

בחינה בתוכנה 1

סמסטר ב תשפ"ב, מועד א', 14 ביולי 2022
לנה דנקין, אמיר הרץ, אלה גולדשמידט

משך הבחינה שלוש שעות.

סך הניקוד על השאלות בבחינה הוא 105, אך הציון המקסימלי אותו ניתן לקבל הוא 100.

יש להניח, אלא אם צויין אחרת, כי:

- הקוד שמופיע במבחן מתאים לגירסא Java 8.
- כל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import בגוף הקוד.
- כל מחלקה שהיא public מופיעה בקובץ Java משלה.
- בכל שאלה, כל המחלקות מופיעות באותה חבילה (package).
- בזמן הבחינה, אתם נדרשים לזהות שגיאות קומפילציה שנוצרות כתוצאה מהפרת עקרונות Java-יים ושימוש לא נכון במחלקות/פונקציות. במידה וישנה טעות הקלדה (סוגר חסר, שימוש באות גדולה שלא לצורך וכו') אין לראות בסיבות אלה גורמים לשגיאות קומפילציה.
- בסוף הבחינה מופיע נספח עם תיעוד של מחלקות שאתם עשויים לעשות בהן שימוש בחלק הפתוח של הבחינה.
- הקוד שאתם נדרשים לספק צריך להיות יעיל ולהימנע ממחזור קוד. חלק מהציון ניתן גם היבטים אלה, ולא רק על נכונות הפתרון.

בבחינה זו מופיע קוד שבחלקו אינו מתקמפל, אינו רץ או שנוגד את הסטנדרטים של Java כפי שנלמדו בקורס, וזאת מתוך מטרה לבחון ידע והבנה של נושאים מסוימים. אין לראות בקטעי קוד אלה דוגמא לכתובה נכונה ב Java.

מבנה הבחינה:

- הבחינה מורכבת משני חלקים: חלק פתוח (שתי שאלות על סך 55 נקודות) ושאלות אמריקאיות (10 שאלות, כל אחת שווה 5 נק'). עליכם לענות על הבחינה באופן הבא:
- בשאלות הפתוחות להשלים את הקוד החסר במקומות המסומנים ע"י מסגרת. שימו לב שלא חייבים למלא את כל המסגרות.
 - בשאלות האמריקאיות:
 - לסמן את התשובות הנכונות על גבי טופס סימון התשובות שתקבלו בנפרד.
 - לנמק את תשובתכם על גבי טופס הבחינה. הנימוק הוא לא חובה, אך יכול לעזור לכם במקרים של ערעורים או קבלת יותר מתשובה אחת נכונה.

בסוף שאלה 2 ניתן למצוא מסגרת חירום לשימוש במקרה שהמסגרות שמופיעות בגוף השאלות הפתוחות לא מספיקות לכם.

© כל הזכויות שמורות למחברים. 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. בהצלחה!

שאלה 1 (37 נק'):

שאלה זו עוסקת בהצפנות (ciphers) של טקסטים, כאשר בכל פעולה נצפין או נפענח הודעה אחת שכוללת משפט אחד. עליכם לממש שלושה צפנים בסיסיים ומוכרים (לפחות בחוגים מסויימים).

צופן ספר – בצופן זה נעזרים בספר, וכל מילה מוחלפת במיקום של מופע כלשהו שלה בספר (מבחינתנו, המיקום בספר מיוצג באמצעות מחרוזת).

צופן החלפה – בצופן זה כל אות באלפבית מוחלפת באות אחרת, כאשר המיפוי הוא חח"ע ועל במילים אחרות, הצופן הוא פרמוטציה כלשהי על האלפבית.

צופן ויז'נר – צופן שבו כל אות מוחלפת באות אחרת, אך להבדיל מצופן ההחלפה, אותה האות יכולה להיות מוחלפת באותיות שונות, כתלות במיקום שלה במשפט ובמפתח ההצפנה (פרטים בהמשך).

עצבו את המחלקות על פי הדרישות ע"י שמירה על עקרונות של עיצוב נכון ושימוש חוזר בקוד. תחילה, נגדיר את הממשק ICipher אשר מייצג צופן.

```

/* for each text, decipher(encipher(text)) == text */
public interface ICipher {

    public String encipher(String text);

    public String decipher(String text);

    public static void printLog(String from, String to,
                                int cnt, int total) {
        System.out.println(
            from + " -> " + to + " [" + cnt + "/" + total + "]);
    }
}

```

הפונקציה encipher מקבלת משפט ומחזירה משפט מוצפן שבו כל 2 מילים מופרדות ע"י רווח בודד. ההצפנה נעשית מילה מילה ולאחר הצפנת כל מילה תודפס הודעת סטטוס על התקדמות ההצפנה.

הפונקציה decipher מקבלת משפט מוצפן ומחזירה משפט מופענח שבו כל 2 מילים מופרדות ע"י רווח בודד. הפענוח נעשה מילה מילה ועבור כל מילה שפוענחה תודפס הודעת סטטוס בדומה להצפנה.

הפונקציה printLog היא פונקציית עזר להדפסת סטטוס ההתקדמות של ההצפנה. לאחר הצפנה/פענוח כל מילה, יודפסו המילה המקורית והמפוענחת/מוצפנת, וכן מספר המילים שכבר הוצפנו/פוענחו מתוך סך המילים בהודעה (ראו פלט לדוגמא בעמוד הבא).

ניתן להניח שכל המשפטים מכילים רק מילים המורכבות מ 26 האותיות a-z (ב lowercase). בין כל 2 מילים יפריד לפחות רווח אחד. בנוסף, הניחו כי הקלטים חוקיים, כל מחרוזת מכילה לפחות מילה אחת וכי כל הודעה ניתנת לפענוח/הצפנה, כך שאין צורך לטפל בשגיאות הנובעות מקלטים לא חוקיים.

הנחיה כללית: עליכם לשבור כל מחרוזת למילים ובמידת הצורך לאותיות, ואז לבצע את ההצפנות/פענוחים. את ההדפסות יש לבצע תוך כדי תהליך הפענוח/הצפנה, לאחר השלמת פענוח/הצפנה של כל מילה. הקפידו על יעילות הקוד ועל שימוש חוזר בקוד.

בסעיפים א+ב תממשו חלקית את צופן הספר (בנאי והצפנה). בסעיפים ג+ד תממשו חלקית (בנאי והצפנה) את צופן ההחלפה. בסעיף ה' תממשו חלקים מהקוד של צופן ויז'נר (חלק מהקוד נתון לכם). בסעיף ו' תסבירו במילים כיצד יש להרחיב את הקוד על מנת למתוך גם בפענוח עבור צופן החלפה.

סעיפים א + ג הם סעיפים עצמאיים שפתרונם לא תלוי במימוש של הסעיפים האחרים. שאר הסעיפים תלויים האחד בשני ולכן עליכם לקרוא את כל השאלה על מנת לבחור את עיצוב המחלקות הנכון. החלוקה לסעיפים נועדה לנתק תלויות ולאפשר לכם לקבל ניקוד מקסימלי על ידע חלקי.

סעיף א' (5 נק')

במחלקה זו תממשו את בנאי המחלקה BookCipher המממשת הצפנת ספר ברמת מילה. בהנתן ספר מסויים, כל מילה w בהודעה המקורית מוחלפת במיקום שלה בספר (מיקום מיוצג ע"י מספר עמוד, מספר שורה, מספר מילה בשורה). בשאלה אנו מניחים שכל מילה בהודעה המקורית מופיעה בספר שנבחר. אם מילה w מופיעה יותר מפעם אחת בספר, נבחר בכל פעם מופע אקראי של המילה. דוגמא להודעה מוצפנת: "moomins are great" ⇒ "<6,13,4> <19,1,3> <3,12,1>"

המילה moomins מופיעה בעמוד 6, שורה 13, מילה 4, ולכן תוצפן ע"י המחרוזת <6,13,4>. את תוכן הספר נייצג ע"י מילון אשר מכיל את המיפוי של כל מילה בספר לרשימה של מחרוזות המייצגות את מיקומי המופעים השונים של המילה בספר. בדוגמא המצורפת, bookCipherMap מייצג את תוכן הספר. כל ההדפסות שהתוכנית מבצעת מופיעות בהערה.

```
Map<String, List<String>> bookCipherMap = new HashMap<>();
bookCipherMap.put("moomins", Arrays.asList("6,13,4"));
bookCipherMap.put("are", Arrays.asList("1,3,4", "19,1,3"));
bookCipherMap.put("great", Arrays.asList("3,12,1", "12,12,12"));
String text = "moomins are great";
ICipher bCipher = new BookCipher(bookCipherMap);
String enc = bCipher.encrypt(text);           //moomins -> <6,13,4> [1/3]
                                           //are -> <19,1,3> [2/3]
                                           //great -> <3,12,1> [3/3]
String dec = bCipher.decrypt(enc);           //<6,13,4> -> moomins [1/3]
                                           //<19,1,3> -> are [2/3]
                                           //<3,12,1> -> great [3/3]
System.out.println(enc); //<6,13,4> <19,1,3> <3,12,1>
System.out.println(dec); //moomins are great
```

עבור המילים are ו great קיימות 2 הצפנות אפשריות (מופיעות ב 2 מקומות שונים בספר), כך שיתכנו הצפנות שונות עבור משפט זה.

עליכם לשמור שני מיפויים: מיפוי עבור ההצפנה (encryptionMap – כבר מאותחל) ומיפוי עבור הפענוח (decryptionMap). עליכם לאתחל את decryptionMap ולהשלים את ההצהרה על השדה.

הנחיה: התוים ">" ו "<" מתווספים ומוסרים רק בעת הצפנה/פענוח, ואינם נשמרים במבני הנתונים של המחלקה.

דוגמא: עבור הצופן שמופיע בדוגמא נקבל: decryptionMap.get("<19,1,3>") = "are"

```
public class BookCipher /** Fill in next section */ {

    private Map<String, List<String>> encryptionMap;
    private Map<String, String> decryptionMap;

    public BookCipher(Map<String, List<String>> bookContent) {
        this.encryptionMap = bookContent;

        this.decryptionMap = new HashMap<>();
        for (String word : encryptionMap.keySet()) {
            for (String dec : encryptionMap.get(word)) {
                decryptionMap.put(dec, word);
            }
        }
    }
}
```

סעיף ב' (8 נק')

בסעיף זה תמשיכו את מימוש המחלקה BookCipher המממשת את המנשק ICipher. עליכם להשלים את ההצהרה על המחלקה ובנוסף יש לממש את השירות encipher. אין צורך לממש את הפענוח (decipher). להזכירכם, הפונקציה encipher מחזירה את המחרוזת המוצפנת וכן מדפיסה הודעות סטטוס תוך כדי תהליך ההצפנה.

במידת הצורך, הגדירו מחלקות אבסטרקטיות על מנת לשתף קוד עם המחלקות שימושו בסעיפים הבאים. תזכורת – בכל הצפנה יש לפצל את המשפט למילים ולאחר הצפנה של כל מילה יש לבצע הדפסת סטטוס. אין להוסיף שדות או בנאים מעבר למה שהוגדר עבורכם בסעיפים הקודמים.

```
public class BookCipher extends AbstractWordLevelCipher {
```

```

@Override
    public String encipherSingleWord(String word) {
        //rand should have been declared as a member.
        Random rand = new Random();
        List<String> cipherOptions = encryptionMap.get(word);
        return "<" +
            cipherOptions.get(rand.nextInt(cipherOptions.size()))
                + ">";
    }

/* This solution contains also the decription part, which was
unnecessary in this question. It's only relevant to part 6 */
public abstract class AbstractWordLevelCipher implements ICipher{

    @Override
    public String encipher(String text) {
        return encryptDecrypt(text, true);
    }

    @Override
    public String decipher(String text) {
        return encryptDecrypt(text, false);
    }

    public String encryptDecrypt(String text, boolean encrypt) {
        String[] words = text.split(" ");
        List<String> resSentWords = new ArrayList<>();
        for(String word: text.split(" ")) {
            if (encrypt) {
                resSentWords.add(encipherSingleWord(word));
            }
            else {
                resSentWords.add(decipherSingleWord(word));
            }
        }
        ICipher.printWordsNum(words.length, encrypt);
        return String.join(" ", resSentWords);
    }
}

```

```
protected abstract String encipherSingleWord(String word);
protected abstract String decipherSingleWord(String word);
}
```

סעיף ג' (5 נק'):

בסעיף זה תממשו את בני המחלקה SubstitutionCipher הממשת צופן החלפה. צופן ההחלפה הוא צופן המחליף כל אות באלפבית באות אחרת על פי פרמוטציה כלשהי. לדוגמא, נגדיר את הפרמוטציה הבאה: כל אות מוחלפת באות שנמצאת 2 מקומות אחריה באופן מעגלי (כשמגיעים לסוף האלפבית חוזרים להתחלה, כך שהאות y תוחלף ב a והאות z תוחלף ב b).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	אינדקס
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	אות מקורית
c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	אות מוצפנת

להלן דוגמת קוד אשר עושה שימוש בצופן זה (בדוגמא מופיעה רק הצפנה על מנת לקצר את הדוגמא. פענוח עובד בצורה דומה). ליד כל שורה שבה מתבצעות הדפסות מופיע הפלט.

```
char[] permutation = /* initialization here */
ICipher cipher = new SubstitutionCipher(permutation);
String text = "i love moomins";
String enciphered = cipher.encipher(text); //i -> k [1/3]
                                           //love -> nqyg [2/3]
                                           //moomins -> oqqokpu [3/3]
System.out.println(enciphered); //k nqyg oqqokpu
```

בדוגמא הזו, הניחו כי המערך permutation מאוחל עם הפרמוטציה שמתוארת בטבלה. באינדקס 0 המייצג את האות a מופיעה האות c, באינדקס 1 המייצג את b מופיעה האות d וכו'.

בנאי המחלקה מקבל מערך המייצג את הפרמוטציה ומאתחל שני שדות: מערך עבור הצפנה (encipherArr - מאוחל עבורכם), ומערך עבור פענוח (decipherArr). בשני המערכים, התא הראשון מייצג את האות a, התא השני את האות b וכן הלאה, בדומה למה שהוסבר קודם. מערך ההצפנה הוא למעשה הפרמוטציה שמקבלים כקלט (אותחל עבורכם), ואת מערך הפענוח עליכם לבנות בעצמכם.

עבור צופן החלפה עם הפרמוטציה שמופיעה בדוגמא:

```
decipherArr[25] = 'x', decipherArr[24] = 'w'
```

באינדקס 25 מופיע הפענוח של z. אנחנו יודעים ש x מוצפן ל z, ולכן הפענוח של z הוא x.

תזכורת לעבודה עם תוים (char-ים): `(char)('a'+1) = 'b'`

```

public class SubstitutionCipher /** fill in next section */ {
    private char[] encipherArr;
    private char[] decipherArr;

    public SubstitutionCipher(char[] permutation) {
        this.encipherArr = permutation;

        this.decipherArr = new char[26];
        for (int i = 0; i < 26; i++){
            decipherArr[permutation[i] - 'a'] = (char)('a' + i);
        }
    }
}

```

סעיף ד' (8 נק')

השלימו את מימוש המחלקה SubstitutionCipher הממשת את ICipher. בדומה לסעיף הקודם, עליכם להשלים את ההצהרה על המחלקה ואת מימוש השירות encipher.

במידת הצורך, הגדירו מחלקות אבסטרקטיות על מנת לשתף קוד עם המחלקות שימומשו בסעיפים הבאים. תזכורת – בכל הצפנה יש לפצל את המשפט למילים ולאחר הצפנה של כל מילה יש לבצע הדפסת סטטוס. אין להוסיף שדות או בנאים מעבר למה שהוגדר עבורכם בסעיפים הקודמים.

```

public class SubstitutionCipher extends AbstractCharLevelCipher {

```

```

/* This solution also contain decryption implementation, which is unnecessary */

protected char encryptSingleChar(char c) {
    return encipherArr[c-'a'];
}

protected char decryptSingleChar(char c) {
    return decipherArr[c-'a'];
}

public abstract class AbstractCharLevelCipher extends
    AbstractWordLevelCipher {

    protected String encryptDecryptSingleWord(String word,
        boolean encrypt) {
        StringBuffer sb = new StringBuffer();
        for (char c : word.toCharArray()) {
            if (encrypt) {
                sb.append(encryptSingleChar(c));
            }
            else {
                sb.append(decryptSingleChar(c));
            }
        }
        return sb.toString();
    }
}

```

```

@Override
protected String encipherSingleWord(String word) {
    return encryptDecryptSingleWord(word, true);
}

@Override
protected String decipherSingleWord(String word) {
    return encryptDecryptSingleWord(word, false);
}

protected abstract char encryptSingleChar(char c);

protected abstract char decryptSingleChar(char c);
}
    
```

סעיף ה' (6 נק'):

בסעיף זה תממשו (באופן חלקי) צופן שלישי. צופן זה הוא הפשטה של צופן ויז'נר והוא מוגדר באופן הבא: הצופן עושה שימוש במפתח, שהוא בעצם מערך של מספרים שלמים בין 0 ל 25 (כולל הקצוות). המערך יכול להיות בכל גודל.

נגדיר מפתח לדוגמא [1,5,12,0] וננסה להצפין את ההודעה i love the moomins.

i		l	o	v	e		t	h	e		m	o	o	m	i	n	s	הודעה מקורית
1		5	12	0	1		5	12	0		1	5	12	0	1	5	12	מפתח
j		q	a	v	f		y	t	e		n	t	a	m	j	s	e	הצפנה

למעשה, אנחנו פורשים את המפתח לאורך ההודעה המקורית (אם הוא קצר מדי חוזרים עליו מההתחלה), ומתאימים לכל אות בהודעה המקורית מספר. המספר מגדיר את ההיסט (offset) שבו נזיז את האות קדימה בשביל להצפין אותה. האות הראשונה בהודעה (i) תוצפן עם היסט 1, כלומר, ע"י האות j. האות l המופיעה אחריה תוצפן עם היסט 5 (נקבל את האות q), וכן הלאה.

שימו לב: בכל קריאה ל cipher/decipher נתחיל מהאיבר הראשון במפתח (אחרת לא נוכל לפענח שום הודעה מוצפנת).

השלימו מימוש חלקי של VigenereCipher. הבנאי כבר מומש עבורכם ואין להוסיף שדות או בנאים נוספים. עליכם לממש רק את החלק שמטפל בהצפנה (encipher). העזרו בשירות shiftChar אשר עבור תו c והיסט offset יחזיר את התו שנמצא בהיסט של offset מהתו c. למשל, עבור הקלט c='a', offset=3 הפונקציה תחזיר את התו d.

למעשה, הלוגיקה היחידה שעליכם לממש היא הלוגיקה שמתאימה לכל תו את ההיסט הנכון. העזרו בשדה currIndex אשר מציין היכן אתם נמצאים כרגע במפתח, ועדכנו אותו כשצריך.

```

public class VigenereCipher extends AbstractCharLevelCipher {
    private int[] keyArr;
    private int currKeyIndex;

    public VigenereCipher(int[] keyArr) {
        this.keyArr = keyArr;
        this.currKeyIndex = 0;
    }

    /* shiftChar('a', 1) = 'b', shiftChar('d', -2) = 'b' */
    protected char shiftChar(char c, int offset) {
        /* implementation is provided, no need to implement */
    }
}
    
```

```

@Override
public String encryptDecrypt(String text, boolean encrypt) {
    this.currKeyIndex = 0; /* each new sentence starts with
                           currKeyIndex = 0 */
    return super.encryptDecrypt(text, encrypt);
}

@Override
protected char encryptSingleChar(char c) {
    return shiftChar(c, getNextOffset());
}

@Override
protected char decryptSingleChar(char c) {
    return shiftChar(c, -getNextOffset());
}

private int getNextOffset() {
    return keyArr[currKeyIndex++ % keyArr.length];
}
}

```

סעיף ו' (5 נק'):

קעת נרצה לממש את הפענוח (decipher) עבור המחלקה SubstitutionCipher. הסביר אילו שירותים יש לנו או יתווספו ל SubstitutionCipher ולאו למחלקות אחרות (כתלות בעיצוב שלכם). בשאלה זו אינכם נדרשים לכתוב קוד, רק לתאר את השינויים שידרשו ואת הסיבות לשינויים.

הרעיון במימוש של ה decipher הוא שהקוד כמעט זהה לקוד של encipher, למעט הפעולה האטומית של הצפנה ברמת אותמילה. מסיבה זו צריך להיות שיתוף קוד בין 2 הפעולות, למשל, באופן דומה לזה המופיע בפתרון של סעיפים ב,ד,ה (יש גם פתרונות אחרים).

כל פתרון שלא התייחס לשיתוף הקוד הזה, ותאר מסלול כללי (נממש, בדומה ל encipher) קיבל בין 0 ל 1 נק' (כתלות ברמת הפירוט), כיוון שמדובר בתשובה טריוויאלית. תשובות אשר התייחסו לכך שפעולת פענוח היא סוג של פעולת הצפנה, ולכן צריך לשתף קוד, קיבלו את רובלכל הניקוד, גם אם לא פרטו את השינויים שצריך להכניס לקוד שלהם.

התשובה לשאלה זו לא תלויה בנכונות עיצוב המחלקות שהוגדר בסעיפים הקודמים. גם אם אין שיתוף קוד בין BookCipher ו SubstitutionCipher, עדין ניתן לשתף קוד בין פעולה של פענוח לפעולה של הצפנה.

שאלה 2 (18 נק')

סעיף א' (10 נק'):

נגדיר כיווץ רשימות ע"י כיווץ של רצפים של איברים זהים (זהות נקבעת ע"י equals).

```
[ "x", "x", "x", "x", "x", "y", "y", "x" ] ⇒ [ <"x", 5>, <"y", 2>, <"x", "1"> ]
```

כלומר, כל רצף של אותו האיבר מיוצג ע"י זוג של איבר ואורך הרצף.

המחלקה הגנרית CompactList מממשת רשימה מכווצת. להלן דוגמת שימוש במחלקה:

```
CompactList<String> cList = new CompactList<>();
cList.add("x", 5);
cList.add("y", 2);
cList.add("x", 1);
for (String elem : cList) {
    System.out.println(elem);
}
```

הרשימה cList היא רשימה מכווצת (שומרת רצפים של איברים זהים), אך היא מייצגת רשימה אמיתית של 8 איברים ולולאת ה for תחזיר סה"כ 8 איברים.

נגדיר את מחלקת העזר RepeatedElement שתייצג רצף (איבר + מספר חזרות):

```
RepeatedElement<String> e = new RepeatedElement<>("x", 5);
System.out.println(e.getElement()); //x
System.out.println(e.getRepetitionsNum()); //5
```

עבור הרשימה שנוצרה בדוגמת הקוד העלונה נצטרך שלושה מופעים של RepeatedElement: הראשון שיאותחל ע"י ("x", 5), השני יאותחל ע"י ("y", 2) והשלישי יאותחל ע"י ("x", 1).

השלימו את המימוש של המחלקה CompactList כך שהאיטרטור שלה ידפיס את איברי הרשימה המלאה (בדוגמא שמופיעה בשאלה - כל 8 האיברים). האיטרטור ממומש בתוך המחלקה הפנימית CompactIterator.

שימו לב: בשום שלב של הריצה אין לשחזר את הרשימה הלא מכווצת בזכרון. הגדירו שדות עזר ואתחלו אותם בשורת ההגדרה של השדה (הבנאי של המחלקה הוא ריק).

את הקוד יש להשלים בתוך הריבועים הריקים, אך לא חובה למלא את כולם.

```
public class CompactList<T> implements Iterable <T> {

    private List<RepeatedElement<T>> innerList = new ArrayList<>();

    /* @pre repetitions > 0 */
    public void add(T element, int repetitions) {
        this.innerList.add(
            new RepeatedElement<T>(element, repetitions));
    }

    public Iterator<T> iterator() { return new CompactIterator(); }
}
```

```
//this is an inner class inside CompactList<T>
public class CompactIterator  implements Iterator  {
```

```
//add members
private int currIndex = 0;
private int returned = 0;
```

```
public CompactIterator{
```

```
public boolean hasNext() {
```

```
    boolean moreRepeatedElements =
        currIndex < innerList.size() - 1 ;
    boolean moreRepetitions =
        returned < innerList.get(currIndex).getRepetitionsNum();
    return moreRepeatedElements || moreRepetitions;
```

```
}
```

```
public T next() {
```

```
    # more repetition of currElement
    if (returned <
        innerList.get(currIndex).getRepetitionsNum()) {
    }
    # move to next RepeatedElement
    else {
        returned = 0;
        currIndex++;
    }
    returned++;
    return innerList.get(currIndex).getElement();
```

```
}
```

```
}
```

```
}
```

```
}
```

סעיף ב' (8 נק')

קבוצת סידון (sidon set) היא קבוצת מספרים שבה לא קיימים 2 זוגות שונים של מספרים שסכומם זהה. לדוגמא {1,2,8,10} היא קבוצת סידון, ו {1,2,8,9} אינה קבוצת סידון ($2+8 = 1+9$).

ממשו את הפונקציה getMaxKForSidonSet אשר עבור רשימה lst מחזירה את ה k המקסימלי עבורו k האיברים הראשונים ברשימה list מהווים קבוצת סידון. ניתן להניח שאין ב list מספר שחוזר על עצמו פעמיים.

לדוגמא: עבור הרשימה [1,2,8,10,9,13], הפונקציה תחזיר את המספר 4 כיוון ש {1,2,8,10} מהווה קבוצת סידון, אך אם נוסיף את המספר החמישי (9) נקבל 2 זוגות שסכומם זהה ($1+10 = 2+9$). עבור הרשימה [1,2,10,9] הפונקציה תחזיר את המספר 3, שכן {1,2,10} מהווה קבוצת סידון, אך אם נוסיף את המספר הרביעי (9) נקבל 2 זוגות שסכומם זהה.

הנחיה: חשבו סכום של כל זוג מספרים פעם אחת בלבד. ניתן להגדיר מבני נתונים לעזר.

```

/ * @pre: lst.size() >= 1
 * @post: min(3, lst.size()) <= $ret <= lst.size() */
public static int getMaxKForSidonSet(List<Integer> lst) {

```

```

    Set<Integer> sums = new HashSet<>();
    for (int k = 1; k < lst.size(); k++) {
        for (int i = 0; i < k; i++) {
            int newSum = lst.get(i) + lst.get(k);
            if (sums.contains(newSum)){
                return k; /*f element in position k violated
the sidon set property, then we have k valid elements: from 0 to k-1*/
            }
            sums.add(newSum);
        }
    }

    return lst.size(); /* We reach here in case all the list
qualifies as a sidon set, and thus k = lst.size() */
}
}

```

A large empty rectangular box with a thin black border, occupying most of the page. It is intended for a drawing or a written response.

שאלה 3 (5 נק'): _____

```

public class Q3 {
    public static void main(String[] args) {
        List<Integer> intsList = new ArrayList<>();
        List rawList = intsList; /*
        rawList.add("abc"); //#
        Object obj = rawList.get(0); //$
        Integer i = intsList.get(0); //@
    }
}

```

בחר/י בתשובה הטובה ביותר:

- א. יש שגיאת קומפילציה בשורה המסומנת ב *.
- ב. יש שגיאת קומפילציה בשורה המסומנת ב #.
- ג. יש שגיאת קומפילציה בשורה המסומנת ב @.
- ד. התוכנית מתקמפלת. בזמן ריצת התוכנית תיזרק שגיאת זמן ריצה בשורה המסומנת ב *.
- ה. התוכנית מתקמפלת. בזמן ריצת התוכנית תיזרק שגיאת זמן ריצה בשורה המסומנת ב #.
- ו. התוכנית מתקמפלת. בזמן ריצת התוכנית תיזרק שגיאת זמן ריצה בשורה המסומנת ב \$.
- ז. **התוכנית מתקמפלת. בזמן ריצת התוכנית תיזרק שגיאת זמן ריצה בשורה המסומנת ב @.**
- ח. התוכנית מתקמפלת ותרופץ ללא שגיאות.

נימוק:

שאלה 4 (5 נק'): _____

להלן מספר טענות על מנשקים. בחר/י בתשובה הטובה ביותר:

- א. אם שני מנשקים |1| ו |2| מגדירים שירות אבסטרקטי עם אותה החתימה, מחלקה C לא יכולה לממש את שניהם (נקבל שגיאת קומפילציה).
- ב. מנשק יכול לרשת ממחלקה אבסטרקטית, בתנאי שאינה מגדירה שדות ולא מממשת פונקציות.
- ג. ניתן להגדיר שירות בניראות protected במנשק.
- ד. שירות דיפולטי (default) במנשק חייב להידרס ע"י כל מחלקה שמממשת את המנשק.
- ה. **מלבד תשובה זו כל התשובות לא נכונות.**
- ו. מלבד תשובה זו יש יותר מתשובה נכונה אחת.

נימוק:

שאלה 5 (5 נק):

```
public class Base{
    public void foo() throws IOException{ /* implementation here*/ }
}

public class Sub extends Base{
    public void foo() {
        /*super.foo(); */ //#
    }
}

public class Main {
    public static void main(String[] args) {
        Base b = new Sub();
        b.foo();
    }
}
```

בחר/י בתשובה הטובה ביותר:

- א. כל המחלקות מתקמפלות. אם נוציא את שורה # מההערה הקודד ימשיך להתקמפל.
- ב. כל המחלקות מתקמפלות. אם נוציא את שורה # מההערה נקבל שגיאת קומפילציה בשורה זו.
- ג. רק Base ו Sub מתקמפלות. אם נוציא את שורה # מההערה, Sub תמשיך להתקמפל.
- ד. רק Base ו Sub מתקמפלות. אם נוציא את שורה # מההערה, Sub לא תתקמפל.
- ה. רק Base ו Main מתקמפלות. אם נוציא את שורה # מההערה, גם Sub תתקמפל.
- ו. רק Base ו Main מתקמפלות. אם נוציא את שורה # מההערה Sub תמשיך לא להתקמפל.
- ז. רק Base מתקמפלת. אם נוציא את שורה # מההערה, גם Sub תתקמפל.
- ח. רק Base מתקמפלת. אם נוציא את שורה # מההערה, Sub תמשיך לא להתקמפל.

נימוק:

שאלה 6 (5 נק):

בחר/י בתשובה הטובה ביותר:

- א. המהדר (compiler) הופך קבצים עם סיומת class לקבצי הרצה.
- ב. המפרש (interpreter) קורא קבצים עם סיומת java ומתרגם אותם לקבצים עם סיומת class.
- ג. ה Garbage collector יכול למחוק אובייקט מהזכרון אם אין אליו שום מצביע פעיל (כלומר, משתנה או שדה שעדיין קיים בתוכנית), גם ללא ביצוע מחיקה יזומה בקוד.
- ד. מלבד תשובה זו כל התשובות לא נכונות.
- ה. מלבד תשובה זו יש יותר מתשובה נכונה אחת.

נימוק:

שאלה 7 (5 נק'):

```
public class Q7 {
    private static int cnt;
    public int[] arr;

    public void func() {
        String s = "abc";
    }
}
```

לפניכם 3 טענות על הקוד המצורף:

- טענה 1: השדה cnt נשמר על ה stack ומקבל אתחול דיפולטי של 0.
- טענה 2: השדה arr נשמר על ה heap ומאוחלל למערך ריק (`arr.length() == 0`).
- טענה 3: המשתנה s נשמר על ה stack ומצביע לאובייקט שנמצא על ה heap.

בחר/י בתשובה הטובה ביותר:

- 1. רק כל הטענות לא נכונות.
- 2. רק טענה 1 נכונה.
- 3. רק טענה 2 נכונה.
- 4. רק טענה 3 נכונה.
- 5. רק טענות 1+2 נכונות.
- 6. רק טענות 1+3 נכונות.
- 7. רק טענות 2+3 נכונות.
- 8. כל הטענות נכונות.

נימוק:

שאלה 8:

בחר/י בתשובה הטובה ביותר:

- א. על מנת למיין רשימה של אובייקטים מטיפוס X, המחלקה X חייבת לממש את הממשק Comparable.
- ב. ניתן לממש Comparator רק עבור מחלקה שאינה Comparable.
- ג. הממשק Comparator עושה שימוש בממשק Comparable.
- ד. מימוש נכון של `compareTo` של הממשק Comparable מקיים `x.compareTo(y) == 0` אם `x.equals(y) == true`.
- ה. מלבד תשובה זו יש יותר מתשובה נכונה אחת.
- ו. מלבד תשובה כל התשובות לא נכונות.

נימוק:

שאלה 9 (5 נק'):

לפניכם 3 טענות על Gui:

טענה 1: אם בתוכנית Gui כלשהי הוספנו כפתור שאין לו שום מאזין (Listener), התוכנית תרוץ והכפתור יוצג, אך לא יגיב ללחיצות.

טענה 2: הלולאה הבאה מופיעה במימוש של כל מאזין (Listener), אם גם לא כתבנו אותה מפורשות.

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

טענה 3: האובייקט המאותחל ע"י השורה הבאה מקשר בין ה SWT לבין מערכת ההפעלה.

```
Display display = Display.getDefault();
```

בחר/י בתשובה הטובה ביותר:

- א. כל הטענות לא נכונות.
- ב. רק טענה 1 נכונה.
- ג. רק טענה 2 נכונה.
- ד. רק טענה 3 נכונה.
- ה. רק טענה 1+2 נכונה.
- ו. רק טענה 1+3 נכונה.
- ז. רק טענה 2+3 נכונה.
- ח. כל הטענות נכונות.

נימוק:

שאלה 10 (5 נק'):

```
public class Q10<S> {
    public <T> void f1(Collection<S> c, List<T> l ) {
        c = l;
    }
    public void f2(List<? extends Number> l1, List<?> l2) {
        l1 = l2;
    }
    public void f3(List<? super Number> l1, List<? super Number> l2) {
        l2.add(l1.get(0));
    }
}
```

אילו מבין הפונקציות f המופיעות במחלקה A מתקמפלות? בחר/י את התשובה הטובה ביותר .

- א. רק f1 מתקמפלת.
- ב. רק f2 מתקמפלת.
- ג. רק f3 מתקמפלת.
- ד. רק f1+f2 מתקמפלות.
- ה. רק f1+f3 מתקמפלות.
- ו. רק f2+f3 מתקמפלות.
- ז. כל הפונקציות מתקמפלות.
- ח. כל הפונקציות לא מתקמפלות.

נימוק:

שאלה 11 (5 נק')

```
public class Base{
    public int i = 3;
    public int func(Object s) {return i; }

    public static int foo(int x) {return x*3; }
}

public class Sub extends Base{
    public int i = 1;
    public int func(String s) {
        return 3*i+ func((Object)s) + foo(1);
    }

    public int func(Object o) {
        return 2*i + super.func(o) + foo(1);
    }

    public static int foo(int x) { return 2*x; }

    public static void main(String[] args) {
        Base b = new Sub();
        System.out.print(b.func("a"));
    }
}
```

מה יקרה בהרצת התוכנית הבאה? בחר/י בתשובה הטובה ביותר:

- א. התוכנית תיכנס לרקורסיה אינסופית.
- ב. ריצת התוכנית תסתיים בהצלחה ויודפס 6.
- ג. ריצת התוכנית תסתיים בהצלחה ויודפס 7.
- ד. ריצת התוכנית תסתיים בהצלחה ויודפס 8.
- ה. ריצת התוכנית תסתיים בהצלחה ויודפס 9.
- ו. ריצת התוכנית תסתיים בהצלחה ויודפס 10.
- ז. ריצת התוכנית תסתיים בהצלחה ויודפס 11.
- ח. ריצת התוכנית תסתיים בהצלחה ויודפס 12.

נימוק:

שאלה 12 (5 נק')

נתון מימוש המחלקה Sub.

```
public class Sub extends Base{
    public String func(String str) {
        return (String)super.func(str);
    }
}
```

לפניכם שלושה מימושים אפשריים למחלקה base.

```
//v1
public class Base{
    public Object func(String str) { return null;}
}

//v2
public class Base{
    private List<String> func(String str) { return null;}
}

//v3
public class Base{
    public final String func(String str){ return null;}
}
```

אילו מבין המימושים מתאים כך שגם Base וגם Sub יתקמפלו? בחר/י בתשובה הטובה ביותר:

- א. כל הגירסאות של Base לא מתאימות.
- ב. רק גירסא v1 מתאימה.
- ג. רק גירסא v2 מתאימה.
- ד. רק גירסא v3 מתאימה.
- ה. רק גירסאות v1+v2 מתאימות.
- ו. רק גירסאות v1+v3 מתאימות.
- ז. רק גירסאות v2+v3 מתאימות.
- ח. כל הגירסאות של Base מתאימות.

נימוק:

public interface Map<K,V>

Modifier and Type	Method and Description
boolean	<code>containsKey(Object key)</code> Returns true if this map contains a mapping for the specified key.
<code>Set<Map.Entry<K, V>></code>	<code>entrySet()</code> Returns a <u>Set</u> view of the mappings contained in this map.
V	<code>get(Object key)</code> Returns the value to which the specified key is mapped, or <u>null</u> if this map contains no mapping for the key.
V	<code>getOrDefault(Object key, V defaultValue)</code> Returns the value to which the specified key is mapped, or <u>defaultValue</u> if this map contains no mapping for the key.
boolean	<code>isEmpty()</code> Returns true if this map contains no key-value mappings.
<code>Set<K></code>	<code>keySet()</code> Returns a <u>Set</u> view of the keys contained in this map.
V	<code>put(K key, V value)</code> Associates the specified value with the specified key in this map. Returns the previous value associated with key, or null if there was no mapping for key.
V	<code>remove(Object key)</code> Removes the mapping for a key from this map if it is present (optional operation). Returns the previous value associated with key, or null if there was no mapping for key.
int	<code>size()</code> Returns the number of key-value mappings in this map.
<code>Collection<V></code>	<code>values()</code> Returns a <u>Collection</u> view of the values contained in this map.

public interface Set<E> extends Collection<E>

boolean	<code>add(E e)</code> Adds the specified element to this set if it is not already present.
void	<code>clear()</code> Removes all of the elements from this set (optional operation).
boolean	<code>contains(Object o)</code> Returns true if this set contains the specified element.
boolean	<code>isEmpty()</code> Returns true if this set contains no elements.
<code>Iterator<E></code>	<code>iterator()</code> Returns an iterator over the elements in this set.
boolean	<code>remove(Object o)</code> Removes the specified element from this set if it is present.
int	<code>size()</code> Returns the number of elements in this set (its cardinality).

public class Random

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

public interface List<E> extends Collection<E>

boolean	add(E e) Appends the specified element to the end of this list. Always returns true.
boolean	contains(Object o) Returns true if this list contains the specified element.
E	get(int index) Returns the element at the specified position in this list.
int	indexOf(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
Iterator<E>	iterator() Returns an iterator over the elements in this list in proper sequence.
boolean	remove(Object o) Removes the first occurrence of the specified element from this list, if it is present. Returns true if this list contained the specified element
int	size() Returns the number of elements in this list.
default void	sort(Comparator<? super E> c) Sorts this list according to the order induced by the specified <u>Comparator</u> .

public final class String

char	charAt(int index) Returns the <u>char</u> value at the specified index.
int	compareTo(String anotherString) Compares two strings lexicographically.
static String	join(CharSequence <u>delimiter</u> , CharSequence... <u>elements</u>) Returns a new String composed of copies of the <u>CharSequence elements</u> joined together with a copy of the specified <u>delimiter</u> .
int	length() Returns the length of this string.
String[]	split(String <u>regex</u>) Splits this string around matches of the given regular expression.
String	substring(int beginIndex) Returns a string that is a substring of this string.
char[]	toCharArray() Converts this string to a new character array.
String	trim() Returns a string whose value is this string, with any leading and trailing whitespace removed.

public interface Iterator<E>

boolean	hasNext() Returns true if the iteration has more elements.
E	next() Returns the next element in the iterator .

public interface Iterable<T>

Iterator<T>	iterator() Returns an iterator over elements of type T.
-------------	--