

בחינה בתוכנה 1

סמסטר א תשפ"ב, מועד ב', 11 בפברואר 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 לגרסה Java8.
- 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 לכתיבה נכונה ב Java.

מבנה הבחינה:

- הבחינה מורכבת משני חלקים: חלק פתוח (שתי שאלות על סך 50 נקודות) ושאלות אמריקאיות (11 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 (33 נק')

בשאלה זו נעסוק במימושים שונים של מטריצות דלילות המאפשרות מימוש יעיל יחסית של פעולות על שורות. מטריצות דלילות הן מטריצות שרוב התאים שלהן מכילים את האיבר 0, ולכן נרצה לייצג אותן במבנה נתונים אשר שומר רק את המספרים ששונים מ 0.

בשאלה זו, המטריצות הן אמנם דלילות, אבל קיים לפחות ערך אחד שונה מאפס בכל שורה. בנוסף, כל הפעולות שנבצע על המטריצות הן פעולות לקריאה בלבד, כלומר, לא נשנה את המטריצה לאחר יצירתה.

תחילה, נגדיר את הממשק שמייצג מטריצה כזו:

```
public interface ISparseRowMatrix {

    /* @post: $ret > 0 */
    public int getRowsNum();

    /* @post: $ret > 0 */
    public int getColsNum();

    /* @pre: 0 <= row < getRowsNum()
    /* @post: $ret > 0 */
    public int getNumNonZerosInRow(int row);

    /* @pre: 0 <= row < getRowsNum()
    * @pre: 0 <= col < getColsNum()*/
    public float get(int row, int column);

    /* @pre: 0 <= row < getRowsNum()
    * @post: for each 0 <= i < getColsNum():
    *         get(row, i) <= $ret
    * @post: there exist i such that get(row,i) = $ret*/
    public float maxInRow(int row);
}
```

השירותים `getRowsNum` ו `getColsNum` יחזירו את מספרי השורות\עמודות במטריצה.

השירות `getNumNonZerosInRow` יחזיר את מספר האיברים ששונים מ 0 בשורה מסויימת.

השירות `get` יחזיר את הערך הנמצא במיקום מסויים במטריצה, על פי שורה ועמודה.

השירות `maxInRow` יחזיר את הערך המקסימלי מבין כל האיברים בשורה (אפסים ולא אפסים).

את המטריצה נייצג ב 2 דרכים ע"י מימוש 2 מחלקות:

א. `MapMatrix` – מימוש ע"י תחזוקת `Map`.

ב. `CSRMatrix` – מימוש ע"י תחזוקת שלושה מערכים.

המימוש שלנו יחולק לשני חלקים: סעיפים א+ב (התלויים זה בזה) יממשו את כל הפונקציות שמגיעות מהממשק. בסעיפים ג' וד' נטפל בבניית המטריצות מתוך קובץ.

בכל הסעיפים: את הקוד המשותף, במידה שקיים, עליכם להוסיף למחלקה האבסטרקטית `AbstractMatrix` שממנה יורשות שתי המחלקות. הוסיפו שדות ופונקציות עזר במידת הצורך, והקפידו על ניראות מתאימה.

ממשו את המחלקה MapMatrix אשר עושה שימוש ב Map. השדה innerMap כבר מוגדר עבורכם. הוסיפו שדות נוספים אם צריך וממשו את כל הפונקציות של המנשק (יש סה"כ 5 פונק'). אין צורך להגדיר בנאים. מומלץ מאוד לעבור גם על סעיף ב' לפני שמממשים את סעיף א', על מנת לטפל בקוד המשותף, במידה וקיים.

```
public abstract class AbstractMatrix implements ISparseRowMatrix{
```

```
//members here
    protected int rowNum;
    protected int colNum;
```

```
//the rest of the code here
```

```
    public int getRowsNum() {
        return rowNum;
    }

    public int getColsNum() {
        return colNum;
    }

    public float maxInRow(int row){
        float maxNonZeros = getMaxNonZeros(row);
        if (getNumNonZerosInRow(row) < colNum){
            return Math.max(maxNonZeros, 0);
        }
        return maxNonZeros;
    }

    public abstract float getMaxNonZeros(int row);
```

```
}
```

```
public class MapMatrix extends AbstractMatrix {
```

```
    private Map<Integer, Map<Integer, Float> innerMap;
```

```
//more members here
```

```
//the rest of the code here
```

```
    public float get(int row, int column) {
        // it's guaranteed that row exists in innerMap
        return innerMap.get(row).getOrDefault(column, 0.f);
    }

    public int getNumNonZerosInRow(int row) {
        return innerMap.get(row).size();
    }
```

```

    }

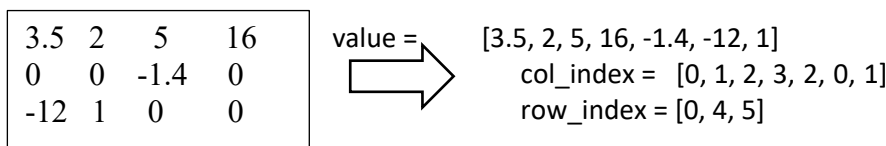
    public float getMaxNonZeros(int row) {
        return Collections.max(innerMap.get(row).values());
    }
}
}
}

```

סעיף ב' (8 נק'): _____

ממשו את המחלקה CSRMatrix אשר מייצגת מטריצה דלילה ע"י 3 מערכים באופן הבא:

ניקח כדוגמא את המטריצה המופיעה בצד שמאל ומיוצגת ע"י שלושת המערכים המופיעים מימין:



המטריצה היא בגודל 3 שורות על 4 עמודות. גודלם של המערכים value ו col_index הוא זהה ושווה למספר האיברים ששונים מ 0 במטריצה. במקרה שלנו יש 7 כאלה. גודל row_index הוא כמספר השורות (3). המערך value מכיל את כל הערכים השונים מ 0, בזה אחר זה. המערך col_index מכיל את מספרי העמודות המתאימים לכל ערך ב-value. לדוגמא: הערך הראשון ב value הוא המספר 3.5. הוא מופיע בעמודה 0, ולכן הערך הראשון ב col_index יהיה 0. הערך הבא ב value הוא 2. הוא מופיע בעמודה עם האינדקס 1, ולכן ב col_index יופיע במקום השני המספר 1. המערכים value ו col_index לא מכילים מידע על השורות, והמידע הזה נשמר ב row_index. הטבלה הבאה מכילה את הערכים של value ו col_index, כשהשורה הראשונה היא השורה של האינדקסים במערך.

	(*) 0	1	2	3	(*) 4	(*) 5	6
value	3.5	2	5	16	-1.4	-12	1
col_index	0	1	2	3	2	0	1

הערכים שבין אינדקס 0 ל 3 (כולל) מכילים את המידע של השורה הראשונה במטריצה. החל מאינדקס 4 מתחילה השורה השניה (המספר היחיד שאינו 0 בשורה זו הוא -1.4), והחל מאינדקס 5 מתחילה השורה השלישית. במילים אחרות, האינדקסים 0, 4 ו 5 value וב col_index (מסומנים ב *) מתחילים שורה חדשה, וזה בדיוק המידע שנשמר ב row_index. פורמלית: row_index במקום ה i מכיל את האינדקס j כך ש value[j] ו col_index[j] מכילים את הערך והעמודה של האיבר הראשון שאינו 0 בשורה i.

השלימו את המימוש של CSRMatrix. הוסיפו שדות ושירותי עזר במידת הנדרש והשלימו את מימוש הפונקציות שמגדיר המנשק. שימו לב – בסעיף זה עליכם לעבוד עם מערכים בלבד. אין לייצר מבני נתונים מטיפוס Collection (List, Map וכיו) בשום שלב של ריצת הקוד.

```
public class CSRMatrix extends AbstractMatrix{
```

```
    private float[] values;
    private int[] colIndex;
    private int[] rowIndex;
```

```
// more members here
```

```
//the rest of the code here
private int rowEndIndex(int row){
    return row < getRowsNum() - 1? rowIndex[row+1] : colIndex.length;
}

public int getNumNonZerosInRow(int row) {
    return rowEndIndex(row) - rowIndex[row];
}

public float get(int row, int column) {
    for(int i = rowIndex[row]; i < rowEndIndex(row); i++){
        if (colIndex[i] == column){ return values[i]; }
    }
    return 0;
}

public float getMaxNonZeros(int row) {
    float max = Float.MIN_VALUE;
    for(int i = rowIndex[row] ; i < rowEndIndex(row); i++){
        max = Math.max(max, values[i]);
    }
    return max;
}
}
```

סעיף ג' (12 נק'): _____

בסעיפים ג' וד' נטפל בבניית מטריצה ע"י טעינה מקובץ. בסעיף ג' נממש את הטעינה עבור CSRMatrix בלבד, ובסעיף ד' נבחן כיצד אפשר להתאים את הקוד כך לפחות חלקו יהיה שימוש עבור טעינה של מטריצות מטיפוס MapMatrix.

את המטריצה נטען על פי מידע המופיע בשני קבצים: קובץ config וקובץ mat. בקובץ config תופיע שורה אחת עם שלושה מספרים: הראשון מייצג את מספר השורות, השני את מספר העמודות והשלישי את מספר האיברים ששונים מ 0. לדוגמא:

```
2 4 3
```

בקובץ mat תופיע במלואה (כולל אפסים) המטריצה המתאימה ל config. לדוגמא:

```
0 0 0 1
0 -2 0 2.5
```

בדוגמא המובאת, הקובץ mat מכיל מטריצה 2 שורות ו 4 עמודות, ובדיוק 3 ערכים שאינם 0.

מספר זהות: _____ מספר מחברת: _____ עמוד 6 מתוך 20

ממשו את הבנאי של CSRMatrix המקבל 2 פרמטרים: שם קובץ config ושם קובץ mat וטוען את המטריצה בקוד.

- א. עליכם לאכלס את המטריצות תוך כדי הקריאה מהקובץ. בכל רגע ניתן לשמור בזכרון (בנוסף למטריצה שאתם בונים) רק את תוכנה של שורה בודדת שנקראה מהקובץ.
- ב. העזרו בפונקציה Float.parseFloat המקבלת מחרוזת שמכילה מספר (למשל, "1.5" או "3") ומחזירה את ערכו כ float. בנוסף, תוכלו למצוא בנספח תיעוד של FileReader ו BufferedReader.
- ג. הפונקציה תזרוק שגיאה מטיפוס MatrixLoadException בכל מקרה של שגיאה בעבודה עם הקבצים (קובץ לא נמצא וכו'). שימו לב שהחריג הוא unchecked.
- ד. הוסיפו פונקציות עזר במידת הצורך ועדכנו את AbstractMatrix במידת הצורך.
- ה. המימוש בסעיף זה מתייחס אך ורק לטעינה של מטריצה מטיפוס CSRMatrix, ולא חייב להיות מותאם לשימוש חוזר ב MapMatrix. בסעיף ד' תידרשו להציע התאמות שיאפשרו שימוש חוזר בקוד גם עבור MapMatrix.
- ו. הקלט הוא חוקי. mat ו config מכילים נתונים של מטריצה דלילה עם לפחות איבר אחד ששונו מ 0 בכל שורה. השגיאות היחידות בהן עליכם לטפל קשורות לעבודה עם הקבצים.

```
public class MatrixLoadException extends RuntimeException{ //unchecked
    public MatrixLoadException(String fileName){
        super("failed to load matrix from " + fileName);
    }
}
```

```
public abstract class AbstractMatrix implements ISparseRowMatrix{

    protected int nonZeros;

    public AbstractMatrix(String configFile){
        BufferedReader br = null;
        try{
            br = new BufferedReader(new FileReader(configFile));
            String[] words = br.readLine().split(" ");
            rowNum = Integer.parseInt(words[0]);
            colNum = Integer.parseInt(words[1]);
            nonZeros = Integer.parseInt(words[2]);
            br.close();
        }
        catch (Exception exp){
            throw new MatrixLoadException("Error loading config file");
        }
    }
}
```

```
public class CSRMatrix extends AbstractMatrix{

    public CSRMatrix(String pathToConfig, String pathToMat){
        super(pathToMath);
        values = new float[this.nonZeros];
        colIndex = new int[this.nonZeros];
        rowIndex = new int[this.rowNum];
        BufferedReader br = null;
        try{
            br = new BufferedReader(new FileReader(pathToMath));
```

```

        int colIndexInd = 0;
        for (int row= 0; row < this.rowNum; row++){
            rowIndex[row] = colIndexInd;
            String[] wordStr = br.readLine().split(" ");
            for (int col = 0 ; col < this.getColsNum(); col++){
                float num = Float.parseFloat(wordStr[col]);
                if (num != 0){
                    colIndex[colIndexInd] = col;
                    values[colIndexInd++] = num;
                }
            }
        }
        br.close();
    }
    catch (Exception exp){
        throw new MatrixLoadException("Error loading mat file");
    }
}

```

סעיף ד' (5 נק')::

נרצה לממש בנאי דומה לבנאי של סעיף ג' גם עבור MapMatrix.

הציעו רעיון עבור שיתוף מיטבי של הקוד בין הבנאים של MapMatrix ושל CSRMatrix. אין צורך לממש את הרעיון במלואו, אך עליכם עליכם לתאר את הרעיון בקצרה, להסביר מה החלק המשותף ומה החלק השונה, ולתאר את השירותים\שדות\מחלקות שיתווספו.

מלבד שיתוף הקוד של קריאת קובץ ה config ניתן לשתף גם את הקוד של הקריאה מ mat.

אפשרות אחת היא יצירת מחלקה שהיא סוג של איטרטור לערכי המטריצה. היא יודעת להחזיר בכל פעם את הזוגות <ערך, מספר עמודה> שאינם 0 בכל שורה. בסוף השורה ניתן יהיה לקדם את האיטרטור לשורה הבאה.

אפשרות נוספת היא להוסיף פונקציות מופשטות מעדכנות את הערך הבא שאינו 0 בכל שורה כשכל הלוגיקה של הקריאה מקובץ תופיע במחלקה האבסטרקטית. המחלקה האבסטרקטית תקרא לפונק' שמאתחלת שורה חדשה, ובכל שורה תקרא לפונק' שעושה set לערך מסויים בעמודה מסויימת. זה יצריך גם הוספת שדות שישמרו את מספר השורה הנוכחית שמאכלסים (בשתי המחלקות) ואת האינדקסים שבהם נמצאים ב colIndex/values ב CSRMatrix

שאלה 2 (17 נק'):

סעיף א' (9 נק'):

בסעיף זה עליכם לממש את המחלקה InterleaveIt אשר מתארת איטרטור המשלב מספר איטרטורים, ומחזיר בכל פעם איבר של איטרטור אחר. לדוגמא: עבור שלושה איטרטורים:

```

Iterator<Integer> it1 = Arrays.asList(1,2,3).iterator();
Iterator<Integer> it2 = Arrays.asList(5).iterator();
Iterator<Integer> it3 = Arrays.asList(6,7,8,9,10).iterator();
List<Iterator<Integer>> itsList = Arrays.asList(it1, it2, it3);
Iterator<Integer> interleaveIt =
    new InterleaveIterator<>(itsList);
while (interleaveIt.hasNext()){
    System.out.println(interleaveIt.next());
}
    
```

פעולת ה interleave על שלושת האיטרטורים תחזיר את האיברים בסדר הבא (סה"כ 9 איטרציות):

1,5,6,2,7,3,8,9,10

שלושת האיברים הראשונים שיחזרו הם שלושת האיברים הראשונים שמחזיר כל אחד מהאיטרטורים (1,5,6). מכיוון שאחרי איטרציה אחת it2 לא יחזיר עוד איברים, נמשיך לקבל איברים רק מ it1 ו it2. אחרי 3 איטרציות גם אברי it1 יסתיימו, ואז נחזיר את האיברים הנותרים ב it3 (9,10).

השלימו את המימוש של InterleaveIt על פי דוגמת הריצה.

הנחיה: אין לייצר מראש את כל האיברים ש InterleaveIt צריך להחזיר. את קבלת האיבר הבא מכל איטרטור יש לעשות רק כ InterleaveIt צריך להחזיר אותו ב next. פתרונות שמייצרים רשימה של כל האיברים לפי הסדר "הנכון", ואז עוברים עליה ב next/hasNext יקבלו ניקוד חלקי בלבד.

```
public InterleaveIt<T> implements Iterator<T>{
```

```

//members here
List<Iterator<T>> iterators;
int currIndex = 0;
    
```

```
public InterleaveIt(List<Iterator<T>> iterators ){
```

```

    this.iterators = iterators;
    while(currIndex < iterators.size() &&
        !iterators.get(currIndex).hasNext()){
        this.currIndex++;
    }
}
    
```

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

```
    return iterators.get(currIndex).hasNext();
```

```
}
```



```

public T next() {
    T nextValue = iterators.get(currIndex).next();
    for (int i = 1; i <= iterators.size(); i++){
        int tmpIndex = currIndex+i % iterators.size();
        if (iterators.get(tmpIndex).hasNext()){
            currIndex = tmpIndex;
            break;
        }
    }
    return nextValue;
}
}

```

סעיף ב' (8 נק'):

תאומים ראשוניים הם זוג מספרים ראשוניים שההפרש ביניהם הוא 2. למשל: (11,13), (17,19), (29,31). ממשו את הפונקציה printPrimeTwins אשר מדפיסה את כל התאומים הראשוניים עד k, כולל k עצמו. כל זוג יודפס בשורה נפרדת, פורמט ההדפסה הוא לבחירתכם.

הנחיה – מכיוון שבדיקת ראשוניות של מספר היא בדיקה יקרה, הקפידו לבדוק ראשוניות של כל מספר פעם אחת בלבד. את בדיקת הראשוניות יש לממש בפונקציה isPrime.

```

public void printPrimeTwins(int k){
    int prevPrime = 1;
    int currNum = 2;
    for (currNum = 1 ; currNum <= k; currNum++){
        if (isPrime(currNum)){
            if (currNum - prevPrime == 2){
                System.out.println(currNum + "," + prevPrime);
            }
            prevPrime = currNum;
        }
    }
}

public static boolean isPrime(int num){
    if (num == 1 || num == 2){
        return true;
    }
    for (int i = 2; i < Math.sqrt(num) + 1; i++){
        if (num % i == 0){
            return false;
        }
    }
    return true;
}
}

```

מסגרת חירום:

שאלה 3:

שאלות 3-4 מתייחסות לקוד הבא: נרצה לממש משחק מילים שבו צריך לנחש מילה שחלק מאותיותיה מוסתרות על פי תבנית הסתרה מסויימת, כאשר תבנית מיוצגת ע"י מערך של pattern. עבור מילה word ותבנית pattern האות במקום ה i של word תוסתר אם pattern[i]=true. מכיוון שלא כל התבניות הן חוביות נממש את הפונקציה checkLegalPattern אשר בודקת אם תבנית הסתרה מסויימת היא חוקית עבור מילה.

בשאלה זו עליכם להבין מה הקוד עושה על שלושת התבניות המופיעות ב main. לא הגדרנו מפורשות מהי תבנית חוקית, כיוון שהדרישה היא להבין מה הקוד עושה.

תזכורת – האותיות a-z מופיעות ברצף בטבלת ה ascii.

```

/* @pre: word != null and pattern != null */
public static boolean checkLegalPattern(boolean[] pattern,
                                       String word) {
    boolean[] masked = new boolean[26]; //26 English letters
    boolean[] unmasked = new boolean[26];
    int cntHidden = 0;
    for (int i = 0; i < word.length(); i++) {
        int currCharIndx = word.charAt(i)-'a';
        if (pattern[i] == true) {
            if (unmasked[currCharIndx]) {
                return false;
            }
            masked[currCharIndx] = true;
            cntHidden++;
        }
        else {
            if (masked[currCharIndx]) {
                return false;
            }
            unmasked[currCharIndx] = true;
        }
    }
    return cntHidden < word.length();
}

public static void main(String[] args) {
    boolean[] p1 = new boolean[] {false, false, true, false};
    boolean[] p2 = new boolean[] {true, false, false, true};
    boolean[] p3 = new boolean[] {false, false, false, false};

    System.out.println(checkLegalPattern(p1, "tilt"));
    System.out.println(checkLegalPattern(p2, "tilt"));
    System.out.println(checkLegalPattern(p3, "tilt"));
}

```

לפניכם 3 טענות על הפונקציה main:

- טענה 1: עבור התבנית p1 יודפס true בקריאה ל checkLegalPattern
- טענה 2: עבור התבנית p2 יודפס true בקריאה ל checkLegalPattern
- טענה 3: עבור התבנית p3 יודפס true בקריאה ל checkLegalPattern

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:

נרצה להוסיף לפונקציה תנאי pre בנוסף לתנאי המופיע בשאלה 3 בשביל למנוע שגיאות זמן ריצה מהקוד. לפניכם 3 אופציות:

```
/* @pre1: word.length() <= pattern.length
 * @pre2: for 0 <= i < word.length:
 *         'a' <= word.charAt[i] <= 'z'
 * @pre3: word.length() != 0
 * */
```

נרצה לבחון אילו מבין התנאים יש להוסיף על מנת למנוע שגיאות זמן ריצה של הקוד הקשורות לקוד (למשל, NullPointerException, גישה לאינדקס לא חוקי במערך וכו'). בשאלה זו אנחנו יוצאים מנקודת הנחה שהשימוש בפונקציה נעשה אך ורק על פי החוזה.

בחר/י בתשובה הטובה ביותר:

- א. הקוד ירוץ ללא שגיאות גם בלי הוספת תנאי pre החדשים.
- ב. רק אם נוסיף את תנאי pre1 הקוד ירוץ ללא שגיאות ללא צורך בתנאים נוספים.
- ג. רק אם נוסיף את תנאי pre2 הקוד ירוץ ללא שגיאות ללא צורך בתנאים נוספים.
- ד. רק אם נוסיף את תנאים pre1 ו pre2 הקוד ירוץ ללא שגיאות ללא צורך בתנאים נוספים.
- ה. רק אם נוסיף את תנאים pre1 ו pre3 הקוד ירוץ ללא שגיאות ללא צורך בתנאים נוספים.
- ו. רק אם נוסיף את תנאים pre2 ו pre3 הקוד ירוץ ללא שגיאות ללא צורך בתנאים נוספים.
- ז. אם נוסיף את תנאים pre1, pre2, pre3 הקוד ירוץ ללא שגיאות ללא צורך בתנאים נוספים.
- ח. גם אם נוסיף את תנאים pre1, pre2, pre3 הקוד עדין יכול לייצר שגיאת זמן ריצה.

נימוק:

```

public class A{
    public String func(String str) throws Exception{
        return str+"a";
    }
}

public class B extends A{
    /* // implementation 1
    public String func(String str) {
        return super.func(str +"x");
    }*/

    /* // implementation 2
    public String func(Object str) throws Exception{
        return "b";
    }*/

    public static void main(String[] args) throws Exception{
        A a = new B();
        System.out.println(a.func("5"));
    }
}
    
```

לפניכם מימוש של שתי מחלקות A ו B. נרצה לבחון איך הקוד פועל ואיך הוא יתנהג אם נוציא כל אחד משני המימושים של func ב B מהערה.

טענה 1: הקוד כמו שהוא מתקמפל וידפיס 5a

טענה 2: אם נוציא את המימוש הראשון של func מהערה, הקוד יתקמפל וידפיס 5xa.

טענה 3: אם נוציא את המימוש השני של func מהערה, הקוד יתקמפל וידפיס b.

בחר/י את התשובה 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 6:

```
public class A{
    int i;
    public void setI(int i) {this.i = i;}
    private void func() {}
}

public class B extends A{
    private void func() {}
}

public class C extends A{
    private void statFunc() {
        A a = new A();
        a.setI(13);
    }
}
```

לפניכם 3 טענות על הקוד המצורף:
טענה 1: אם נשנה את הניראות של i ל protected, יתכן שקוד מסויים שעושה שימוש ב i והתקמפל לפני השינוי יפסיק להתקמפל אחריו.
טענה 2: המחלקה B לא מתקמפלת.
טענה 3: המחלקה C לא 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 7:

לפניכם שאלה על שדות מופע (instance members). נניח כי במחלקה X קיים שדה מופע m בניראות public. בחר\י בתשובה הטובה ביותר:

- א. m מאותחל עם יצירת מופע מטיפוס X והוא נגיש רק ממתודות מופע.
- ב. m מאותחל עם יצירת מופע מטיפוס X והוא נגיש ממתודות מופע ומתודות סטטיות שמכילות משתנה מטיפוס סטטי X.**
- ג. m מאותחל עם טעינת המחלקה X והוא נגיש רק ממתודות מופע.
- ד. m מאותחל עם טעינת המחלקה X, והוא נגיש ממתודות מופע ומתודות סטטיות שמכילות משתנה מטיפוס סטטי X.

נימוק:

שאלה 8:

```
public class Q8 {
    public static void main(String[] args) {
        Stream<Integer> s= Stream.generate(new SmallNumbers());
        System.out.println(s.peek(x->System.out.print("x"))
            .map(x->x/2)
            .peek(x->System.out.print("y"))
            .filter(x->x%2== 0)
            .allMatch(x-> x < 2));
    }
}

public class SmallNumbers implements Supplier<Integer>{
    int currNum;
    public Integer get() {
        currNum++;
        if (currNum >= 5) {currNum = 0;}
        return currNum;
    }
}
```

מה יקרה בהרצת הקוד הבא?

בחר\י בתשובה הטובה ביותר:

- א. הקוד ידפיס את הרצף xy בלולאה אינסופית.
- ב. הקוד ידפיס את הרצף xxy בלולאה אינסופית.
- ג. הקוד ידפיס את הרצף xyx בלולאה אינסופית.
- ד. הקוד ידפיס xxxxyyyyyyytrue ויעצור (x מופיע 5 פעמים ו y מופיע 5 פעמים).
- ה. הקוד ידפיס falsexyxxyxxy ואז יעצור (הרצף xy מופיע 4 פעמים).
- ו. הקוד ידפיס falsexyxxy ואז יעצור (הרצף xxy מופיע פעמיים).
- ז. הקוד ירוץ בלולאה אינסופית ולא ידפיס כלום.
- ח. כל התשובות מלבד תשובה זו לא נכונות.

נימוק:

שאלה 9:

בחר\י בתשובה הטובה ביותר:

- א. בנאי ברירת מחדל (בנאי דיפולטי) קיים רק אם לא הוגדר בנאי אחר במחלקה.
- ב. במחלקה X כלשהי: אם השורה הראשונה של בנאי אינה קריאה ל super או ל this, הקומפיילר "מוסיף" קריאה לבנאי הדיפולטי של המחלקה ממנה X יורשת.
- ג. ניתן להגדיר כמה בנאים למחלקה, אך הם חייבים לקרוא אחד לשני.
- ד. במחלקה X : אם כל הבנאים של מחלקה מסויימת הם private, לא קיימים אובייקטים מטיפוס זמן ריצה (דינאמי) X במערכת.
- ה. תשובות א-ד לא נכונות.
- ו. מבין תשובות א-ד יש 2 תשובות נכונות.**
- ז. מבין תשובות א-ד יש 3 תשובות נכונות.
- ח. תשובות א-ד נכונות.

נימוק:

שאלה 10:

נרצה לממש את המחלקה MapToString אשר מממשת את הממשק Map ומתארת מיפויים מ String לעצמים מטיפוס גנריים כלשהם.

אופן השימוש הוא כזה:

```
public static void main(String[] args) {
    Map<String, Integer> m1 = new StringMap<Integer>();
    m1.put("abc", 3);
    int i = m1.get("abc");
    StringMap<String> m2 = new StringMap<String>();
    m2.put("abc", "abc");
    String s = m2.get("abc");
}
```

כיצד נגדיר את StringMap כך שהקוד ב main יתקמפל? בחר\י בתשובה הטובה ביותר:

- א. `public class StringMap<String,T> implements Map<String, T>`
- ב. `public class StringMap<String> implements Map<String, T>`
- ג. `public class StringMap<T> implements Map<String, T>`**
- ד. `public class StringMap<V> implements Map<T extends String, V>`
- ה. `public class StringMap<T,V> implements Map<V extends String, V>`
- ו. יש יותר מתשובה נכונה אחת.

נימוק:


```

/* code block 1 */
try {
    /* code block 2 */
}
catch (IOException exp){
    /* code clock 3 */
}
finally {
    /* code clock 4 */
}
    
```

בחר/י בתשובה הטובה ביותר:

- א. אם ב block 1 נזרק חריג, הקוד ב finally לא יתבצע.
- ב. אם ב block 2 נזרק חריג שלא נתפס ע"י ה catch, הקוד של finally לא יתבצע.
- ג. אם ב block 3 נזרק חריג, הקוד של finally לא יתבצע.
- ד. מלבד תשובה זו יש יותר מתשובה נכונה אחת.
- ה. מלבד תשובה זו כל התשובות לא נכונות.

נימוק:

שאלה 12:

```

public interface I1{
    public void func(int i);
}

public interface I2{
    public void func(int i);
}
    
```

לפניכם מספר טענות על מנשקים:

- טענה 1: מתודה המוגדרת כ default היא מתודת מופע שקיים לה מימוש במנשק.
- טענה 2: כל השדות במנשק יהיו שדות בניראות public, ובנוסף הם יהיו final ו static.
- טענה 3: ניתן להגדיר מנשק אשר מרחיב (extends) גם את I1 וגם את I2.

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 13:

```
List<Integer> l1 = ???;
List<? extends Number> l2 = ???;
List<Number> l3 = ???;
List<? super Number> l4 = ???;
List<Object> l5 = ???;
l2 = l1; // *
l3 = l2; // #
l4 = l5; // %
```

הניחו כי במקום ??? מופיעות השמות חוקיות לכל אחד מ 5 המשתנים שמוגדרים בקוד, וש 5 השורות הראשונות שבהן מוגדרים 1-5 מתקמפלות. לפניכם 3 טענות על הקוד המצורף:

- טענה 1: השורה המסומנת ב * מתקמפלת.
- טענה 2: השורה המסומנת ב # מתקמפלת.
- טענה 3: השורה המסומנת ב % מתקמפלת.

בחרו 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:

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.
Set<Map.Entry<K,V>>	entrySet() Returns a Set view of the mappings contained in this map.
V	get(Object key) Returns the value to which the specified key is mapped, or null 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 defaultValue if this map contains no mapping for the key.
Set<K>	keySet() Returns a Set 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.
int	size() Returns the number of key-value mappings in this map.
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.
void	clear() Removes all of the elements from this set (optional operation).
boolean	contains(Object o) Returns true if this set contains the specified element.
Iterator<E>	iterator() Returns an iterator over the elements in this set.
boolean	remove(Object o) Removes the specified element from this set if it is present.
int	size() Returns the number of elements in this set (its cardinality).

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.
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.

public final class String

char	charAt(int index) Returns the char value at the specified index.
int	compareTo(String anotherString) Compares two strings lexicographically.
int	length() Returns the length of this string.
String[]	split(String regex) Splits this string around matches of the given regular expression.
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.
-------------	--

public class BufferedReader

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

public class FileReader (extends Reader)

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