מס' סידורי:_____ ת.ז.:_____

פתרון הבחינה בתוכנה 1

סמסטר א' וב', מועד ב' ,תש"ע
18/08/2010

דן הלפרין, אוהד ברזילי, אסף זריצקי, מתי שמרת
ליאור וולף, ליאור שפירא, רובי בוים

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.

- בכל השאלות 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 בשרותים המופיעים בבחינה, ובלבד שאין הן סותרות את תנאי השאלה. יש לתעד הנחות אלו כחוזה (תנאי קדם, תנאי בתר) בתחביר המקובל, שיכתב בתחילת השרות.

לשימוש הבודקים בלבד:

שאלה	א	ב	ג	ד	סה"כ
1	15				15
2	10	15	15		40
3	7	7	6		20
4	5	8	8	4	25

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)

ממשו את השרות maxRun אשר בהינתן מחרוזת (string) מחזיר את הריצה (run) הארוכה ביותר במחרוזת. **ריצה** מוגדרת בתור מספר הפעמים שבו מופיע אותו התו ברצף (השרות מחזיר את אורך הרצף הארוך ביותר במחרוזת).

להלן כמה דוגמאות:

maxRun("hoopla") → 2
maxRun("$") → 1
maxRun("abbCCCddBBBxx") → 3
maxRun("") → 0

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש:

```
public static int maxRun(String str) {

	int result = 0;
	for (int i = 0; i<str.length(); i++){
		int currentBlock = 1;
		for (int j = i+1; j<str.length(); j++){
			if(str.charAt(j) == str.charAt(j-1))
				currentBlock++;
			else
				break;
		}
		if (currentBlock > result)
			result = currentBlock;
	}
	return result;
}
```

שאלה 2 (40 נקודות)

א. רב-מפה (multimap) הוא מנשק דומה למפה, אשר בשונה ממפה מאפשר לשמור כמה ערכים לכל מפתח. השימוש ברב מפה שימושי עבור מיפויים שאינם חד ערכיים. לדוגמא:

- "עובד" עם כמה "כלי רכב"
- "לקוח" עם כמה "חשבונות בנק".

```
import java.util.Set;

public interface MultiMap<K,V> {

    public void put(K key, V value);

    public Set<V> get(K key);

}
```

ממשו מחלקה המממשת את המנשק רב-מפה לעיל (ניתן להיעזר במחלקות המתועדות בנספח):

```
public class SimpleMultiMap<K,V> implements MultiMap<K, V> {

      Map<K, Set<V>> rep = new TreeMap<K, Set<V>>();

      @Override
      public Set<V> get(K key) {
            return rep.get(key);
      }

      @Override
      public void put(K key, V value) {
            Set<V> curr = rep.get(key);
            if (curr == null){
                  curr = new TreeSet<V>();
            }
            curr.add(value);
            rep.put(key, curr);
      }
}
```

ב. מפה מתויגת (TaggedMultiMap) הוא מנשק אשר מרחיב את מנשק המפה (MultiMap), המאפשר להכניס לכל זוג עוד עצם מטיפוס כלשהו (tag), הקשור במובן כלשהו לזוג הממופה. מפה מתויגת היא שימושית כאשר רוצים להוסיף מידע שאינו שייך ספציפית לא למפתח (key) ולא לערך (value), וכן כדי להבחין בין מספר ערכים הממופים לאותו מפתח.

לדוגמא: במנוע חיפוש ניתן ליצור מיפוי בין "שאילתה" ובין "עמודי אינטרנט", עם תגיות מטיפוס Double, אשר יהיה מספר בין 0 ל-1 שמייצג את מידת הרלוונטיות של העמוד לשאילתה.

להלן מנשק הטיפוס TaggedMultiMap:

```
import java.util.Set;

public interface TaggedMultiMap<K,V,T> extends MultiMap<K,V> {

      public void put(K key, V value, T tag);

      public T getTag(K key, V val);
      public T getTag(Entry<K, V> entry);
      public Set<T> getTags(K key);

      public interface Entry<K,V> {
            K getKey();
            V getValue();
      }
}
```

שימו לב כי ניתן להשתמש בשרות put של הטיפוס TaggedMultiMap גם עבור key ו-value שהוכנסו כבר ל map ללא tag (למשל ע"י שרות ה- put של MultiMap). במקרה כזה, השרות מוסיף למיפוי את התגית (או, אם כבר הייתה תגית למיפוי, מחליף אותה). עבור ערכים (values) ללא תגית מחזיר השרות getTag את הערך null. אם עבור מפתח (key) מסוים לא הוכנסו תגיות כלל, מחזיר השרות getTags את הקבוצה הריקה.

ממשו מחלקה המממשת את המנשק TaggedMultiMap:

```
public class SimpleTaggedMultiMap<K, V, T> extends SimpleMultiMap<K, V>
            implements TaggedMultiMap<K, V, T> {

      private Map<Entry<K, V>, T> tags = new HashMap<Entry<K, V>, T>();

      @Override
      public T getTag(K key, V val) {
            return getTag(new Entry<K, V>(key, val));
      }

      @Override
      public T getTag(TaggedMultiMap.Entry<K, V> entry) {
            return tags.get(entry);
      }
```

```
	@Override
	public Set<T> getTags(K key) {
		Set<V> values = get(key);
		Set<T> result = new TreeSet<T>();
		for (V v : values) {
			result.add(getTag(key, v));
		}
		return result;
	}

	@Override
	public void put(K key, V value, T tag) {
		super.put(key, value);
		tags.put(new Entry<K, V>(key, value), tag);
	}

	public static class Entry<K, V> implements TaggedMultiMap.Entry<K, V> {

		K key;
		V val;

		public Entry(K key, V val) {
			this.key = key;
			this.val = val;
		}

		@Override
		public K getKey() {
			return key;
		}

		@Override
		public V getValue() {
			return val;
		}

		@Override
		public int hashCode() {
			final int prime = 31;
			int result = 1;
			result = prime * result + ((key == null) ? 0 : key.hashCode());
			result = prime * result + ((val == null) ? 0 : val.hashCode());
			return result;
		}

		@Override
		public boolean equals(Object obj) {
			if (this == obj)
				return true;
			if (obj == null)
				return false;
			if (getClass() != obj.getClass())
				return false;
			Entry other = (Entry) obj;
			if (key == null) {
				if (other.key != null)
					return false;
			} else if (!key.equals(other.key))
				return false;
			if (val == null) {
				if (other.val != null)
					return false;
			} else if (!val.equals(other.val))
				return false;
			return true;
		}
	}
}
```

ג. בחברת מכשירי הטלפון הסלולרי Mokia, מתכננים את ספר הטלפונים של מכשירי הדור הבא. בגרסה הראשונה של המכשיר ניתן יהיה להזין אנשי קשר ולהם כמה מספרי טלפון. בעל/ת המכשיר הנייד י/תוכל לציין עבור כל מספר טלפון בעת הזנתו האם הוא טלפון בבית, בעבודה, נייד או אחר. אם מספר טלפון הוזן ללא ציון סוגו, ניתן אחר כך להוסיף לו סוג ע"י הזנתו מחדש תוך ציון סוגו.

להלן המנשק Phonebook המתאר את הפונקציונליות לעיל:

```
import java.util.Set;

public interface Phonebook {

	Set<String> getContactPhoneNubers(String name);
	String getContactSpecificPhoneNuber(String name, NumberType type);

	void insert(String name, String number);
	void insert(String name, String number, NumberType type);

	public enum NumberType {
		Home, Work, Mobile, Other;
	}
}
```

ממשו מחלקה המממשת את המנשק Phonebook. אם ברצונכם להוסיף הנחות נוספות על השימוש במחלקה ציינו אותן כהערות מעל מימושי השרותים בתחביר פורמאלי ככל הניתן.

```
public class SimplePhonebook implements Phonebook {

	private TaggedMultiMap<String, String, NumberType> db =
				new SimpleTaggedMultiMap<String, String, NumberType>();

	public Set<String> getContactPhoneNubers(String name){
		return db.get(name);
	}

	public String getContactSpecificPhoneNuber(String name, NumberType type){
		Set<String> numbers = getContactPhoneNubers(name);
		for (String number : numbers) {
			if (db.getTag(name, number) == type)
				return number;
		}
		return null;
	}

	public void insert(String name, String number){
		db.put(name, number);
	}

	public void insert(String name, String number, NumberType type){
		db.put(name, number, type);
	}
}
```

שאלה 3 (20 נקודות)

א. נתונה המחלקה TwoDPoint הבאה:

```
1. public class TwoDPoint {
2.     int x,y;

3.     public TwoDPoint(int x, int y) {
4.         this.x = x;
5.         this.y = y;
6.     }
7. }
```

אומה הסתומה, שמצאה את המחלקה באינטרנט החליטה להשתעשע איתה:

```
1. TwoDPoint p1 = new TwoDPoint(1,2);
2. TwoDPoint p2 = new TwoDPoint(1,2);
3. System.out.println(p1.equals(p2));
```

מה רבה הייתה הפתעתה כאשר גילתה שהפלט אינו כפי שציפתה, אלא: **false**

האם אתם, ילדים, תוכלו לעזור לאומה להבין את פשר הדבר? אילו שינויים יש לעשות בקוד כדי שיודפס true? ממשו בקוד את כל השינויים הדרושים:

יש להוסיף למחלקה TwoDPoint את מימוש השרותים equals ו- hashCode (לא חובה לממש את hashCode):

```
@Override
public int hashCode() {
	final int prime = 31;
	int result = 1;
	result = prime * result + x;
	result = prime * result + y;
	return result;
}

@Override
public boolean equals(Object obj) {
	if (this == obj)
		return true;
	if (obj == null)
		return false;
	if (getClass() != obj.getClass())
		return false;
	TwoDPoint other = (TwoDPoint) obj;
	if (x != other.x)
		return false;
	if (y != other.y)
		return false;
	return true;
}
```

ב. ביל הדביל מצא את המחלקה הבאה באינטרנט:

```
1. public class ThreeDPoint extends TwoDPoint {
2.     int z;

3.     public ThreeDPoint(int x, int y, int z) {
4.         super(x, y);
5.         this.z = z;
6.     }
7. }
```

והחליט להשתעשע איתה קצת. אומה, שבינתיים הספיקה לתקן את TwoDPoint, נתנה לו את המימוש שלה (שגורם לקוד בסעיף א' להדפיס true) והם שניהם הריצו את הקוד הבא:

```
1. TwoDPoint p1 = new ThreeDPoint(1,2,3);
2. TwoDPoint p2 = new ThreeDPoint(1,2,3);
3. TwoDPoint p3 = new ThreeDPoint(1,2,4);

4. Set<TwoDPoint> set = new HashSet<TwoDPoint>();
5. set.add(p1);
6. System.out.println(set.contains(p1));
7. System.out.println(set.contains(p2));
8. System.out.println(set.contains(p3));
```

מה רבה הייתה הפתעתם כאשר גילו שהפלט אינו כפי שציפו...
מה מדפיסה התוכנית? הסבירו את הסיבה לכל אחת מההדפסות:

התשובה תלויה בתשובתכם לסעיף הקודם:

- עבור המימוש המופיע בסעיף א' תדפיס התוכנית:

  true
  true
  true

  הכנסת האברים ל Set ושליפתם מתבססות על השרותים equals ו- hashCode ממחלקת הבסיס אשר מתייחסים רק לשני האברים הראשונים.

- תלמידים אשר מימשו בסעיף א' רק את השרות equals (ולא את hashCode) יקבלו את ההדפסה:

  true
  false
  false

  הכנסת האברים ל Set ושליפתם מתבססות על השרות equals של מחלקת הבסיס ומימוש hashCode של המחלקה Object, אשר מקצה לעצמים hashCode ללא קשר לתוכנם. במקרה כזה גם שני עצמים זהים בתוכנם עלולים לא להיות באותו תא ב- Set ולכן השרות contains לא יגיע להשוות בינהם.

הערה: תשובות "נכונות" ללא הסבר לא קיבלו ניקוד מלא!

ג. אילו שינויים יש לעשות בקוד כדי שפלט התוכנית יהיה:

true
true
false

יש לממש את השרותים equals ו- hashCode של המחלקה ThreeDPoint

שאלה 4 (25 נקודות)

א. בנבכי קוד המקור של הרשת החברתית Fakebook מצאה המפתחת אלכסנדרה את השרות הבא:

```
private static void postCommentsToFriendsWall(Entry entry, Contact [] friends){

        for(int i = 0; i < MAX_NUMBER_OF_FRIENDS; i++){
                friends[i].postCommentToWall(entry.getComments());
        }
}
```

MAX_NUMBER_OF_FRIENDS הוא משתנה ציבורי (public) שהוגדר קודם לכן.

למרות שהשרות מתקפל ורץ עבור קלטים רבים, אלכסנדרה מבינה מקריאת הקוד שכותב השרות מימש אותו עבור הקשר מסוים ותחת הנחות מסוימות, והיא מחליטה לנסח אותן בעזרת חוזה.

עיזרו לאלכסנדרה לנסח את החוזה של השרות, השתמשו בתחביר פורמאלי חוקי במידת האפשר:

```
@pre entry != null
@pre friends != null
@pre friends.length > MAX_NUMBER_OF_FRIENDS
@pre foreach friend in friends: friend != null
```

חוזים שגויים היו:

MAX_NUMBER_OF_FRIENDS > 0
MAX_NUMBER_OF_FRIENDS > friends.length
entry.getComments() != null

ב. מארק, מפתח ותיק בחברה, ממליץ לאלכסנדרה לשכתב את הקוד כך שלא יהיו לשרות תנאי קדם כלל (precondition: true) אולם יהיו לו תנאי צד. מארק ממליץ לעשות שימוש בחריגים שאינם נבדקים (unchecked exceptions) שימסכו מקרי קצה (או חריגים אחרים) אשר עשויים להפתיע את לקוחות הפונקציה וייתנו חיווי טוב יותר על אופי התקלות שקרו. עיזרו לאלכסנדרה לשכתב את מימוש הפונקציה כפי שהמליץ לה מארק. במקרה הצורך, ניתן להגדיר מבני עזר חדשים לצורך המימוש:

```
private static void postCommentToFriendsWall (Entry entry, Contact [] friends){

  if(entry == null)
        throw new IllegalArgumentException("entry could not be null");

  if(friends == null)
        throw new IllegalArgumentException("friends could not be null");

  if(friends.length < MAX_NUMBER_OF_FRIENDS)
        throw new IllegalArgumentException("too few friends");

  for(int i = 0; i < MAX_NUMBER_OF_FRIENDS; i++){
    if(friends[i] == null)
      throw new IllegalArgumentException("friend[" + i + "] could not be null");
    else
      friends[i].postCommentToWall(entry.getComments());
  }
}
```

פתרונות נוספים שהתקבלו היו לעטוף את המימוש המקורי בבלוק try-catch, לתפוס חריגים אם נזרקו, ואז לזרוק חריגים משמעותיים ללקוח (ע"י הגדרת חריגים חדשים או הוספת הודעות שגיאה לשדה ה message של חריגים קיימים).

שגיאות נפוצות:

- זריקת חריג נבדק
- זריקת חריג כללי מדי ללא הודעה מתאימה:
  - לא טוב: throw new ArrayIndexOutOfBoundException
  - כן טוב: throw new ArrayIndexOutOfBoundException("friends array too short")
- הדפסה למסך
- אותו חריג נזרק בכמה מקרי קצה
- שימוש ב instanceof במקום שימוש בריבוי בלוקי catch

ג. אלכסנדרה לא מרוצה מהמימוש הקודם ומחליטה לשכתב את מימוש השרות כך שלא יהיו לו תנאי קדם או תנאי צד כלל (אינה זורקת חריגים, precondition: true) – ובמקום זאת יהיה לו **ערך מוחזר משמעותי**. עיזרו לאלכסנדרה לשכתב את הפונקציה והשלימו את החוזה שלה (יש יותר מפתרון אחד אפשרי). השתמשו בתחביר פורמאלי חוקי במידת האפשר. במקרה הצורך, ניתן להגדיר מבני עזר חדשים לצורך המימוש:

```
/**
 * @pre true
 *
 * @post $ret == number of friends that were actually posted
 * @post (entry == null || friends == null) $implies $ret == 0
 */
private static int postCommentToFriendsWall (Entry entry, Contact [] friends){
  int numOfFriendsPostedTo = 0;
    try {
      for(int i = 0; i < MAX_NUMBER_OF_FRIENDS && i < friends.length; i++){
        if(friends[i] != null){
          friends[i].postCommentToWall(entry.getComments());
          numOfFriendsPostedTo++;
        }
      }
    } catch (Exception e) {   }
    return numOfFriendsPostedTo;
}
```

פתרונות נוספים שהתקבלו:

- החזרת enum עם קודי שגיאה מתאימים

שגיאות נפוצות:

- החזרת ערך שגיאה כללי מדי (שלא מאפשר הפרדה בין מקרים שונים)
- החזרת ערך שגיאה מטיפוס String

ד. שכתוב חתימה של שרות הוא צעד נדיר עבור קוד הנמצא כבר בשימוש. מדוע במקרה זה מותר היה לאלכסנדרה לשכתב גם את חתימת השרות? ציינו 2 סיבות:

1. ניתן להשתמש בשרות פרטי (private) רק מתוך המחלקה שבה הוא מוגדר, ולכן שינוי של חתימת השרות יכול להשפיע רק על קוד מתוך המחלקה, ולא יפגע בלקוחות של המחלקה.
2. גם אם השרות לא היה פרטי, כאשר שרות עם ערך מוחזר void משוכתב להחזיר ערך משמעותי, אין חשש להפתיע לקוחות, מכיוון שהם לא קלטו ערך מוחזר מהגרסה הקודמת.

נקודה נוספת נכונה (אבל לא קשורה לשאלה הזו), שצוינה על ידי כמה תלמידים היתה שהוספת ערך מוחזר לשרות void עומדת בכללי עיצוב ע"פ חוזה (design by contract).

ושוב, בהצלחה!