

# פתרון בחינה בתוכנה 1

סמסטר ב', מועד ב', תשע"א

14/09/2011

ליאור וולף, מתי שמרת, הדס צור, אסף זריצקי

## 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 לאורך השאלה 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 כחוזה (תנאי קדם, תנאי בתר) בתחביר המקובל, שייכתב בתחילת השרות.

לשימוש הבודקים בלבד:

שאלה	א	ב	ג	סה"כ
1				
2				
3				
4				
בחינה				

## בהצלחה!

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

## שאלה 1 (20 נקודות)

א. (15 נקודות) ממשו את השרות `maxPalindrome` אשר בהינתן מחרוזת (`string`) מחזיר את אורך הפלינדרום הארוך ביותר במחרוזת. תזכורת: פלינדרום הוא רצף שניתן לקרוא משני הכיוונים ללא שינוי בתוצאה. (לדוגמה: `aba`, `radar`, "ילד כותב בתוך דלי")

להלן כמה דוגמאות:

`maxPalindrome("aba")` → 3  
`maxPalindrome("k")` → 1  
`maxPalindrome("")` → 0  
`maxPalindrome("myradarisjamed")` → 5  
`maxPalindrome("nopalindrome")` → 1

בסעיף זה אין התייחסות לסיבוכיות זמן הריצה של האלגוריתם. אם יש לכם הנחות לגבי הקלט לפונקציה ציינו אותן במפורש בתחביר פורמלי ככל האפשר ע"ש שימוש בטענות עיצוב בעזרת חוזה:

```
/**
 * @pre text != null
 *
 */
public static int maxPalindrome(String text) {
    int maxLength = 0;
    for (int i = 0; i < text.length(); i++) {
        for (int j = text.length(); j - i > maxLength; j--) {
            if (isPalindrome(text.substring(i, j))) {
                maxLength = j - i;
            }
        }
    }
    return maxLength;
}

private static boolean isPalindrome(String text) {
    StringBuilder reversed = new StringBuilder(text);
    reversed.reverse();
    return reversed.toString().equals(text);
}
```

ב. (5 נקודות) נסחו במדויק את תנאי האחר של הפונקציה שמימשתם תוך שימוש בכמתי לכל  $(\forall)$  וקיים  $(\exists)$ :

```

    נגדיר את הפרדיקט palindrome(text, len) שמשוערך ל-true אם קיים
    פלינדרום באורך len ב-text.
    palindrome(text, len)  $\equiv$ 
     $\exists i. (0 \leq i \leq \text{text.length}() - \text{len}) \ \&$ 
     $(\forall k. 0 \leq k < \text{len} \rightarrow \text{text.charAt}(i+k) = \text{text.charAt}(i+\text{len}-1-k))$ 
    נגדיר את תנאי האחר בעזרת הפרדיקט:
Boundaries:  $0 \leq \$ret \leq \text{text.length}()$ 
Palindrome: palindrome(text, $ret)
Max:  $\forall m. m > \$ret \rightarrow \text{!palindrome}(\text{text}, m)$ 

```

## שאלה 2 (40 נקודות)

נתונים שני המנשקים Stack ו- Maxable:

```

public interface Stack<T> {
    /** @pre: !isEmpty() */
    T top();

    /** @pre: element != null */
    void push(T element);

    /** @pre: !isEmpty() */
    void pop();

    boolean isEmpty();
}

public interface Maxable<T extends Comparable<T>> {
    /** returns the maximal element */
    T max();
}

```

אומה מעוניינת לפתח את MaxStack, מחסנית מקסימלית, אשר מממשת את שני המנשקים:

```

public class MaxStack<T extends Comparable<T>>
    implements Stack<T>, Maxable<T>

```

ביל מציע לאומה מימוש יעיל במיוחד למחסנית המקסימלית אשר מממש את כל פעולות המנשקים בזמן  $O(1)$ , כלומר אין צורך לעבור על כל אברי המחסנית לצורך שום פעולה. ההצעה של ביל היא כזו: בכל פעם שמתבצעת פעולת push, נדחוף לא רק את האיבר החדש, אלא גם את המקסימום החדש של המחסנית (אולי זהה למקסימום הישן). כך שבעצם האיבר העליון במחסנית יהיה ה-max שלה בכל רגע נתון, ואיבר ה-top האמיתי יהיה האיבר שאחריו במחסנית. כלומר, במחסנית עם n אברים, ישמרו למעשה 2n אברים לסרוגין: ה-max העדכני ו'איבר אמיתי'.

אומה כל כך מתרגשת מהצעתו של ביל, שהיא כבר רוצה לממש את המחסנית, אבל לא יודעת איך לממש את מבנה הנתונים שיחזיק את המחסנית (מערך / רשימה / Collection). ביל מציע לה לדחות את ההחלטה על המימוש תוך שימוש בתבנית עיצוב ה-Bridge שנלמדה בהרצאה. ביל עוזר לאומה, ומממש בנאי אשר מקבל מימוש של מחסנית ושומר אותו בשדה פנימי.

א. (15 נקודות) עיזרו לאומה להשלים את מימוש המחסנית לפי הצעת ביל (בעמוד הבא):

```

public class MaxStack<T extends Comparable<T>>
    implements Stack<T>, Maxable<T> {

    private Stack<T> stackImpl;

    /**
     * @pre: stackImpl != null && stackImpl.isEmpty()
     */
    public MaxStack(Stack<T> stackImpl) {
        this.stackImpl = stackImpl;
    }
}

```

```
@Override
// @pre !isEmpty()
public T max() {
    return stackImpl.top();
}

@Override
// @pre element != null
public void push(T element) {
    T max = isEmpty() ? element : stackImpl.top();
    stackImpl.push(element);
    stackImpl.push(element.compareTo(currentMax) > 0 ?
        element : max);
}

@Override
// @pre !isEmpty()
public void pop() {
    stackImpl.pop();
    stackImpl.pop();
}

@Override
// @pre !isEmpty();
public T top() {
    T currentMax = stackImpl.top();
    stackImpl.pop();
    T top = stackImpl.top();
    stackImpl.push(currentMax);
    return top;
}

@Override
public boolean isEmpty() {
    return stackImpl.isEmpty();
}
```

ב. **(10 נקודות)** אומה מגלה בעיה עם המימוש מסעיף א', אברי המחסנית עשויים להשתנות לאחר הכנסתם למחסנית (אם הם mutable), ואולם מכיוון שהאיבר המקסימלי חושב לפי ערכו של כל איבר בעת ההכנסה הערך `max()` עשוי להיות לא מעודכן (למשל: ערכו של האיבר המקסימלי יכול לרדת מתחת לערכו של איבר אחר בין הזמן שבו הוכנס למחסנית והזמן שבו בוצע ה-`max()`).

אומה מבינה שקיימות מספר דרכים לטפל בבעיה ומחליטה להוסיף למחלקה `MaxStack` שירות שיעזור במימוש הפתרונות האפשריים. אומה מחליטה להוסיף למחסנית את השרות `elementChanged` שתפקידו להחזיר את המחסנית למצב תקין. פתרונות אפשריים לבעיה יוכלו לרשת מהמחלקה `MaxStack` ולעשות שימוש בשירות זה (ראו סעיף הבא).

השלימו את מימוש השרות `elementChanged` של המחלקה `MaxStack`. בסעיף זה אין מגבלה על סיבוכיות זמן הריצה (או הזיכרון) של המימוש.

```
public class MaxStack<T extends Comparable<T>>
    implements Stack<T>, Maxable<T> {

    // same as in previous section

    protected void elementChanged() {
        Deque<T> elements = new ArrayDeque<T>();
        while (!isEmpty()) {
            elements.addFirst(top());
            pop();
        }
        for (T element : elements) {
            push(element);
        }
    }
}
```

ג. (15 נקודות) ביל מציע לאומה להוסיף למחסנית מנגנון שיאפשר קריאה אוטומטית לשרות elementChanged בכל פעם שאיבר משתנה. ביל מציע להשתמש בתבנית העיצוב Observer: המחסנית תממש את המנשק Observer ואילו אברי המחסנית יממשו את המנשק Observable.

---

```
public interface Observable {  
    void addObserver(Observer observer);  
    void removeObserver(Observer observer);  
}
```

---

```
public interface Observer {  
    void update(Observable observable);  
}
```

---

המחסנית תרשם לקבלת עדכונים על שינויים בערכי אבריה, בכך שתפעיל את השרות addObserver על כל אחד מהאברים ותעביר את עצמה כארגומנט לשרות. רישום זה יתבצע בעת הוספת האיבר למחסנית. בכל פעם שאיבר במחסנית יתעדכן הוא יודיע זאת לכל הצופים שלו ע"י הפעלת השרות update על כל אחד מהם. כאשר איבר מוצא מהמחסנית (ע"י pop()) על המחסנית להסיר את עצמה מרשימת ה observers שלו.

ממשו את המחלקה AutoUpdatedMaxStack היורשת מהמחלקה MaxStack כדי לתמוך במנגנון החדש. ניתן להניח כי כל אברי המחסנית כבר מממשים כהלכה את המנשק Observable.

שימו לב, השימוש באופרטור '&' בחתימת המחלקה מציין כי הטיפוס הגנרי T מממש גם את Comparable וגם את Observable.

```
public class AutoUpdatedMaxStack<T extends Comparable<T> & Observable>
    extends MaxStack <T> implements Observer {

    public AutoUpdatedMaxStack(Stack<T> stackImpl) {
        super(stackImpl);
    }

    @Override
    public void push(T element) {
        element.addObserver(this);
        super.push(element);
    }

    @Override
    public void pop() {
        T top = top();
        top.removeObserver(this);
        super.pop();
    }

    @Override
    public void update(MyObservable observable) {
        elementChanged();
    }
}
```



## שאלה 3 (30 נקודות)

במהלך הסמסטר הצגנו (בשיעור GUI) דפדפן אשר גולש לתוצאת החיפוש הראשונה בגוגל (לפי הערך שהוזן בתיבת הטקסט). ברצוננו להוסיף לדפדפן זה מנגנון אשר יזכור את היסטוריית החיפושים.

א. (15 נקודות) בשלב ראשון נגדיר את הפעולות אשר מנגנון ההיסטוריה אמור לתמוך בהן. להלן הממשק History המתאר את הפעולות:

```
/**
 * Maintains a history of strings. The elements in the history are
 * kept in the order they were added, thus newer element will appear
 * before older ones. There are no duplicates in the history. If the
 * same query is added twice only one copy is kept (the newer one).
 */
public interface History {
    /**
     * Add text to the history.
     */
    void add(String text);

    /**
     * Retrieve the entire history.
     */
    List<String> getHistory();

    /**
     * Retrieve only elements in the history which starts with the
     * given prefix. The empty string is a prefix of all strings.
     */
    List<String> getHistory(String prefix);

    /**
     * Check whether the history contains elements
     */
    boolean isEmpty();
}
```

- add מוסיף את המחרוזת text להיסטוריית החיפושים. מחרוזת תופיע בהיסטורית החיפושים בדיוק פעם אחת. אם מחרוזת כבר הופיעה בעבר, יש לשמור את המופע האחרון.
- getHistory מחזיר את רשימת כל המחרוזות שבהיסטוריה. מחרוזות יסודרו מהחדשה ביותר (אינדקס 0) ועד לישנה ביותר.
- getHistory(String prefix) מחזיר את רשימת כל המחרוזות שבהיסטוריה המתחילות במחרוזת prefix. מחרוזות יסודרו מהחדשה ביותר (אינדקס 0) ועד לישנה ביותר. המחרוזת הריקה מהווה תחילית של כל מחרוזת.
- isEmpty בודק האם ההיסטוריה ריקה.

ממשו את המחלקה SimpleHistory המממשת את הממשק History. ניתן להגדיר מבני עזר או שירותים חדשים לצורך המימוש:

```
public class SimpleHistory implements History {
    private final List<String> history;

    public SimpleHistory() {
        history = new ArrayList<String>();
    }

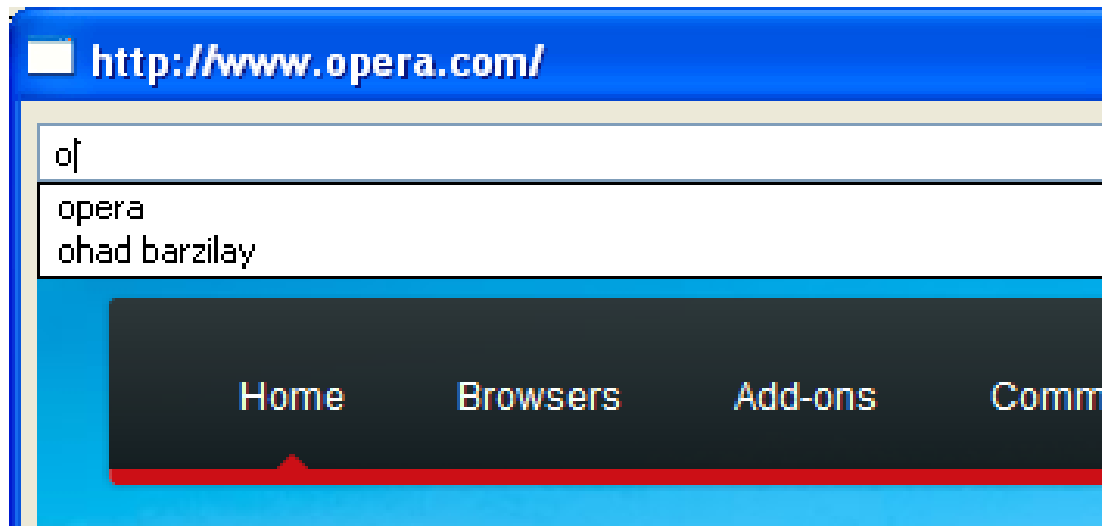
    @Override
    public void add(String text) {
        if (history.contains(text)) {
            history.remove(text);
        }
        history.add(0, text);
    }

    @Override
    public List<String> getHistory() {
        return getHistory("");
    }

    @Override
    public List<String> getHistory(String prefix) {
        String lowerCase = prefix.toLowerCase();
        List<String> result = new ArrayList<String>();
        for (String str : history) {
            if (str.toLowerCase().startsWith(lowerCase)) {
                result.add(str);
            }
        }
        return result;
    }

    @Override
    public boolean isEmpty() {
        return history.isEmpty();
    }
}
```

ב. (15 נקודות) ברצוננו לשלב את רכיב ההיסטוריה שממשנו בסעיף א' בדפדפן שהוצג בכיתה. כאשר המשתמש יקליד את השאלתה תפתח חלונית ובה יוצגו כל השאלות הקודמות המתחילות בטקסט שכבר הוקלד. לדוגמא: אם חיפשנו את "ohad barzilay", את "java" ולבסוף "opera" וכעת אנחנו מקלידים שאלתת חיפוש חדשה המתחילה ב-o תוצג חלונית ובה שתי השאלות המתאימות לתחילית זו (איור 1).



איור 1: חלונית היסטוריה

לפניכם קוד המחלקה GoogleBrowserWithHistory המממש את התכונות נדרשות. ביישום זה חלונית ההצעות ממומשת בעזרת Shell נוסף, אשר מכיל טבלה (Table), ובה ההצעות הרלוונטיות, שמהן יכול המשתמש יכול לבחור בעזרת העכבר. בכל פעם שהמשתמש לוחץ על הכפתור "I'm feeling lucky" מתבצע חיפוש, והשאלתה הנוכחית מתווספת להיסטורית החיפושים.

לצערנו, בעת הבאת המבחן לדפוס נשפכה כוס קפה על הבחינה וחלק מהקוד נמחק. עליכם להשלים את הקוד החסר במלבנים המיועדים לכך והמסומנים בכתמי קפה. מקום זה אמור להספיק לכל הקוד הדרוש, אולם אם ברצונכם להוסיף קוד במקומות אחרים פרט למלבנים המיועדים לכך, הוסיפו אותו בעמוד האחרון וציינו במדויק היכן קוד זה אמור להשתלב.

שימו לב: כמעט כל הקוד הנדרש לשילוב ההיסטוריה בדפדפן כבר ממומש. הקוד החסר עוסק רובו ככולו בשילוב בין הפונקציונליות שמימשתם בסעיף א' וקוד הדפדפן שראיתם בהרצאה. קראו בעיון את הקוד שלפניכם והשלימו את הנדרש בהתאם לקוד הקיים.

```
public class GoogleBrowserWithHistory {
    private Shell shell = null;
    private Shell popupShell;
    private Text text = null;
    private Browser browser = null;
    private Table table;
    private static Display display = Display.getDefault();;
    private History history = new SimpleHistory();

    public static void main(String[] args) {
        final GoogleBrowserWithHistory app =
            new GoogleBrowserWithHistory();
        app.createShell(display);
        while (!app.shell.isDisposed()) {
            if (!display.readAndDispatch())
                display.sleep();
        }
        display.dispose();
    }

    // create the GUI
    private void createShell(Display display) {
        shell = new Shell();
        shell.setText("Browser Example");
        shell.setLayout(new GridLayout(2, false));

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

        Button button = new Button(shell, SWT.NONE);
        button.setText("I'm feeling lucky");
        button.setLayoutData(new GridData(SWT.RIGHT, SWT.CENTER,
            false, false));

        button.addSelectionListener(new SelectionAdapter() {
            @Override
            public void widgetSelected(SelectionEvent e) {
                try {
                    String query = text.getText();
                    history.add(query);

                    String url = null;
                    url = search(query);
                    if (url != null) {
                        shell.setText(url);
                        browser.setUrl(url);
                    }
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
        });
    }
}
```

```

browser = new Browser(shell, SWT.NONE);
browser.setLayoutData(new GridData(SWT.FILL, SWT.FILL,
                                   false, true, 2, 1));

popupShell = createPopupShell(display);
table = new Table(popupShell, SWT.SINGLE);

table.addSelectionListener(new SelectionAdapter() {
    @Override
    public void widgetDefaultSelected(SelectionEvent e) {
        handleTableSelection();
    }
});

```

**// listen to keyboard events**

```

text.addKeyListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        handleKeyReleased();
    }
});

```

```

shell.pack();
shell.open();
}

```

**// Handles key strokes in the text widget.**

```

private void handleKeyReleased() {
    String prefix = text.getText();
    if (prefix.isEmpty()) {
        popupShell.setVisible(false);
        return;
    }
    List<String> matching = null;
    matching = history.getHistory(prefix);

```

```

    if (matching.isEmpty()) {
        popupShell.setVisible(false);
        return;
    }
    table.setItemCount(matching.size());
    TableItem[] tableItems = table.getItems();
    for (int i = 0; i < tableItems.length; i++) {
        tableItems[i].setText(matching.get(i));
    }
    setPopupBounds();
    popupShell.setVisible(true);
}

```

```
private void setPopupBounds() {
    TableItem item0 = table.getItem(0);
    Rectangle textBounds = display.map(shell, null,
                                     text.getBounds());
    Point tableSize = table.computeSize(SWT.DEFAULT,
                                       SWT.DEFAULT, true);
    popupShell.setBounds(textBounds.x,
                        textBounds.y + textBounds.height,
                        textBounds.width,
                        tableSize.y - item0.getBounds().height);
}

// create the history popup shell
private static Shell createPopupShell(Display display) {
    final Shell popupShell = new Shell(display, SWT.ON_TOP);
    popupShell.setLayout(new FillLayout());
    return popupShell;
}

// Handle double-click on a table item
private void handleTableSelection() {
    String str = table.getSelection()[0].getText();
    text.setText(str);
    text.setSelection(str.length());
    popupShell.setVisible(false);
}

// Perform the actual search using Google's JSON API
private String search(String query)
    // same as in class ...
}
}
```

**שאלה 4 (10 נקודות)**

ענו בקצרה על שתי השאלות הבאות (תשובה מילולית בלבד. אין צורך לכתוב קוד):

א. (5 נקודות) כאשר מגדירים מחלקה אנונימית A בתוך פונקציה f, מדוע לא ניתן להשתמש במשתנים המקומיים של f במחלקה האנונימית?

משתנה מקומי מוקצה על המחסנית ו"מת" בסוף פעולת השירות f, בעוד אובייקטים מוקצים על

ה-Heap וחיים מעבר לקריאה לשירות. לפיכך האובייקט של המחלקה A יכול להתקיים מעבר לזמן

של המשתנים המקומיים ב-f.

ב. (5 נקודות) עבור פרמטר גנרי T, הסבירו מדוע התחביר `new T()` אינו חוקי?

כל יצירה של אובייקט מלווה בקריאה לבנאי. כאשר אנו מנסים ליצור אובייקט מטיפוס פרמטרי אין

לקומפיילר יכולת לוודא שאכן קיים בנאי מתאים בטיפוס האקטואלי.