row;
    public final int column;

    public IndexPair(int row, int column) {
        this.row = row;
        this.column = column;
    }

    public boolean equals(Object obj) {...}
    public int hashCode() {...}
}
```

סעיף א (7 נקודות)

כתבו המחלקה StandardMatrix המממשת את IMatrix בעזרת מערכים של ג'אווה. כתבו השדות, וכן בנאי שמקבל כקלט מספר הטורים, מספר השורות, ו-Map שהוחזר על ידי המתודה parseMatrixContents, ויוצר מטריצה בגדלים המתאימים ועם הערכים המתאימים. אין צורך לממש בסעיף זה את המתודות של IMatrix.

```
public class StandardMatrix implements IMatrix {
```

```
private double[][] elementValues;
```

```
public StandardMatrix(int numRows, int numColumns,  
Map<IndexPair, Double> matrixAsMap) {
```

```
    elementValues = new double[numRows][numColumns];
```

```
    for (Entry<IndexPair, Double> entry : matrixAsMap.entrySet()) {
```

```
        IndexPair pair = entry.getKey();
```

```
        Double value = entry.getValue();
```

```
        elementValues[pair.row][pair.column] = value;
```

```
    }
```

```
    }  
}
```


סעיף ב (20 נקודות)

מטריצה דלילה היא מטריצה שרבים מאיבריה הם אפסים. נציג מטריצה כזו בעזרת מערך של רשימות מקושרות, רשימה אחת כנגד כל שורה במטריצה. תא ברשימה יוגדר בעזרת המחלקה `SMCell`:

```
public class SMCell {
    private int column;
    private double value;
    private SMCell next;

    public SMCell(int column, double value) {
        this.column = column;
        this.value = value;
    }

    public int getColumn() {
        return column;
    }

    public double getValue() {
        return value;
    }

    public SMCell getNext() {
        return next;
    }

    /* הוסיפו מתודות אם יש צורך */

```

```
public void setNext(SMCell next) {
    this.next = next;
}

```

```
}
```

השלימו את מימוש המתודות של `SMCell`.
 כתבו המחלקה `SparseMatrix` המממשת את `IMatrix`:
 כתבו השדות הנחוצים וכן ממשו בנאי המקבל כקלט את מספר השורות והעמודות ויוצר `SparseMatrix` ריקה (בה כל הרשימות המקושרות ריקות) והמתודות `add - I element`.
 הניחו כי `nColumns()` -I `nRows()` כבר נתונות לכם.

שימו לב: סדר התאים ברשימה המקושרת אינו תואם בהכרח את סדר העמודות. לדוגמא, ייתכן שהתא הראשון מייצג את הערך בעמודה 7, התא שני מייצג את הערך בעמודה 4, וכך הלאה.

הנחיה: במימוש `add` עליכם ליצור ולהחזיר `SparseMatrix` שתייצג את התוצאה. כדי להוסיף ערכים למטריצה המחושבת, ממשו את מתודת העזר `addElement` (בסוף המחלקה) והשתמשו בה.

```
public class SparseMatrix implements IMatrix {
```

```
private SMCell[] rows;  
private int numColumns;
```

```
/**  
 * Creates an empty sparse matrix (representing a matrix with only 0  
 * elements) of size numRows x numColumns  
 */  
public SparseMatrix(int numRows, int numColumns) {
```

```
    this.numColumns = numColumns;  
    rows = new SMCell[numRows];
```

```
}
```

```
@Override  
public int numRows() {...}
```

```
@Override  
public int numColumns() {...}
```

```
@Override  
public double element(int i, int j) {
```

```
    SMCell cell = rows[i];  
    while (cell != null) {  
        if (cell.getColumn() == j) {  
            return cell.getValue();  
        }  
        cell = cell.getNext();  
    }  
    return 0;
```

```
}
```

```
@Override
```

```
public IMatrix add(IMatrix other) {
```

```
    int rows = nRows();
    int columns = nColumns();
    SparseMatrix sum = new SparseMatrix(rows, columns);

    /* simple implementation */
    for (int i = 0; i < rows; i++) {
        for (int j = columns - 1; j >= 0; j--) {
            double elemSum = element(i, j) + other.element(i, j);
            addElement(sum, i, j, elemSum);
        }
    }

    return sum;
}
```

```
}
```

```
/**
```

```
 * adds an element to the linked list of the given row,
 * with the given column and value
```

```
*/
```

```
private static void addElement(SparseMatrix matrix, int row,
    int column, double value) {
```

```
    if (value != 0.0) {
        SMCell cell = new SMCell(column, value);
        SMCell oldCell = matrix.rows[row];
        matrix.rows[row] = cell;
        cell.setNext(oldCell);
    }
}
```

```
}
```

```
}
```

סעיף ג (8 נקודות)

מתוך מחשבה שנשתמש במספר גדול מאד של תאים של SparseMatrix לאורך חיי התוכנית, אבל בכל רגע נתון מספרם לא יהיה רב, נוסיף בתוך המחלקה SMCell מנגנון הקצאה מקומי שימחזר תאים. (המטרה היא לחסוך בפעולות היקרות יחסית של הקצאת אובייקטים חדשים ושחרורם על ידי אוסף הזבל.)

הנחיה: המנגנון ישתמש בשדה סטטי freeList שמצביע לתחילת רשימה של תאים הממוחזרים (תאים שנוצרו בעבר ולאחר מכן "שוחררו", כך שאינם בשימוש כרגע). השלימו השורות החסרות במתודה הסטטית alloc ובמתודת המופע free.

ניתן להניח כי free אינה נקראת עבור תאים אשר עדיין משמשים כחלק מרשימה כלשהי, ובפרט לא עבור תאים ברשימה freeList. לכן, free אינו צריך לטפל בהסרת התא מרשימה.

```
public class SMCell {
    ...

    private static SMCell freeList;

    public static SMCell alloc(int column, double value) {
        if (freeList == null)
            return new SMCell(column, value);
        else {
```

```
            SMCell cell = freeList;
            freeList = freeList.getNext();
            cell.setNext(null);
            cell.column = column;
            cell.value = value;
            return cell;
```

```
        }
    }

    public void free() {
```

```
        setNext(freeList);
        freeList = this;
```

```
    }
}
```

שאלה 3 (15 נקודות)

קראו את קוד המחלקות הבאות וענו על השאלות.

```
public class Base {  
  
    public String fld = "base";  
  
    public Base(String fld) {  
        /** 1 **/  
        this.fld = fld;  
    }  
  
    public static void print(String value, Object source) {  
        System.out.println(source.getClass().getSimpleName() + " "  
            + value);  
    }  
  
    public String getFld() {  
        return fld;  
    }  
  
}
```

```
public class Sub extends Base{  
  
    public String fld;  
  
    public Sub() {  
        /** 2 **/  
        super("bus");  
        this.fld = "sus";  
    }  
  
    public String getFld() {  
        return super.getFld() + "sub";  
    }  
  
    public static void main(String[] args) {  
        Base base = new Sub();  
        /** 3 **/  
    }  
  
}
```

בכל אחד מהסעיפים הבאים נוספת שורת קוד באחד מהמקומות המסומנים בקוד. הנכם מתבקשים לציין מהו הפלט של הרצת פונקציית ה-main בכל אחד מהמקרים. אם לדעתכם התכנית אינה עוברת קומפילציה או זורקת שגיאה בזמן ריצה, הסבירו מה סוג השגיאה (זמן קומפילציה או זמן ריצה) וממה היא נובעת. בכל מקרה (שגיאה או ריצה תקינה) יש לכתוב הסבר קצר. ניתן להיעזר בנספח עבור `getClass` ו-`getSimpleName`.

סעיף א' (3 נק')

```
print(fld, this);
```

/** 1 */ מוחלף ב-

הפלט יהיה: Sub bus
 הבנאי של Sub מפעיל את הבנאי של Base. בתוך בנאי זה, fld (בניגוד ל- this.fld) הוא הארגומנט של הבנאי וערכו "bus". הקריאה ל-print מדפיסה את הטיפוס האמיתי של העצם הנוכחי, כלומר Sub, ואז את "bus".

סעיף ב' (3 נק')

```
print(getFld(), this);
```

/** 1 */ מוחלף ב-

הפלט יהיה: Sub basesub
 במקודם, הבנאי של Sub מפעיל את הבנאי של Base. getFld() נקראת לפי הטיפוס הדינאמי, כלומר Sub, ומחזירה את השרשור של ערך השדה fld ב-Base ו-"sub". בנק' הנוכחית ערך השדה הוא "base". במקודם, הקריאה ל-print מדפיסה Sub ואז את תוצאת getFld(), "basesub".

סעיף ג' (3 נק')

```
print(fld, this);
```

/** 2 */ מוחלף ב-

תקבל שגיאת קומפילציה.
 השורה הראשונה בבנאי חייבת להכיל קריאה ל-super(...) או this(...), לא ניתן להוסיף שם קוד.

סעיף ד' (3 נק')

```
print(getFld(), base);
```

/** 3 */ מוחלף ב-

תקבל שגיאת קומפילציה.
 לא ניתן לקרוא למתודת מופע (getFld) בהקשר סטטי (main) מבלי לציין על איזה מופע המתודה נקראת.

סעיף ה' (3 נק')

```
print(base.fld, base);
```

/** 3 */ מוחלף ב-

הפלט יהיה: Sub bus
 print קוראת למימוש ב-Base. base.fld בוחר את השדה לפי הטיפוס הסטטי של base, כלומר, את fld במחלקה Base, שערכו לאחר האתחול הוא "bus". הקריאה ל-print מדפיסה את הטיפוס הדינאמי של base, כלומר Sub, ואז את "bus".

שאלה 4 (25 נקודות)

בשאלה זו נממש תוכנית לדפדוף אקראי בין עוגיות מזל הכוללת ממשק משתמש גרפי (GUI).

א. (10 נקודות) ממשו מתודה המקבלת מערך של מחרוזות ומערבבת אותו בצורה אקראית.

להלן חתימת המתודה:

```
private void shuffle (String[] arrayToShuffle)
```

אלגוריתם הערבוב שיש לממש: עברו על המערך מהסוף להתחלה, מהאיבר האחרון ועד לאיבר השני, ובכל פעם בצעו החלפה (swap) בין האיבר הנוכחי לאיבר אקראי. בחרו את האיבר האקראי בכל פעם מתוך החלק של המערך מן האיבר הראשון ועד לאיבר הנוכחי (כולל).

הערות:

- יש להחזיר את אותו אובייקט של מערך כאשר איבריו מעורבבים (in-place shuffle).
- אין להשתמש בשורת shuffle של המחלקה Collections.
- יש להשתמש בשירות nextInt של המחלקה Random כפי שמופיע בסוף הבחינה.

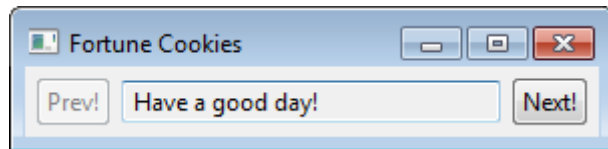
```
private void shuffle (String[] arrayToShuffle) {  
  
    Random random = new Random();  
    for (int i=arrayToShuffle.length-1; i>0; i--) {  
        int newPlace = random.nextInt(i+1);  
        String temp = arrayToShuffle[newPlace];  
        arrayToShuffle[newPlace] = arrayToShuffle[i];  
        arrayToShuffle[i] = temp;  
    }  
  
}
```

ב. (15 נקודות) התוכנית מזל כתבה ממשק משתמש גרפי לדפדוף אקראי בין עוגיות מזל. התוכנית שלה קוראת רשימת מחרוזות מקובץ אל תוך מערך ומערבבת אותו באמצעות המתודה שכתבתם בסעיף א'. התוכנית מציגה למשתמש את עוגיית המזל הראשונה (במערך המעורבב) ומאפשרת לו לדפדף קדימה ואחורה בין עוגיות המזל (במערך המעורבב) באמצעות שני כפתורים, Prev! ו Next!.

התוכנית של מזל שומרת ומעדכנת את המצב הנוכחי שלה (מי העוגייה המוצגת כרגע). על מנת למנוע גלישה אל מחוץ למערך, מזל דאגה לכך שכאשר מוצגת העוגייה הראשונה, כפתור ה Prev! יהיה disabled, וכאשר מוצגת העוגייה האחרונה, כפתור ה Next! יהיה disabled. היא השתמשה לצורך כך במתודה setEnabled של המחלקה Button אשר קובעת אם כפתור יהיה enabled או לא:

```
public void setEnabled(boolean enabled)
```

לדוגמא, כך נראה הממשק הגרפי שכתבה מזל כאשר הוא מראה את העוגייה הראשונה במערך המעורבב:



לרוע מזלה של מזל, ארבעה קטעי קוד נמחקו מהתוכנית שלה. עליכם לעזור לה להשלים הקטעים החסרים.


```
public class FortuneCookiesBrowser {
```

```
// ADD MISSING CODE HERE
private Shell shell = null;
private Button buttonNext = null;
private Button buttonPrev = null;
private Text textFC = null;
private int current;
```

```
private String[] cookies;
```

```
/* call createShell and run event loop */
```

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

```
    Display display = Display.getDefault();
    FortuneCookiesBrowser app = new FortuneCookiesBrowser();
    app.buildCookies();
    app.createShell(display);
```

```
    while (!app.shell.isDisposed()) {
        if (!display.readAndDispatch())
            display.sleep();
    }
```

```
// ADD MISSING CODE HERE
```

```
display.dispose();
```

```
}
```

```
// load cookies from file and shuffle them
```

```
private void buildCookies() {...}
```

```
private void shuffle(String[] arrayToShuffle) {...}
```

```
/* show current cookie and enable/disable next/prev buttons */
```

```
private void update() {
```

```
// ADD MISSING CODE HERE (BODY OF UPDATE METHOD)
```

```
textFC.setText(cookies[current]);
buttonNext.setEnabled(current < cookies.length - 1);
buttonPrev.setEnabled(current > 0);
```

```
}
```

```

/* create the GUI */
private void createShell(Display display) {

    shell = new Shell(display);

    shell.setText("Fortune Cookies");

    // layout manager: a grid with 3 unequal columns
    shell.setLayout(new GridLayout(3, false));

    buttonPrev = new Button(shell, SWT.NONE);
    buttonPrev.setText("Prev!");
    buttonPrev.setLayoutData(new GridData(SWT.LEFT,
                                           SWT.CENTER, false, false));

    textFortune = new Text(shell,
                           SWT.BORDER | SWT.READ_ONLY);
    textFortune.setLayoutData(new GridData(SWT.FILL,
                                           SWT.CENTER, true, false));

    buttonNext = new Button(shell, SWT.NONE);
    buttonNext.setText("Next!");
    buttonNext.setLayoutData(new GridData(SWT.RIGHT,
                                          SWT.CENTER, false, false));

```

```

// ADD MISSING CODE HERE (BEHAVIOR OF THE TWO BUTTONS)
// USING THE METHOD
// Button.addSelectionListener(SelectionListener listener),
// THE ABSTRACT CLASS SelectionAdapter,
// AND ITS METHOD public void widgetSelected(SelectionEvent e)

buttonNext.addSelectionListener(
    new SelectionAdapter() {
        public void widgetSelected(SelectionEvent e) {
            current++;
            update();
        }
    });

buttonPrev.addSelectionListener(
    new SelectionAdapter() {
        public void widgetSelected(SelectionEvent e) {
            current--;
            update();
        }
    });

current = 0;

```

```

update();

```

```

// causes the layout manager to lay out the shell
shell.pack();

```

```

// opens the shell on the screen
shell.open();

```

```

}

```

נספח

Class File

Constructor Summary

[File](#)([String](#) pathname)

Creates a new File instance by converting the given pathname string into an abstract pathname.

Class FileReader

Constructor Summary

[FileReader](#)([File](#) file)

Creates a new `FileReader`, given the `File` to read from.

[FileReader](#)([String](#) fileName)

Creates a new `FileReader`, given the name of the file to read from.

Method Summary

Methods inherited from class [java.io.InputStreamReader](#)

[close](#), [getEncoding](#), [read](#), [read](#), [ready](#)

Methods inherited from class [java.io.Reader](#)

[mark](#), [markSupported](#), [read](#), [read](#), [reset](#), [skip](#)

Class BufferedReader

Constructor Summary

[BufferedReader](#)([Reader](#) in)

Create a buffering character-input stream that uses a default-sized input buffer.

Method Summary

void	close ()	Close the stream.
int	read ()	Read a single character.
int	read (char[] cbuf, int off, int len)	Read characters into a portion of an array.
String	readLine ()	Read a line of text.
boolean	ready ()	Tell whether this stream is ready to be read.
void	reset ()	Reset the stream to the most recent mark.
long	skip (long n)	Skip characters.

Class Scanner

Constructor Summary

[Scanner](#) ([File](#) source)

[Scanner](#) ([InputStream](#) source)

[Scanner](#) ([Readable](#) source)

Method Summary

void	close () Closes this scanner.
boolean	hasNext () Returns true if this scanner has another token in its input.
boolean	hasNextLine () Returns true if there is another line in the input of this scanner.
String	next () Finds and returns the next complete token from this scanner.
String	nextLine () Advances this scanner past the current line and returns the input that was skipped.
Scanner	useDelimiter (String pattern) Sets this scanner's delimiting pattern to a pattern constructed from the specified String.

Interface List<E>

Method Summary

boolean	add (E e)
boolean	addAll (Collection <? extends E > c)
void	clear ()
boolean	contains (Object o)
E	get (int index)
int	indexOf (Object o)
Iterator < E >	iterator ()
E	remove (int index)
E	set (int index, E element)
int	size ()

Interface Set<E>

Method Summary

boolean	add (E e)
boolean	addAll (Collection <? extends E > c)
void	clear ()
boolean	contains (Object o)
boolean	isEmpty ()
Iterator < E >	iterator ()
boolean	remove (Object o)
int	size ()

Class Collections

Method Summary	
<pre>static <T extends Comparable <? super T>> void</pre>	<pre>sort(List<T> list) Sorts the specified list into ascending order, according to the <i>natural ordering</i> of its elements.</pre>
<pre>static <T> void</pre>	<pre>sort(List<T> list, Comparator<? super T> c) Sorts the specified list according to the order induced by the specified comparator.</pre>

Interface Comparator<T>

Method Summary	
<pre>int</pre>	<pre>compare(T o1, T o2) Compares its two arguments for order.</pre>

Interface Comparable<T>

Method Summary	
<pre>int</pre>	<pre>compareTo(T o) Compares this object with the specified object for order.</pre>

Class Random

Constructor Summary	
<pre>Random()</pre>	<pre>Creates a new random number generator.</pre>

Method Summary	
<pre>int</pre>	<pre>nextInt(int n) Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.</pre>

Class Object

Method Summary	
<pre>boolean</pre>	<pre>equals(Object obj) Indicates whether some other object is "equal to" this one.</pre>
<pre>Class<?></pre>	<pre>getClass() Returns the runtime class of this Object.</pre>
<pre>int</pre>	<pre>hashCode() Returns a hash code value for the object.</pre>
<pre>String</pre>	<pre>toString() Returns a string representation of the object.</pre>

Class Class<T>

Method Summary	
<pre>String</pre>	<pre>getSimpleName() Returns the simple name of the underlying class</pre>