

בחינה בתוכנה 1

סמסטר א' תשפ"ג, מועד ב', 27 בפברואר 2023
לנה דנקין, אלה גולדשמידט, אמיר ברדה

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:

1. הקוד שמופיע במבחן מתאים לגירסא Java 17.
2. כל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import בגוף הקוד.
3. כל מחלקה שהיא public מופיעה בקובץ Java משלה.
4. בכל שאלה, כל המחלקות מופיעות באותה חבילה (package).
5. 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.
6. בסוף הבחינה מופיע נספח עם תיעוד של מחלקות שאתם עשויים לעשות בהן שימוש בחלק הפתוח של הבחינה.
7. 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 לכתיבה 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:
1. בשאלות הפתוחות להשלים את הקוד החסר במקומות המסומנים ע"י מסגרת. שימו לב שלא חייבים למלא את כל המסגרות.
 2. 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 בעיבוד תמונות בגווי אפור (Grayscale). כל תמונה בנויה מריבועים קטנים הנקראים פיקסלים. תמונה בגודל x על y פיקסלים מיוצגת ע"י מטריצה בגודל x על y , כשבכל תא מופיע ערך מספרי המייצג את צבע הפיקסל. ערכי הצבעים נעים בין 0 לבין 255, כש 0 מייצג את הצבע השחור ו 255 מייצג את הצבע הלבן. כל הצבעים שבין 0 ל 255 מייצגים גוונים שונים הנעים בין שחור ללבן. ככל שהמספר קטן יותר, הגוון כהה יותר. לדוגמא, המטריצה משמאל מייצגת את התמונה מימין:

255	255	255	0	0	0
128	200	0	0	0	255
0	200	128	255	0	255
255	255	255	128	128	128



את התמונה נייצג באמצעות מערך דו מימדי של ערכים מטיפוס int, בעיקר מטעמי נוחות (ניתן היה להשתמש גם ב byte כיוון שסה"כ יש 256 ערכים אפשריים לכל תא)

הקוד שתממשו יטען את התמונה מקובץ טקסט וייצר תמונה חדשה שעברה עיבוד. פעולות העיבוד שבהן נתמוך: בינריזציה, ניקוי רעשים ופיקסול. בהמשך השאלה נסביר כל אחת מהשיטות.

נגדיר תחילה את הממשק שאותו עליכם לממש:

```
public interface ImageProcessorInterface {
    public static Image loadImage(String path){ /* section A */}
    public Image processImage(Image origImage);
}
```

השירותים הסטטי loadImage טוען את התמונה מתוך קובץ טקסט, ואותו תממשו בסעיף א'.

השירותים processImage מחזיר תמונה מעובדת המבוססת על התמונה שנטענה בעוד שהתמונה המקורית לא משתנה. תממשו גירסאות שונות שלו בסעיפים ב – ד.

המחלקה Image מוגדרת כך:

```
public record Image(int[][] arr) {
    public int rowNum() { return arr.length;}
    public int columnsNum() { return arr[0].length;}
    public int get(int row, int col) { return arr[row][col]; }
    public void set(int row, int col, int color) {
        arr[row][col] = color;
    }
}
```

ולמעשה עוטפת את המערך המדו מימדי שמייצג תמונה וחושפת פונקציות המחזירות את מספר השורות/עמודות, וכן פונקציות אשר מחזירות/מעדכנות פיקסל על פי שורה ועמודה.

להלן דוגמת שימוש סטנדרטית בממשק:

```
Image origImage = ImageProcessorInterface.loadImage("img.txt");
ImageProcessorInterface ip = new *****/;
Image processedImage = ip.processImage(origImage);
```

הקובץ img.txt מכיל את תוכן המטריצה, בפורמט שיוסבר בסעיף א'.

המשתנה ip מאוחל ע"י קריאה לבנאי של אחת המחלקות המממשות את ImageProcessorInterface. תממשו שלושה כאלה בסעיפים ב-ד.

סעיף א' לא תלוי בסעיפים ב-ד, אך סעיפים ב' עד ד' עשויים להיות תלויים זה בזה.

סעיף א' (8 נק'):

ממשו את השירות הסטטי loadImage של הממשק ImageProcessorInterface אשר מקבל נתיב לקובץ וטוען אותו למטריצה המייצגת תמונה.

השירות מקבל נתיב לקובץ אשר מכיל את התוכן הבא: השורה הראשונה מכילה 2 ערכים: מספר השורות ומספר העמודות. אם מספר השורות הוא x ומספר העמודות הוא y, אחרי השורה הראשונה יופיעו x שורות, בכל שורה y מספרים המופרדים ע"י רווח יחיד, וכולם יהיו בין 0 ל 255 (כולל הקצוות). ניתן להניח שהקובץ קיים ושפורמט הקובץ חוקי, כלומר, ניתן לטעון את המטריצה באופן תקין. שימו לב, השירות לא זורק חריגים.

דוגמת שימוש כללית עבור קובץ הקלט img.txt אשר מכיל את 3 השורות הבאות:

```
2 4
255 255 255 175
0 0 0 175
```

קובץ זה מייצג תמונה עם 2 שורות ו 4 עמודות של פיקסלים (סה"כ 8 פיקסלים).

בהפעלת הקוד הבא:

```
Image origImage = ImageProcessorInterface.LoadImage("img.txt");
```

השדה arr של origImage הוא מערך אשר מכיל 2 שורות ו 4 עמודות, ואם נדפיס את השורות שלו בזו אחר זו נקבל:

```
[255, 255, 255, 175]
```

```
[0, 0, 0, 175]
```

```
public static Image loadImage(String path){
```

```
/* reminder: given int[][] arr, use new Image(arr) to generate an
Image object. */
```

```
    int[][] image = null;
```

```
    try {
```

```
        BufferedReader bReader = new BufferedReader(new
        FileReader(path));
```

```
        String[] firstRow = bReader.readLine().split(" ");
```

```
        int rowNum = Integer.parseInt(firstRow[0]);
```

```
        int colsNum = Integer.parseInt(firstRow[1].trim());
```

```
        image = new int[rowNum][colsNum];
```

```
        String line;
```

```
        int currRow = 0;
```

```
        while((line = bReader.readLine()) != null) {
```

```
            String[] row = line.split(" ");
```

```
            for (int i = 0; i < row.length; i++) {
```

```
                image[currRow][i] = Integer.parseInt(row[i].trim());
```

```
            }
```

```
            currRow++;
```

```
        }
```

```
        bReader.close();
```

```
    }
```

```
    catch(IOException exp) {
```

```
    }
```

```
    return new Image(image);
```

סעיף ב' (9 נק'):

בסעיפים ב' – ד' נמשש שלושה ImageProcessor-ים שונים.

תחילה נגדיר את המחלקה האבסטרקטית AbstImageProcessor אשר מממשת את המנשק שהוגדר בעמוד הקודם. מחלקה זו תהווה מחלקת בסיס לכל מעבדי התמונה השונים אותם תממשו.

```
public abstract class AbstImageProcessor implements ImageProcessorInterface{
```

במחלקה אין שדות (ואין להוסיף שדות), וניתן להוסיף לה קוד משותף, במידה וקיים כזה..

השלימו את מימוש המחלקה BinarizationImageProcessor אשר מבצעת בינאריזציה, כלומר מייצרת תמונה המכילה 2 צבעים בלבד – שחור ולבן.

```
public class BinarizationImageProcessor extends /*****/ {
    private int ths;

    /* @pre: 0 <= ths <= 255 */
    public BinarizationProcessor(int ths) {
        this.ths = ths;
    }
}
```

בנאי המחלקה אשר נתון לכם מאתחל שדה בשם ths. בתמונה המתקבלת, בכל הפיקסלים שבהם הצבע המקורי היה גדול או שווה ל ths יופיע צבע לבן (255), ובכל הפיקסלים שבהם הופיע צבע הקטן מ ths יופיע צבע שחור (0).

לדוגמא: בינאריזציה עם ths=150 על התמונה השמאלית תייצר את התמונה הימנית:

255	255	255	0	0	0
128	200	0	0	0	255
0	200	128	255	0	255
255	255	255	128	128	128



255	255	255	0	0	0
0	255	0	0	0	255
0	255	0	255	0	255
255	255	255	0	0	0

השלימו את השירות processImage של מחלקה זו, אשר מבצע בינאריזציה לתמונה שנטענה.

אין להוסיף שדות למחלקה, אך ניתן להוסיף פונקציות עזר וכן להוציא קוד משותף, במידה שקיים, למחלקה AbstImageProcessor או למחלקה חדשה, אם יש צורך. לכל שירות ציינו באיזו מחלקה הוא ממומש. כמו כן, עליכם לציין מאיזו מחלקה יורשת BinarizationImageProcessor.

```
public class BinarizationImageProcessor extends PerPixelImageProcessor {
```

```
    protected int getColorForPixel(int row, int col,
        Image origImage) {
        return origImage.arr()[row][col] >= this? 255 : 0;
    }
}
```

AbstImageProcessor:

```
public Image processImage(Image origImage){
    int[][] newImageArr =
        new int[origImage.rowsNum()][origImage.columnsNum()];
    Image newImage = new Image(newImageArr);
    process(origImage, newImage);
    return newImage;
}
```

```
protected abstract void process(Image origImage, Image newImage);
```

PerPixelImageProcessor (new abstract class):

```
protected void process(Image origImage, Image newImage) {
    for (int i = 0; i < origImage.rowsNum(); i++) {
        for (int j = 0; j < origImage.columnsNum(); j++) {
            newImage.arr()[i][j] = getColorForPixel(i, j, origImage);
        }
    }
}
```

```
protected abstract int getColorForPixel(int row, int col,
    Image origImage);
```

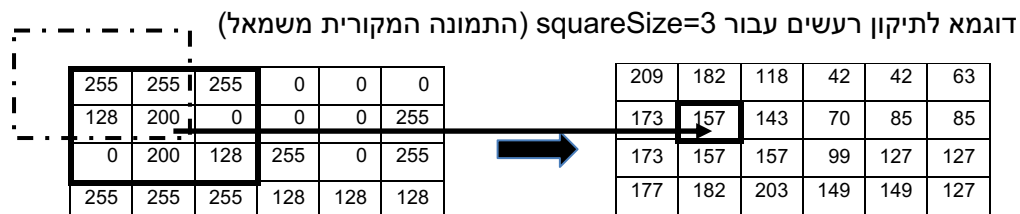
השלימו את המימוש של המחלקה DenoiseImageProcessor אשר מתקנת "רעשים" ע"י כך שהיא קובעת צבע של פיקסל על פי ממוצע צבעי הפיקסלים בסביבה שלו.

```
public class DenoiseImageProcessor extends /*****/ {

    private int squareSize;

    /* @pre: squareSize > 0
     * @pre: squareSize % 2 == 1 */
    public DenoiseImageProcessor (int squareSize) {
        this.squareSize = squareSize;
    }
}
```

אופן התיקון מבוצע כך: הפיקסל i, j מקבל את הערך הממוצע בריבוע בגודל squareSize המקיף את הפיקסל. זה יתקן רעשים כמו פיקסל מאוד כהה שנמצא באיזור מאוד בהיר. כשנסתכל על הממוצע של הפיקסלים בכל הריבוע העוטף פיקסל כהה זה, הממוצע יהיה בהיר ולכן בתמונה החדשה צבעו יתבהר. מכיוון שממוצע הוא לא דווקא שלם, עגלו אותו למעלה או למטה, כרצונכם.



הסבר: התא ה-1,1 במטריצה החדשה (צד ימין) מחושב ע"י ממוצע התאים בריבוע בגודל 3×3 המקיף את התא ה-1,1 במטריצה המקורית (שמאל).

עבור תאים שנמצאים בקצוות המטריצה, ולא ניתן להקיף אותם בריבוע מלא, נחשב ממוצע של החלק של הריבוע שנמצא בגבולות הלוח. לדוגמא, עבור התא ה-0,0 נחשב ממוצע של 4 תאים ולא של 9, כי חלק מהתאים נמצאים מחוץ ללוח – ראו הריבוע המקווקו שמדמה ריבוע בגודל 3×3 סביב התא).

אין להוסיף שדות למחלקה, אך ניתן להוסיף פונקציות עזר וכן להוציא קוד משותף, במידה שקיים, למחלקה AbstImageProcessor או למחלקה חדשה, אם יש צורך. לכל שירות ציינו באיזו מחלקה הוא ממומש. כמו כן, עליכם לציין מאיזו מחלקה יורשת DenoiseImageProcessor.

```
public class DenoiseImageProcessor extends PerPixelImageProcessor {

    protected int getColorForPixel(int row, int col, Image origImage) {
        int offset = (squareSize-1)/2;
        return getMeanColorInSquare(row-offset, row+1+offset,
                                    col-offset, col+1+offset, origImage);
    }
}
```

AbstImageProcessor:

```
protected int getMeanColorInSquare(int rowStart, int rowEnd,
    int colStart, int colEnd, Image origImage) {
    int sum = 0;
    rowStart = Math.max(0, rowStart);
    rowEnd = Math.min(rowEnd, origImage.rowsNum());
    colStart = Math.max(0, colStart);
    colEnd = Math.min(colEnd, origImage.columnsNum());
    for (int i = rowStart; i < rowEnd ; i++) {
        for (int j = colStart ; j < colEnd; j++) {
            sum += origImage.arr()[i][j];
        }
    }
    return sum/((rowEnd-rowStart)*(colEnd-colStart));
}
```

סעיף ד' (10 נק':)

השלימו את המימוש של המחלקה PixelizeImageProcessor אשר מבצעת פיקסול לתמונה, כלומר, במקום שלכל פיקסל יהיה צבע משלו, התמונה מחולקת לריבועים וכל אחד מקבל צבע אחיד.

```
public class PixelizeImageProcessor extends /***** */{

    private int squareSize;

    /* @pre: squareSize > 0
    public PixelizeImageProcessor(int squareSize) {
        this.squareSize = squareSize;
    }
}
```

בנאי המחלקה אשר נתון לכם מאתחל שדה בשם squareSize. התמונה המעובדת תחולק לריבועים בגודל squareSize על squareSize, ולכל ריבוע יהיה צבע אחיד אשר יחושב ע"י ממוצע הצבעים של אותו הריבוע בתמונה המקורית.

דוגמא לפיקסליזציה עבור squareSize=2

255	255	255	0	0	0	255
128	200	0	0	0	255	200
0	200	128	255	0	255	0
255	255	255	128	128	128	128
255	255	128	128	128	255	0



209	209	63	63	63	63	227
209	209	63	63	63	63	227
177	177	191	191	127	127	64
177	177	191	191	127	127	64
255	255	128	128	191	191	0

הסבר: גודל התמונה הוא 5 שורות על 7 עמודות. כל ריבוע בגודל 2x2 יצבע בתמונה החדשה באותו הצבע על סמך הערכים של הריבוע המקביל בתמונה המקורית. הריבוע השמאלי העליון יצבע בממוצע של 255,255,128,200 שזה 209. הריבוע השני שנמצא מימנו יצבע בממוצע של 0,0,255,0,0 (שזה 63), וכן הלאה.

עמוד 8 מתוך 20

במידה שמספר השורות/עמודות לא מתחלקים ב `squareSize` ללא שארית, צבע השורות/עמודות האחרונות יחושב ע"י ממוצע ריבועי/מלבן קטן יותר. לדוגמא, הריבוע השמאלי התחתון הוא בגודל שורה אחת ו 2 עמודות, ויצבע בצבע הממוצע של הריבוע המקביל (הממוצע של 255 ו 255 שהוא 255).

אין להוסיף שדות למחלקה, אך ניתן להוסיף פונקציות עזר וכן להוציא קוד משותף, במידה שקיים, למחלקה `AbstImageProcessor` או למחלקה חדשה, אם יש צורך. לכל שירות ציינו באיזו מחלקה הוא ממומש. כמו כן, עליכם להגדיר מאיזו מחלקה יורשת מחלקה זו.

```
public class PixalizeImageProcessor extends AbstImageProcessor {

    public void process(Image origImage, Image newImage) {
        for (int i = 0; i < origImage.rowsNum(); i += squareSize) {
            for (int j = 0; j < origImage.columnsNum(); j += squareSize) {
                int meanColor = getMeanColorInSquare(i, i+squareSize,
                                                    j, j+squareSize, origImage);
                coverArea(newImage, i, i+squareSize,
                        j, j+squareSize, meanColor);
            }
        }
    }

    private void coverArea(Image newImg, int startRow, int endRow,
                           int startCol, int endCol, int color) {
        for (int i = startRow;
             i < Math.min(newImg.rowsNum(), endRow); i++) {
            for (int j = startCol;
                 j < Math.min(newImg.columnsNum(), endCol); j++) {
                newImg.set(i, j, color);
            }
        }
    }
}
```


שאלה 2 (18 נק')סעיף א' (9 נק'):

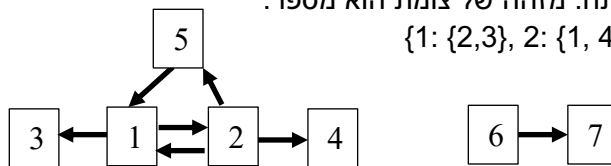
בהנתן שתי רשימות מחרוזות, $lst1$ ו $lst2$, נרצה למצוא את ה k המקסימלי שקיימות k מחרוזות שונות ב $lst2$ אשר מכילות כתתי מחרוזות תחיליות (prefixes) באורך k של מחרוזות מ $lst1$ (למשל, המחרוזת "bab" מכילה תחילית באורך 1 של המחרוזת "ac").

לדוגמא, עבור $lst1 = ["abc", "de", "fg"]$ ו $lst2 = ["a", "abfde", "fg", "kab"]$, ה k המקסימלי הוא 2. ב $lst2$ יש לפחות 2 מילים שמכילות תחיליות באורך 2 של $lst1$: המילים "abfde" ו "kab" מכילות את התחילית "ab", והמילה "fg" מכילה את התחילית "fg". לא קיימות 3 מילים שונות ב $lst2$ המכילות תחיליות באורך 3 מ $lst1$ ולכן $k=2$ הוא ה k המקסימלי.

עבור $lst1 = ["abc", "de", "fg"]$ ו $lst2 = ["ce", "e"]$, ה k המקסימלי הוא 0. גם עבור $k=1$ לא קיימת מילה ב $lst2$ המכילה תחילית באורך 1 של מילה מ $lst1$ (התחיליות הן "a", "d", "f").

```
/* @pre: lst1.size() > 0, lst2.size() > 0
 * @post: $ret >= 0*/
public static int getLargestK(List<String> lst1, List<String> lst2){
    int maxStrLength = lst1.stream().map(x->x.length())
        .reduce((x,y)->Math.max(x,y)).get();
    for (int i = 1; i < maxStrLength; i++) {
        Set<String> prefixes = new HashSet<>();
        Set<String> containingStrings = new HashSet<>();
        //get all substrings of size i
        for (String str1 : lst1) {
            if (str1.length() >= i) {
                prefixes.add(str1.substring(0, i));
            }
        }
        /* check how many strings from lst2 contains a prefix of
           size i from lst1 */
        for (String str2: lst2) {
            for (String p: prefixes) {
                if (str2.contains(p)) {
                    containingStrings.add(str2);
                    break;
                }
            }
        }
        /* we return i-1 when lst2 contains less than I prefixes of
           size i from lst1 */
        if (containingStrings.size() < i) {
            return i-1;
        }
    }
    return 0;
}
```

נניח גרף מכון ע"י מפה (Map) שבה המפתחות הם צמתים והערכים הם קבוצות (Set) הצמתים אליהם יש קשת מהצומת שמייצג המפתח. מזהה של צומת הוא מספר.
לדוגמא, המילון: $\{1: \{2,3\}, 2: \{1, 4, 5\}, 5: \{1\}, 6: \{7\}\}$ מייצג את הגרף הבא:



נרצה לבדוק האם קיים מסלול מצומת from לצומת to (הן יכולות להיות גם זהות, ואז בעצם נחפש מעגל). למשל, בגרף הנוכחי קיים מסלול מ 1 ל 5, אך לא קיים מסלול מ 4 ל 2 ולא קיים מסלול מ 3 ל 6. הפונקציה תחזיר true אם קיים מסלול בין from ל to, ו false אחרת.
הפתרון הנדרש אינו פתרון רקורסיבי, אלא פתרון איטרטיבי שמחשב בכל איטרציה את הצמתים שנגישים מ from. מכיוון שהגרף סופי, בסופו של דבר או שנגיע ל to או שנראה שזה לא אפשרי.
אם נחפש מסלול מ 1 ל 6, נתחיל בלאתר את הצמתים הנגישים מ 1, שהם 2 ו 3. לאחר מכן, נחפש את כל מה שנגיש מ 2 ו 3. מ 2 ניתן להגיע ל 1,4,5 ול 3 אין קשתות יוצאות, לכן סה"כ נבחן את 1,4,5. את 1 כבר בדקנו (משם יצאנו למעשה), כך שנותר לבחון את 4 ו 5. השכן של 5 הוא 1, ועל 1 כבר עברנו. ל 4 אין קשתות יוצאות, כך שבעצם סיימנו לעבור על כל הצמתים הנגישים מ 1, ו 6 לא נכלל ביניהם. לכן, הפונקציה תחזיר false.

```

/* @pre: graph.size() > 0 */
public static boolean checkPathExists(
    Map<Integer, Set<Integer>> graph, int from, int to){

```

```

    Set<Integer> alreadySeen = new HashSet<>();
    Set<Integer> newVerticesToExamine = new HashSet<>();
    newVerticesToExamine.add(from);
    alreadySeen.add(from);
    while (!newVerticesToExamine.isEmpty()) {
        Set<Integer> curr = new HashSet<>();
        for (Integer currVertex: newVerticesToExamine) {
            if (graph.containsKey(currVertex)) {
                curr.addAll(graph.get(currVertex));
            }
        }
        if (curr.contains(to)) {
            return true;
        }
        newVerticesToExamine.clear();
        for (Integer vertex : curr) {
            if (!alreadySeen.contains(vertex)) {
                newVerticesToExamine.add(vertex);
            }
        }
        alreadySeen.addAll(curr);
    }
    return false;
}

```

```

}

```

מסגרת חירום

שאלה 3 (5 נק'):

לפניכם 3 טענות הקשורות למנשקים:

טענה 1: שירות המוגדר כ default הוא שירות מופע שקיים עבורו מימוש במנשק.

טענה 2: ניתן להגדיר שירותים (מופע או סטטיים) בניראות שאינה public במנשק.

טענה 3: כל השדות במנשקים הם final static (בין אם הוגדרו כך מפורשות ובין אם לא).

בחר/י בתשובה הטובה ביותר:

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

נימוק:

שאלה 4 (5 נק'):

איזה משלושת השירותים יכולים להופיע ב A כך שהקוד של B ימשיך להתקמפל? בחנ/י כל שירות
בנפרד, ובחר/י בתשובה הטובה ביותר:

```
public class B extends A{  
    public String myFunc(String s) {return s;}  
}  
  
public class A{  
    //public final String myFunc(Object o) {return "a";} //f1  
    //public String myFunc(String s) throws Exception {return s;} //f2  
    //private String myFunc(String s) {return s;} //f3  
}
```

- א. רק שירות f1
- ב. רק שירות f2
- ג. רק שירות f3
- ד. רק שירות f1 או f2
- ה. רק שירות f1 או f3
- ו. רק שירות f2 או f3
- ז. עבור הוספת כל אחד מהשירותים הקוד יתקמפל.
- ח. עבור הוספת כל אחד מהשירותים הקוד לא יתקמפל.

נימוק:

שאלה 5 (5 נק'):

האם ריצת התוכנית מסתיימת, ואם כן, מה יודפס בריצת התוכנית הבאה?

```
public static void main(String[] args) {
    List<Integer> lst = Arrays.asList(2,3);
    System.out.println("# " + lst.stream()
        .filter(x->{
            System.out.print("f ");
            return x %2 != 0;
        })
        .map(x->{
            System.out.print("m ");
            return x^2;
        }).anyMatch(x -> x >10));
}
```

בחר/י בתשובה הטובה ביותר:

- א. ריצת התוכנית תסתיים ויודפס: f m f m # false
- ב. ריצת התוכנית תסתיים ויודפס: f f m m # false
- ג. ריצת התוכנית תסתיים ויודפס: f f m # false
- ד. ריצת התוכנית תסתיים ויודפס: # f m f m false
- ה. ריצת התוכנית תסתיים ויודפס: # f f m m false
- ו. ריצת התוכנית תסתיים ויודפס: # f f m false
- ז. ריצת התוכנית תסתיים ויודפס: # false
- ח. ריצת התוכנית לא תסתיים (אך יתכנו הדפסות ביניים).

נימוק:

שאלה 6 (5 נק'):

ניח כי מחלקות A ו B ממשות את מנשק I, כל אחת בדרך אחרת. המנשק I מגדיר פונקציה אחת func אשר מקבלת 5 פרמטרים.

- א. תבנית העיצוב Bridge מתארת מצב שבו מחלקה חדשה X צריכה לרשת מ B וגם מ A, אך בהעדר ירושה מרובה X תכיל שדה מטיפוס A וגם שדה מטיפוס B ובנוסף תממש את I.
- ב. תבנית העיצוב Bridge מתארת מצב שבו מחלקה חדשה X צריכה לרשת מ B וגם מ A, אך בהעדר ירושה מרובה X תירש ממחלקה A ותכיל שדה מטיפוס B.
- ג. תבנית העיצוב Bridge מתארת מצב שבו מחלקה חדשה X צריכה לייצר אובייקט חדש מטיפוס I מבלי להכיר את A או את B, ולכן היא עושה שימוש במחלקה שלישית שמכירה את A ואת B ויודעת לייצר אובייקטים מטיפוסים אלה.
- ד. תבנית העיצוב Bridge מתארת מצב שבו מחלקה חדשה X צריכה לייצר אובייקט חדש מטיפוס I מבלי להכיר את A או את B, ולכן היא משכפלת אובייקטים קיימים מטיפוס סטטי I מבלי להכיר את טיפוס זמן הריצה שלהם.
- ה. תבנית העיצוב Bridge מתארת מצב שבו X היא מחלקה חדשה אשר יורשת מ A ורוצה לחשוף למשתמש פונקציה בשם func שתקבל רק 2 פרמטרים במקום 5. לכן X משתמשת בהעמסה (overloading) בשביל לממש פונקציית "גשר" – פונקציה בשם func המקבלת 2 פרמטרים וקוראת ל func של A על 2 הפרמטרים שהתקבלו ועוד 3 פרמטרים קבועים.
- ו. מלבד תשובה זו כל התשובות לא נכונות.

נימוק:

שאלה 7 (5 נק'):

לפניכם הקוד של המחלקה MyClass וכן פונקציית main הממומשת במחלקה אחרת:

```
public class MyClass<T extends List<String>>{  
    private T t;  
  
    //public T func() { return new T(); }  
}
```

```
public static void main(String[] args) {  
    MyClass<String> mC = new MyClass<>();  
}
```

טענה 1: לאחר הפעלת מנגנון מחיקת הטיפוסים (erasure), הטיפוס של השדה t יהפוך ל List<String>.

טענה 2: אם נוציא את הפונקציה func מהערה, הקוד של MyClass ימשיך להתקמפל.

טענה 3: הפונקציה main מתקמפלת.

בחר/י בתשובה הטובה ביותר:

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

נימוק:

שאלה 8 (5 נק'):

לפניכם שלוש טענות על חריגים. אילו מהטענות נכונות? בחר'י בתשובה הטובה ביותר:

טענה 1: שירות יכול להצהיר (ע"י throws) על זריקת יותר מחריג אחד.

טענה 2: אין צורך להצהיר (ע"י throws) על זריקת חריג מסוג un-checked.

טענה 3: ניתן לממש חריגים חדשים שהם checked, אך לא ניתן לממש חריגים חדשים שהם un-checked.

א. רק טענה 1 נכונה.

ב. רק טענה 2 נכונה.

ג. רק טענה 3 נכונה.

ד. רק טענות 1+2 נכונות.

ה. רק טענות 1+3 נכונות.

ו. רק טענות 2+3 נכונות.

ז. כל הטענות נכונות.

ח. כל הטענות לא נכונות.

נימוק:

שאלה 9 (5 נק'):

בחר'י בתשובה הטובה ביותר:

א. ניתן לרשת מ enum.

ב. לא ניתן להוסיף שירותי מופע ל enum.

ג. לא ניתן להוסיף שירותים אבסטרקטיים ל enum.

ד. ניתן לעשות שימוש ב enum בבילוק switch/case.

ה. ב enum ניתן להגדיר שדות מופע משני סוגים: שדות מופע שהם final ושדות מופע שאינם final.

ו. מלבד תשובה זו, כל התשובות לא נכונות.

ז. מלבד תשובה זו יש יותר מתשובה נכונה אחת.

נימוק:

הערה - בשאלה זו קיבלנו גם את תשובה ד' כנכונה, כיוון שלא הראינו בכיתה שניתן להגדיר שדות מופע שאינם final ב enum (זה לא רצוי, אך אפשרי).

```

public class Q10 {
    public static interface I<T>{
        public boolean pred(T t);
    }

    public static class MyIterator<T> implements Iterator<T>{
        Iterator<T> it1;
        Iterator<T> it2;
        I<T> pred;

        public MyIterator(Iterator<T> it1, Iterator<T> it2, I<T> pred) {
            this.it1 = it1;
            this.it2 = it2;
            this.pred = pred;
        }

        public boolean hasNext() {
            return it1.hasNext() && it2.hasNext();
        }

        public T next() {
            T curr1 = it1.next();
            T curr2 = it2.next();
            while (!pred.pred(curr1) && it1.hasNext()) {
                curr1 = it1.next();
                curr2 = it2.next();
            }
            return curr2;
        }
    }

    public static void main(String[] args) {
        Iterator<Integer> it1 = Arrays.asList(1,2,1,4,5).iterator();
        Iterator<Integer> it2 =
            Arrays.asList(11,12,13,14,15,16).iterator();
        Iterator<Integer> it = new MyIterator<>(it1, it2, x-> x%2 == 0);
        while(it.hasNext()) {
            System.out.println(it.next());
        }
        System.out.println(it2.next()); // #
    }
}

```

מה יקרה בהרצת הקוד? בחר/י בתשובה הטובה ביותר:

- א. בלולאת ה while יודפסו 2 איברים ואז תיזרק שגיאה, כך שהקוד לא יגיע לשורה #.
- ב. בלולאת ה while יודפסו 2 איברים, הקוד יבצע את שורה # ובשורה # תיזרק שגיאה.
- ג. בלולאת ה while יודפסו 2 איברים, הקוד יבצע את שורה # ובשורה # יודפס האיבר 16.
- ד. בלולאת ה while יודפסו 2 איברים, הקוד יבצע את שורה # ובשורה # יודפס האיבר 11.
- ה. בלולאת ה while יודפסו 3 איברים ואז תיזרק שגיאה, כך שהקוד לא יגיע לשורה #.
- ו. בלולאת ה while יודפסו 3 איברים, הקוד יבצע את שורה # ובשורה # תיזרק שגיאה.
- ז. בלולאת ה while יודפסו 3 איברים, הקוד יבצע את שורה # ובשורה # יודפס האיבר 16.
- ח. בלולאת ה while יודפסו 3 איברים, הקוד יבצע את שורה # ובשורה # יודפס האיבר 11.

נימוק:

שאלה 11 (5 נק':)

```
public class Q11 {
    private String[] arr1 = new String[3];

    public void func() {
        int[] arr2;
        int[] arr3 = {1,2,3};
        foo(arr3);
    }

    public void foo(int[] arr) {
        // code here
    }
}
```

לפניכם 3 טענות על הקוד המצורף. הטענות שמתייחסות למיקום שדות ומשתנים בזכרון מתייחסות למיקום שדות והמשתנים עצמם, לא למיקום הערכים עליהם הם עשויים להצביע.

טענה 1: השדה arr1 נשמר על ה heap ומאותחל למערך בגודל 3, כך שבכל התאים שלו מופיע הערך null.

טענה 2: המשתנה arr2 נשמר על ה stack ומאותחל ל null.

טענה 3: ניתן לממש את foo כך שלאחר הפעלת foo על arr3 בתוך המשתנה arr3 תופיע כתובת שונה מזו שהופיעה לפני הפעלת הפונקציה.

א. רק טענה 1 נכונה.

ב. רק טענה 2 נכונה.

ג. רק טענה 3 נכונה.

ד. רק טענות 1+2 נכונות.

ה. רק טענות 1+3 נכונות.

ו. רק טענות 2+3 נכונות.

ז. כל הטענות נכונות.

ח. כל הטענות לא נכונות.

נימוק:

```
public class Base{
    public int sum = 0;
    public int i = 2;

    public Base(){
        sum = i + 2;
    }
    public int getNum() {
        return i + this.foo();
    }
    public int foo() {
        return 1;
    }
}

public class Sub extends Base{
    public int i = 4;

    public Sub() {
        sum = sum + ((Base)this).i + ((Base)this).getNum();
    }

    public int getNum() {
        return i;
    }

    public int foo() {
        return 2;
    }

    public static void main(String[] args) {
        Base b = new Sub();
        System.out.println(b.sum);
    }
}
```

מה יודפס בהרצת השירות main של Sub?

- א. 7
- ב. 8
- ג. 9
- ד. 10
- ה. 11
- ו. 12
- ז. 13
- ח. 14

נימוק:

public interface Map<K,V>

Modifier and Type	Method and Description
Boolean	containsKey(Object key) Returns true if this map contains a mapping for the specified key.
V	get(Object key) Returns the value to which the specified key is mapped, or <u>null</u> if this map contains no mapping for the key.
V	getOrDefault(Object key, V defaultValue) Returns the value to which the specified key is mapped, or <u>defaultValue</u> if this map contains no mapping for the key.
Set<K>	keySet() Returns a <u>Set</u> view of the keys contained in this map.
V	put(K key, V value) 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	remove(Object key) 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.
Collection<V>	values() Returns a Collection view of the values contained in this map.

public interface Set<E> extends Collection<E>

boolean	add(E e) Adds the specified element to this set if it is not already present.
boolean	addAll(Collection<? Extends E> c) Adds all of the elements in the specified collection to this set if they're not already present. Returns true if this set changed as a result of the call.
void	clear() Removes all of the elements from this set
boolean	contains(Object o) Returns true if this set contains the specified element.
boolean	isEmpty() Returns true if this set contains no elements.
boolean	remove(Object o) Removes the specified element from this set if it is present.
boolean	removeAll(Collection<?> c) Removes from this set all of its elements that are contained in the specified collection
Boolean	retainAll(Collection<?> c) Retains only the elements in this set that are contained in the specified collection

public interface List<E> extends Collection<E>

boolean	add(E e) Appends the specified element to the end of this list. Always returns true.
void	add(int index, E e) Inserts the specified element at the specified position in this list.

עמוד 20 מתוך 20

boolean	<code>contains(Object o)</code> Returns true if this list contains the specified element.
E	<code>get(int index)</code> Returns the element at the specified position in this list.
boolean	<code>remove(Object o)</code> Removes the first occurrence of the specified element from this list, if it is present. Returns true if this list contained the specified element

public final class String

char	<code>charAt(int index)</code> Returns the char value at the specified index.
boolean	<code>contains(CharSequence s)</code> Returns true if and only if this string contains the specified sequence of char values.
int	<code>length()</code> Returns the length of this string.
String	<code>replace(CharSequence target, CharSequence replacement)</code> Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
String[]	<code>split(String regex)</code> Splits this string around matches of the given regular expression.
String	<code>substring(int beginIndex, int endIndex)</code> Returns a string that is a substring of this string. Throws <code>IndexOutOfBoundsException</code> - if the <code>beginIndex</code> is negative, or <code>endIndex</code> is larger than the length of this <code>String</code> object, or <code>beginIndex</code> is larger than <code>endIndex</code> .
char[]	<code>toCharArray()</code> Converts this string to a new character array.

public class BufferedReader

	<code>BufferedReader(Reader in)</code> Creates a buffering character-input stream that uses a default-sized input buffer
void	<code>close()</code> Closes the stream and releases any system resources associated with it.
int	<code>read()</code> Reads a single character. Returns the character read, as an integer, or -1 if the end of the stream has been reached. Throws <code>IOException</code> if an I/O error occurs.
String	<code>readLine()</code> Reads a line of text. Returns a <code>String</code> containing the contents of the line, not including any line-termination characters, or null if the end of the stream has been reached. Throws <code>IOException</code> if an I/O error occurs.

public class FileReader (extends Reader)

	<code>FileReader(File file)</code> Creates a new <code>FileReader</code> , given the <code>File</code> to read from. Throws <code>FileNotFoundException</code> (extends <code>IOException</code>) if the file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.
--	---