מס' סידורי:_____ ת.ז.:_____

פתרון הבחינה בתוכנה 1

סמסטר א', מועד א' ,תשע"ב
19/2/2012

אוהד ברזילי, דן הלפרין, ניר אטיאס, אלכסיי זגלסקי

הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות**, חלקו את זמנכם ביעילות.
- אסור השימוש בחומר עזר כלשהו, כולל מחשבונים 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 לאורך הבחינה שכל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import.
- במקומות בהם תתבקשו לכתוב מתודה (שירות), ניתן לכתוב גם מתודות עזר, אלא אם צוין במפורש אחרת.
- ניתן להוסיף הנחות לגבי אופן השימוש בשרותים המופיעים בבחינה, ובלבד שאין הן סותרות את תנאי השאלה. יש לתעד הנחות אלו כחוזה (תנאי קדם, תנאי בתר) בתחביר המקובל, שייכתב בתחילת השרות.

לשימוש הבודקים בלבד:

שאלה	א	ב	ג	ד	ה	ו	סה"כ
1	15						
2	8	8	8	8	8		
3	5	10	12				
4	3	3	3	3	3	3	

בהצלחה!

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

שאלה 1 (15 נקודות)

א. (15 נקודות) נגדיר את **<u>קבוצת המראה של מערך</u>** בתור קבוצת איברים המופיעים במערך ברצף בסדר כלשהו, ואשר מופיעים בו גם <u>בסדר הפוך</u>.

נרצה לממש את המתודה **maxMirror** המחזירה את גודל קבוצת המראה המקסימלית במערך.

למשל במערך {1, 2, 3, 8, 9, 3, 2, 1} גודל הקבוצה המקסימלית הוא 3 (עבור {1, 2, 3}).

להלן כמה דוגמאות ריצה של הפונקציה:

```
maxMirror({1, 2, 1, 4}) → 3
maxMirror({7, 1, 2, 9, 7, 2, 1}) → 2
maxMirror({1, 2, 3, 2, 1}) → 5
maxMirror({1, 1, 1}) → 3
maxMirror({1}) → 1
maxMirror({}) → 0
```

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש.
בסעיף זה אין התייחסות לסיבוכיות זמן הריצה של האלגוריתם.
אם יש לכם הנחות לגבי הקלט של הפונקציה ציינו אותן במפורש בתחביר פורמלי ככל האפשר ע"י שימוש בטענות עיצוב בעזרת חוזה, בראש הפונקציה:

```
/ **
  * @pre: arr!=null
  */

public static int maxMirror (int [] arr) {
  int max = 0;
  for(int i=0; i<arr.length; ++i)
    for(int j=i; j<arr.length; ++j) {
      int k = 0;
      while(i+k < arr.length && j-k >= 0 && arr[i+k] == arr[j-k])
        ++k;

      if(k > max)
        max = k;
    }

    return max;
}
```

שאלה 2 (40 נקודות)

בשאלה זו נדון בעצוב ובמימוש Map-Reduce - חבילת תוכנה המאפשרת לבטא בקלות משימות תוכנה שונות כהרכבה של פעולות Map ו- Reduce (כפי שיוסבר בהמשך). שימו לב כי על אף השם אין קשר (לפחות לא ישיר) ל- **java.util.Map**.

א. (8 נקודות) משימת תוכנה מסוג Map היא מחלקה המממשת את המנשק **MapTask**:

```
public interface MapTask<S,T> {
      List<T> map(List<S> input);
}
```

השרות **map** של המחלקה מקבל כארגומנט רשימה של אברים (**List<S>**) ומחזיר רשימה אחרת של אברים (**List<T>**) שהיא תוצאה של הפעלת פעולה כלשהי <u>על כל אחד</u> מאברי הרשימה המקורית. הפעולה **map** אינה משנה את רשימת הקלט שלה.

המחלקה **Squares** מממשת את המנשק **MapTask**, והשרות map שלה מקבל כקלט רשימה של מספרים שלמים ומחזיר רשימה שאבריה הם ריבועי מספרים אלו. להלן דוגמת שימוש במחלקה **Squares**:

```
List<Integer> input = Arrays.asList(1,2,3,4,5);
MapTask<Integer, Integer> s = new Squares();
List<Integer> result = s.map(input); // now result is [1, 4, 9, 16, 25]
```

ממשו את המחלקה **Squares** לפי התאור לעיל:

```
public class Squares implements MapTask<Integer, Integer>{

      @Override
      public List<Integer> map(List<Integer> input) {

            List<Integer> result = new ArrayList<Integer>();

            for (int i = 0; i < input.size(); i++) {
                  result.add(input.get(i)*input.get(i));
            }
            return result;
      }

}
```

טעויות חוזרות:

1. למרות ששימוש ב foreach יותר אלגנטי הוא לא מבטיח שמירה על הסדר
2. תלמידים אשר הגדירו את result כשדה היו צריכים לאתחל אותו (בבנאי או בראש הפונקציה), ולאפס אותו לפני השימוש הבא (map עשוי להקרא כמה פעמים על אותו עצם)
3. מימוש של MapTask<S,T> במקום MapTask<Integer,Integer>
4. הגדרת Squares גנרית, בניגוד לדוגמת השימוש למעלה
5. שימוש ב- casting או instanceof

ב. (8 נקודות) סטיב המגניב זיהה בעיה בעיצוב המנשק **MapTask** מהסעיף הקודם. סטיב טוען כי קיים שכפול קוד בין <u>מחלקות שונות אשר מממשות את המנשק</u>: המחלקה המממשת אינה מממשת רק את הלוגיקה הייחודית של ביצוע הפעולה, אלא גם צריכה לממש לוגיקה כללית המשותפת לכל משימות ה- map השונות.

המתכנתת אולגה מציעה עיצוב חלופי אשר יפתור את הבעיה.

היא מציעה להגדיר מנשק חדש בשם **MapOperation** אשר השרות op שלו הוא ביצוע של פעולה על <u>אחד</u> מאברי הרשימה. כמו כן היא מציעה להגדיר את המחלקה **Mapper**, אשר השרות **map** שלה מקבל שני ארגומנטים: רשימת הקלט ועצם ממחלקה המממשת את **MapOperation**.

```
public interface MapOperation<S,T> {
      T op(S element);
}
```

להלן דוגמת שימוש במחלקה **Squares**, לפי העיצוב שהציעה אולגה:

```
List<Integer> input = Arrays.asList(1,2,3,4);
Mapper<Integer, Integer> m = new Mapper<Integer, Integer>();
MapOpetarion<Integer, Integer> op = new Squares();
List<Integer> result = m.map(input, op); // now result is [1, 4, 9, 16]
```

ממשו את המחלקות **Squares** ו- **Mapper** (בעמוד הבא) לפי העיצוב שהציעה אולגה.

ממשו את המחלקה **Mapper** כך שתענה על העיצוב לעיל:

```
public class Mapper<S,T> {

	public List<T> map(List<S> input, MapOpetarion<S, T> task){

		List<T> result = new ArrayList<T>();

		for (int i = 0; i < input.size(); i++) {
			result.add(task.op(input.get(i)));
		}
		return result;
	}
}
```

ממשו את המחלקה **Squares** כך שתענה על העיצוב לעיל:

```
public class Squares implements MapOpetarion<Integer, Integer> {

	@Override
	public Integer op(Integer element) {
		return element*element;
	}
}
```

טעויות חוזרות:

1. מימוש הפונקציה int op(int) במקום שימוש ב Integer
2. הגדרת Squares גנרית בניגוד לדוגמת השימוש

ג. (8 נקודות) משימת תוכנה מסוג Reduce (צמצום) היא מחלקה המממשת את המנשק **ReduceTask**:

```
public interface ReduceTask<T> {
	T reduce(List<T> input);
}
```

השרות **reduce** של המחלקה מקבל כארגומנט רשימה של אברים (**List<T>**) ומחזיר ערך מטיפוס T שהוא תוצאה של הפעלת פעולה כלשהי על כל אחד מאברי הרשימה המקורית. שימו לב כי בשונה מפעולת ה Map, פעולת ה **Reduce** מצמצמת את הרשימה לכדי איבר אחד מטיפוס T. גם הפעולה **reduce** אינה משנה את רשימת הקלט שלה.

המחלקה **Summarizer** מממשת את המנשק **ReduceTask**, והשרות reduce שלה מקבל כקלט רשימה של מספרים שלמים ומחזיר את סכומם. להלן דוגמת שימוש במחלקה **Summarizer**:

```
List<Integer> input = Arrays.asList(1,2,3,4,5,6);
ReduceTask<Integer> s = new Summarizer();
Integer result = s.reduce(input); // now result is 21
```

ממשו את המחלקה **Summarizer** לפי התאור לעיל:

```
public class Summarizer implements ReduceTask<Integer> {

	@Override
	public Integer reduce(List<Integer> input) {

		Integer result = 0;
		for (Integer integer : input) {
			result += integer;
		}
		return result;
	}

   }
```

ד. (8 נקודות) סטיב המגניב מגלה שבמנשק ReduceTask מהסעיף הקודם, מסתתר שכפול קוד דומה לזה שהיה ב- MapTask (בסעיף ב'). סטיב ניסה להפעיל את העיצוב החלופי שהציעה אולגה עבור Map גם עבור Reduce באופן הבא: הוא מגדיר מנשק חדש בשם **ReduceOperation** שהפעולה **op** שלו מייצגת את הוספת האיבר הבא ברשימה (**nextElement**) לערך הרשימה "המצומצם" (accum, שהולך ונבנה במהלך הקריאות ל-**ReduceOperation**):

```
public interface ReduceOpetarion<T> {
	T op(T accum, T nextElement);
}
```

אולגה מעירה שהעיצוב החלופי מתאים לפעולות op שהן חילופיות (קומוטטיביות), כלומר אפשר לבצע אותן על אברי הרשימה בכל סדר שהוא.

ממשו את המחלקה **Reducer**, אשר יש לה שרות בשם **reduce**, לפי העיצוב החלופי. שימו לב להבדלים בין החתימה של **map** מסעיף ב' לחתימת **reduce** בסעיף זה:

```
public class Reducer<T> {

	public T reduce(List<T> input, ReduceOpetarion<T> task, T initValue){

		T result = initValue;

		for (int i = 0; i < input.size(); i++) {
			result = task.op(result, input.get(i));
		}
		return result;
	}
}
```

טעויות חוזרות:
תלמידים רבים לא קלטו את ערך ברירת המחדל כחלק מהחתימה – אולם במקרה זה היה צריך לאתחל את הערך המוחזר בצורה כלשהי. הצורות הבאות היו שגויות:
1. אתחול ל-0
2. אתחול על ידי new T()
3. אתחול על ידי new Object()
4. אתחול על ידי האיבר הראשון (אף על פי שבמקרה של סכום זה מתאים, זה לא מתאים במקרה הכללי).
תלמידים אשר אתחלו על ידי null היו צריכים להתמודד עם המקרה שמועבר null לפונקציה op.

ממשו את המחלקה **Summarizer** לפי התאור לעיל:

```
public class Summarizer implements ReduceOpetarion<Integer> {

	@Override
	public Integer op(Integer e1, Integer e2) {
		return e1 + e2;
	}
}
```

ה. (8 נקודות) המנשק **Student** מייצג תלמיד:

```
public interface Student {
	int getGrade();

	// more methods...
}
```

במחלקה **CourseUtils** יש שרותי עזר שימושיים לעבודה במערכת תוכנה העוסקת בתלמידים ובקורסים. המחלקה עושה שימוש בתשתית Map-Reduce שהצגנו בסעיפים הקודמים.

ממשו את השרות **getMaxGrade** במחלקה **CourseUtils**. השרות מקבל רשימה של תלמידים ומחזיר את הציון הגבוה ביותר. אם מועברת רשימה ריקה מוחזר הציון -1.

שימו לב: בסעיף זה יש לממש את כל מבני העזר כחלק מהשרות **getMaxGrade**. כמו כן, גם אם לא עניתם על אחד הסעיפים הקודמים ניתן לענות על שאלה זו כאילו המחלקות **Mapper** ו-**Reducer** ממומשות כהלכה.

```
public class CourseUtils {

	public static final Mapper<Student, Integer> mapper =
                                                new Mapper<Student,Integer>();

	public static final Reducer<Integer> reducer =
                                                new Reducer<Integer>();

	/**
	 * @pre students != null
	 * @post students.size() == 0 $implies $ret == -1
	 */
	public static int getMaxGrade(List<Student> students){

		List<Integer> grades = mapper.map(students, new MapOpetarion<Student, Integer>() {
			public Integer op(Student s) {
				return s.getGrade();
			}
		});

		int maxGrade = reducer.reduce(grades, new ReduceOpetarion<Integer>() {
			public Integer op(Integer e1, Integer e2) {
				return Math.max(e1, e2);
			}
		} , -1);

		return maxGrade;
	}
```

שאלה 3 (27 נקודות)

סולם מילים הוא סדרה של מילים חוקיות בשפה שכולן באותו אורך וכל מילה נבדלת מקודמתה באות אחת בדיוק.

דוגמא לסולם מילים באנגלית:

BIRD
BARD
BARN
DARN
DARE
DARK
LARK

בשאלה זו נעסוק בשתי גרסאות של תכנית הבודקת האם סדרת מילים באנגלית היא סולם מילים. גרסה אחת קוראת את המילים מקובץ ואילו הגרסה השנייה מספקת ממשק גרפי.

א. (5 נקודות) השלימו את מימוש המחלקה **Ladder** (בעמוד הבא) הבודקת את חוקיות סולם מילים כלשהו. למחלקה פונקציה ציבורית אחת, **addToLadder**, המקבלת מילה ובודקת אם הוספתה לסולם המילים הנוכחי שומרת על חוקיות הסולם. אם ההוספה חוקית, המילה נוספת לסולם והפונקציה מחזירה true, אחרת הפונקציה מתעלמת מהקלט ומחזירה **false**. השרות **isEnglishWord** מקבל מחרוזת ומחזיר **true** אם זו מילה באנגלית.

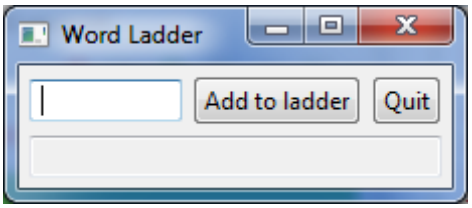
רמז: השרות **public char charAt(int index)** של המחלקה **String** מחזיר את התו במקום ה **index**.

ב. (10 נקודות) ממשו את הפונקציה הסטטית **isWordLadder** המקבלת כקלט שם קובץ, פותחת אותו לקריאה, קוראת ממנו סדרת מילים ומחזירה true אם ורק אם הסדרה מהווה סולם מילים. יש להניח כי בקובץ הקלט כל שורה מכילה בדיוק מילה אחת.

```
public static boolean isWordLadder(String filename) throws FileNotFoundException {
		Scanner file = new Scanner(new File(filename));
		Ladder ladder = new Ladder();

		while(file.hasNext())
			if(!ladder.addToLadder(file.nextLine()))
				return false;

		return true;
}
```

```
public class Ladder {

	private String last;

	private static boolean isEnglishWord(String s) {...}

	private static boolean onlyOneDiff(String s1, String s2) {
		int length = s1.length();
		int diffs = 0;
		if (s2.length() != length)
			return false;
		for (int i = 0; i < length; i++)
			if (s1.charAt(i) != s2.charAt(i))
				++diffs;
		return diffs == 1;
	}

	public boolean addToLadder(String word) {
		if(word == null || word.isEmpty() || !isEnglishWord(word))
			return false;

		if(last == null || onlyOneDiff(last, word)) {
			last = word;
			return true;
		}

		return false;
	}

}
```

ג. (12 נקודות) ביל מימש מנשק משתמש גרפי (GUI) עבור מחלקת הסולם. המנשק מאפשר למשתמש להקליד מילה בתיבת הקלט (התיבה העליונה). לאחר מכן, המשתמש מקיש על הכפתור "Add to ladder". אם המילה ממשיכה סולם מילים חוקי (או מתחילה סולם מילים) יופיע הכיתוב "So far so good" בתיבת הטקסט התחתונה והמילה תתווסף לסולם המילים. אחרת, יופיע הכיתוב "Illegal word".

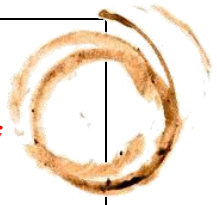


ביל הדפיס את קוד המחלקה שמימש כדי להראות לאומה, אך בדרך עצר במטבחון שם הניח את כוס הקפה על התדפיס. השלימו את קוד המחלקה (במקומות המסומנים בכתמי קפה).

רמז: ל-Widget מסוג Text קיימת המתודה: `public void setText(String string)` אשר קובעת את תוכן ההודעה שה-Widget מציג.

```
public class GUILadder {

  private Shell shell = null;

  private Ladder ladder = new Ladder();

  // this method builds the GUI including the listeners
  private void createShell() {
    // System.out.println("Got into createShell");
    // create the GUI
    shell = new Shell();
    shell.setText("Word Ladder");
    shell.setLayout(new GridLayout(3, false));// layout manager:
                                              // a grid with 3
                                              // unequal columns

    // here the user will enter the next word in the ladder
    final Text inText = new Text(shell, SWT.BORDER);
    inText.setLayoutData(new GridData(SWT.FILL, // horizontal alignment
                         SWT.CENTER,  // vertical alignment
                         true,        // grab horizontal space
                         false));     // don't grab vertical space

    // pressing the next button sends the word to the program for checking
    Button buttonAdd = new Button(shell, SWT.NONE);
    buttonAdd.setText("Add to ladder");
   buttonAdd.setLayoutData(new GridData(SWT.RIGHT, SWT.CENTER, false, false));


    Button buttonQuit = new Button(shell, SWT.NONE);
    buttonQuit.setText("Quit");
    buttonQuit.setLayoutData(new GridData(SWT.RIGHT, SWT.CENTER, false, false));

      buttonQuit.addSelectionListener(new SelectionAdapter() {
        @Override
        public void widgetSelected(SelectionEvent e) {

            shell.dispose()

        }
      });

      // here the program tells the user whether the last word they entered
      // continues the ladder
      final Text outText = new Text(shell, SWT.BORDER|SWT.READ_ONLY);
      outText.setLayoutData(new GridData(SWT.FILL, // horizontal alignment
                         SWT.CENTER,            // vertical alignment
                         true,                // grab horizontal space
                         false,               // don't grab vertical space
                         3,    // horizontal span
                         1));  // horizontal span
```

```
        buttonAdd.addSelectionListener(new SelectionAdapter() {
          @Override
          public void widgetSelected(SelectionEvent e) {
                        String word = inText.getText();
                        if (ladder.addToLadder(word)) {
                              outText.setText("So far so good");
                        } else {
                              outText.setText("Illegal word");
                        }
          }
        });

        shell.pack(); // causes the layout manager to lay out the shell
        shell.open(); // opens the shell on the screen
    }


    public void showLadderUI() {
      Display display = Display.getDefault();
      createShell();

      // the GUI event loop
      while (!shell.isDisposed()) {
        if (!display.readAndDispatch())
          display.sleep();
      }
      display.dispose();
    }

}
```



שאלה 4 (18 נקודות)

הסעיפים בשאלה זו מתייחסים לשלוש המחלקות הבאות:

```
public class A {
    public int foo(int a, int b) throws IOException{
            return a;
    }
}
```

```
public class B extends A {
      **************
}
```

```
public class C {
      public static void main (String[] args){
            A b = new B();
            try {
                  System.out.println(b.foo(5,7));
            } catch (IOException e){}
      }
}
```

בכל אחד מהסעיפים הבאים מוחלפת שורת הכוכביות בקטע קוד. הינכם מתבקשים לציין מהו הפלט של התכנית עבור כל מקרה. במידה ולדעתכם אין פלט לתכנית מכיוון שאינה עוברת קומפילציה או מכיוון שקיימת בעיה בזמן הריצה (זריקת חריג) הסבירו מה הבעיה, פתרון ללא הסבר לא יזכה בנקודות.

סעיף א' (3 נקודות)

```
public int foo(int[] a){return 2;}
```

תשובה:

```
5
we have done here overloading and the original A.foo was called.
```

סעיף ב' (3 נקודות)

```
int foo(int c, int a){return a;}
```

תשובה:

```
compilation error
Cannot reduce the visibility of the inherited method from A –
Supplier cannot supply less than promised in contract (lesser
visibility).
```

סעיף ג' (3 נקודות)

```
public int foo(int c, int d) throws Exception{return c;}
```

תשובה:

```
compilation error
Exception Exception is not compatible with throws clause in
A.foo(int,int) – supplier cannot throw a more general Exception
then the one in the contract.
```

סעיף ד' (3 נקודות)

```
public float foo(int a, int b){return 3;}
```

תשובה:

```
compilation error
The return type is incompatible with A.foo(int,int) – can't
overload with the same signature but a different return type;
```

סעיף ה' (3 נקודות)

```
public float foo(float a, int b){return a;}
```

תשובה:

```
5
we have done here overloading and the original A.foo was called.
```

סעיף ו' (3 נקודות)

```
public int foo(int a, int c) throws RuntimeException{return c;}
```

תשובה:

```
7
overridden A.foo with B.foo and it was called. The overriding is
legal since RunTimeException is an unchecked Exception
```

ושוב, בהצלחה