

בחינה בתוכנה 1

סמסטר ב תשפ"ב, מועד ב', 4 בספטמבר 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'):

בשאלה זו עליכם לממש מחלקות המתארות פולינומים. הפולינומים בהם נעסוק יהיו פולינומים עם מקדמים (coefficients) שלמים. החזקות (exponents) יהיו אי שליליות. כל איבר בפולינום נקרא מונום, כאשר לכל מונום יש מקדם וחזקה. פולינום הוא סכום של מונום אחד או יותר, והוא מכיל לפחות מונום אחד שהמקדם שלו שונה מ 0. אין צורך להתייחס לפולינום האפס בתרגיל זה.

דוגמא לפולינומים אשר עומדים בתנאים שהגדרנו:

$$p1 = x^2$$

$$p2 = 3 + 8x + 5x^5$$

$$p3 = 5$$

הפולינום  $p1$  הוא פולינום בעל מונום יחיד, עם דרגה 2. הפולינום  $p2$  הוא פולינום בעל שלושה מונומים, ודרגתו היא 5. הפולינום השלישי בעל מונום יחיד, ודרגתו היא 0.

את הפולינומים תממשו בשתי מחלקות שונות:

`ListPolynomial` המייצגת פולינום באמצעות רשימה של אובייקטים המייצגים מונומים עם מקדם שונה מ 0 (כלומר, בייצוג זה יישמרו רק מונומים שהמקדם שלהם שונה מ 0).

`ArrayPolynomial` המייצגת פולינום ע"י מערך בגודל קבוע. המערך מכיל את מקדמי הפולינום כשהתא  $i$  מייצג את המקדם של החזקה  $i$ .

תחילה, נגדיר את מחלקת העזר `Monom` המתארת מונום יחיד עם מקדם שונה מ 0.

```
public class Monom {

    private int coeff;
    private int exp;

    /* @pre: exp >= 0, coeff != 0*/
    public Monom(int coeff, int exp){
        this.coeff = coeff;
        this.exp = exp;
    }

    /* @post: $ret != 0*/
    public int getCoeff() { return coeff;}

    /* @post: $ret >= 0*/
    public int getExp() { return exp;}

}
```

נמשיך עם הגדרת הממשק `IPolynomial` המתאר פולינום.

```
public interface IPolynomial extends Iterable<Monom> {

    public final static int MAX_DEGREE = 128;

    public int getDegree();

    public float eval(float x) ;

    /* pre: 0 <= exp <= MAX_DEGREE */
    public void updateCoeff(int coeff, int exp);

    /* pre: 0 < k <= getDegree() */
    public IPolynomial getKthDerivation(int k) ;

}
```

הקבוע MAX\_DEGREE קובע את הדרגה המקסימלית של פולינום המוגדר ע"י IPolynomial. השירות getDegree מחזיר את דרגת הפולינום, כלומר – החזקה הכי גדולה שהמקדם שלה שונה מ 0.

השירות updateCoeff מעדכן את המקדם של המונום עם החזקה exp לערך coeff. ניתן להניח שאחרי פעולת updateCoeff לא ייווצר פולינום האפס (אך כן יתכן שחזקה כלשהי תתאפס).

השירות eval מחזיר את ערכו של הפולינום עבור הערך x.

השירות getKthDerivation מחזיר את פולינום הנגזרת ה k של הפולינום המקורי.

המנשק IPolynomial מרחיב את Iterable<Monom>. האיטרטור יחזיר רק את המונומים שהמקדמים שלהם שונים מ 0 בסדר עולה של חזקות.

להלן דוגמת שימוש באובייקט מסוג IPolynomial:

```
List<Monom> monoms = new ArrayList<>();
monoms.add(new Monom(1,0)); //1
monoms.add(new Monom(2,1)); //2x
monoms.add(new Monom(3,3)); //3x^3
IPolynomial poly = new IPolynomial(monoms); //poly= 1+2x+3x^3
for (Monom m : poly) {
    System.out.print(m.getExp() + " "); //0 1 3
}
System.out.println(poly.eval(2)); //29.0
System.out.println(poly.getDegree()); //3
IPolynomial der1 = poly.getKthDerivation(1); //der1 = 2+9x^2
IPolynomial der2 = poly.getKthDerivation(2); //der2 = 18x
poly.updateCoeff(5,2); //poly = 1+2x+5x^2+3x^3
```

הפולינום שהמשתנה poly מצביע אליו הוא  $1 + 2x + 3x^3$ . פולינום זה נוצר ע"י קריאה לאחד מהבנאים של שתי המחלקות שמממשות את המנשק IPolynomial (שני הבנאים מקבלים רשימה של Monom-ים).

הלולאה על הפולינום תרוץ 3 פעמים וידפסו בסדר עולה שלושת החזקות של מונומים שהמקדמים שלהם שונים מ 0.

החישוב עבור eval בנקודה  $x=2$  הוא  $1+2*2+3*8=29$

כאשר מפעילים את השירות updateCoeff (שורה אחרונה), מעדכנים את המקדם של החזקה 2 להיות 5 (כרגע המקדם הוא 0). אם נרוץ בלולאה על poly לאחר עדכון זה, הלולאה תרוץ 4 פעמים כיוון שיש 4 מונומים עם מקדם שונה מ 0.

הנחיה כללית – הקפידו על חוסר תלות בין המחלקות – המחלקות הקונקרטיות (שני המימושים של פולינום) לא מכירות האחת את השניה, וגם המחלקה האבסטרקטית לא מכירה אף אחת מהן.

בשאלה זו יש 4 סעיפים, ובהם תממשו את שתי המחלקות במקביל. בסעיף א' תשלימו את מימוש הבנאים. בסעיף ב' תשלימו את המימוש של getCoeff עבור שתי המחלקות. בסעיף ג' תשלימו את המימוש של eval, ובסעיף האחרון את המימוש של getKthDerivation.

סעיף א' (7 נק'): \_\_\_\_\_

נתחיל מתיאור כללי של שתי המחלקות. שתי המחלקות יורשות מהמחלקה האבסטרקטית AbstractPolynomial אשר כרגע מכילה רק את השדה degree ומימוש לפונקציה getDegree, וניתן להוסיף אליה קוד משותף במידת הצורך.

```
public abstract class AbstractPolynomial implements IPolynomial{
    protected int degree;

    public int getDegree() { return this.degree; }
}
```

**המחלקה ListPolynomial** מממשת פולינום באמצעות רשימה של אובייקטים מטיפוס Monom עבור מונומים שהמקדמים שלהם שונים מ 0. להלן קוד חלקי של המחלקה (את השאר תשלימו בהמשך השאלה):

```
public class ListPolynomial extends AbstractPolynomial {
    private List<Monom> monomsList;

    public Iterator<Monom> iterator() {
        return monomsList.iterator();
    }
}
```

**המחלקה ArrayPolynomial** מממשת פולינום באמצעות מערך של מקדמים. מכיוון שהדרגה המקסימלית של פולינום ידועה מראש (מוגדרת ע"י הקבוע MAX\_DEGREE), ניתן לייצג את כל הפולינומים ע"י מערכים בגודל הקבוע (כלומר, לא תלוי בדרגת הפולינום הספציפי), כאשר בתא ה i ישמר המקדם שמתאים לחזקה ה i.

```
List<Monom> monoms = new ArrayList<>();
monoms.add(new Monom(1,0)); //1
monoms.add(new Monom(2,1)); //2x
monoms.add(new Monom(3,3)); //3x^3
IPolynomial poly = new ArrayPolynomial(monoms);
```

הפולינום poly שנוצר בדוגמא זו מיוצג ע"י מערך אשר 4 התאים הראשונים שלו ניראים כך:

אינדקס	0	1	2	3
ערך	1	2	0	3

התא באינדקס 2 מכיל את הערך 0, כיוון שהמקדם של החזקה 2 בפולינום הוא 0. שאר התאים במערך יכילו גם כן את הערך 0.

להלן הקוד החלקי של המחלקה:

```
public class ArrayPolynomial extends AbstractPolynomial{
    private int[] coeffsArr;

    public Iterator<Monom> iterator() {
        /* implementation is provided inside */
    }
}
```

שימו לב – מימושי האיטרטורים של שתי המחלקות נתונים לכם (גם אם לא כל מוצג), כך שאתם לא צריכים לממש אותם ויכולים להשתמש בהם במידת הצורך.

השלימו את מימוש הבנאים של שתי המחלקות לפי החוזה שמופיע בתחילת העמוד הבא. החוזה רלוונטי לשני הבנאים.

הוסיפו פונקציות עזר ו\או קוד משותף במחלקה האבסטרקטית לפי הצורך. אין להוסיף שדות חדשים. בנוסף, אין להוסיף בנאים חדשים למחלקות ArrayPolynomial ו ListPolynomial.

```

/* applies to both ListPolynomial and to ArrayPolynomial:
@pre: monoms.size() > 0
@pre: for each i, 0 <= monoms[i].getExp() <= MAX_DEGREE
      for each i, monom[i].getExp() < monom[i+1].getExp() */

```

```

public class ListPolynomial extends AbstractPolynomial {
    private List<Monom> monomsList;

```

```

    public ListPolynomial(List<Monom> monoms) {

```

```

        super(monoms);
        monomsList = new ArrayList<>();
        monomsList.addAll(monoms);

```

```

    }}

```

```

public class ArrayPolynomial extends AbstractPolynomial{
    private int[] coeffsArr;

```

```

    public ArrayPolynomial(List<Monom> monoms) {

```

```

        super(monoms);
        this.coeffsArr = new int[MAX_DEGREE+1];
        for (Monom m: monoms) {
            this.coeffsArr[m.getExp()] = m.getCoeff();
        }

```

```

    }}

```

```

public abstract class AbstractPolynomial implements IPolynomial{
    protected int degree

```

```

    public AbstractPolynomial(List<Monom> monoms) {
        degree = monoms.get(monoms.size() - 1).getExp();
    }

```

### סעיף ב' (12 נק').

השלימו את המימוש של השירות `updateCoeff` עבור שתי המחלקות. תזכורת – השירות מקבל שני שלמים המייצגים מקדם (`coeff`) וחזקה (`exp`), וקובע את המקדם של `exp` בפולינום להיות `coeff`. הקפידו להתייחס לכל הקלטים שהחוצה מאפשר, אך ניתן להניח שאחרי `updateCoeff` לא מתקבל פולינום שכל מקדמיו הם 0 (פולינום ה-0).

במסגרת המיועדת לסעיף זה מופיעה הגדרת השירות מתוך המנשק. במסגרת לא מופיעה חלוקה למחלקות, כך שעליכם לציין עבור כל פונקציה את המחלקה שבה היא ממומשת.

הוסיפו פונקציות עזר ולא קוד משותף במחלקה האבסטרקטית לפי הצורך. אין להוסיף שדות חדשים. בנוסף, אין להוסיף בנאים חדשים למחלקות `ArrayPolynomial` ו `ListPolynomial`.

```
/* pre: 0 <= exp <= MAX_DEGREE */
public void updateCoeff(int coeff, int exp); */

// ArrayPolynomial
public void updateCoeff(int coeff, int exp) {
    this.coeffsArr[exp] = coeff;
    for (int i = this.coeffsArr.length-1; i >=0 ; i--) {
        if (this.coeffsArr[exp] > 0) {
            this.degree = i;
            return;
        }
    }
}

// ListPolynomial
public void updateCoeff(int coeff, int exp) {
    if (exp > this.getDegree() && coeff != 0){
        this.monomsList.add(new Monom(coeff, exp));
        this.degree = exp;
        return;
    }
    for (int i =0; i < this.monomsList.size(); i++) {
        if (this.monomsList.get(i).getExp() >= exp) {
            if (this.monomsList.get(i).getExp() == exp) {
                this.monomsList.remove(i);
            }
            if (coeff != 0) {
                Monom newMonom = new Monom(coeff, exp);
                this.monomsList.add(i, newMonom);
            }
            break;
        }
    }
    /* the degree should be updated because we might have updated the
    coeff of the highest degree to 0. */
    this.degree = this.monomsList.get(
        this.monomsList.size() - 1).getExp();
}
```

סעיף ג' (6 נק')::

השלימו את מימוש השירות eval עבור שתי המחלקות:

במסגרת המיועדת לסעיף זה מופיעה הגדרת השירות מתוך המנשק. במסגרת לא מופיעה חלוקה למחלקות, כך שעליכם לציין עבור כל פונקציה את המחלקה שבה היא ממומשת.

הוסיפו פונקציות עזר ולא קוד משותף במחלקה האבסטרקטית לפי הצורך. אין להוסיף שדות חדשים. בנוסף, אין להוסיף בנאים חדשים למחלקות ArrayPolynomial ו ListPolynomial.

```

/* public float eval(float x) ; */

//AbstractPolynomial

public float eval(float x) {
    float res = 0;
    for(Monom m :this) {
        res += m.getCoeff() * Math.pow(x, m.getExp());
    }
    return res;
}

```

סעיף ד' (12 נק')::

השלימו את מימוש הפונקציה getKthDerivation עבור שתי המחלקות. תזכורת, הנגזרת ה k של הפולינום p הוא הפולינום שמתקבל אחרי שגוזרים k פעמים את p. הפונק' תחזיר פולינום חדש מטיפוס זהה לטיפוס של הפולינום שנגזר. הקפידו על כך שתהיה חוסר תלות בין שתי המחלקות: בעת גזירת ListPolynomial אין לייצר אובייקט מסוג ArrayPolynomial, ולהיפך.

במסגרת המיועדת לסעיף זה מופיעה הגדרת השירות מתוך המנשק. במסגרת לא מופיעה חלוקה למחלקות, כך שעליכם לציין עבור כל פונקציה את המחלקה שבה היא ממומשת.

**הנחיה** – המימוש שלכם צריך להקפיד על שיתוף קוד מירבי. בפרט, הלוגיקה המופעלת בעת ביצוע פעולת הגזירה צריכה להיות ממומשת במשותף לשתי המחלקות.

הוסיפו פונקציות עזר ולא קוד משותף במחלקה האבסטרקטית לפי הצורך. אין להוסיף שדות חדשים. בנוסף, אין להוסיף בנאים חדשים למחלקות ArrayPolynomial ו ListPolynomial.

```

/* pre: 0 < k <= getDegree()
public IPolynomial getKthDerivation(int k) ; */

// AbstractPolynomial

/* @pre: m.getExp() >= k */
public Monom deriveSingleMonom(Monom m, int k) {
    int coeff = m.getCoeff();
    while (k > 1) {
        coeff *= k;
        k--;
    }
    return new Monom(coeff, m.getExp() -k);
}

public List<Monom> getDerivedMonoms(int k) {
    List<Monom> res = new ArrayList<>();
    for (Monom m : this) {
        if (m.getExp() >= k){
            res.add(deriveSingleMonom(m, k));
        }
    }
    return res;
}

// ArrayPolynomial

public IPolynomial getKthDerivation(int k) {
    return new ArrayPolynomial(getDerivedMonoms(k));
}

// ListPolynomial

public IPolynomial getKthDerivation(int k) {
    return new ListPolynomial(getDerivedMonoms(k));
}

```



## שאלה 2 (18 נק')

סעיף א' (9 נק'): \_\_\_\_\_

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

[`"x", "x", "x", "x", "y", "y", "x"`]  $\Rightarrow$  [`<"x", 4>`, `<"y", 2>`, `<"x", "1">`]

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

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

```
RepeatedElement<String> e = new RepeatedElement<>("x", 4);
System.out.println(e.getElement()); //x
System.out.println(e.getRepetitionsNum()); //4
```

נרצה לממש איטרטור בשם `CompIterator` אשר מקבל איטרטור ומחזיר איברים מכווצים.

```
List<String> lst = Arrays.asList("x", "x", "x", "x", "y", "y", "x");
Iterator<?> iter = new CompIterator<>(lst.iterator());
while(iter.hasNext()) {
    System.out.println(iter.next());
}
```

בדוגמת קוד זו, האיטרטור ירוץ שלוש פעמים ויחזיר שלושה איברים מסוג `RepeatedElement`: הראשון יאותחל ע"י ("x", 4), השני יאותחל ע"י ("y", 2) והשלישי יאותחל ע"י ("x", 1).

**שימו לב:** בשום שלב אין לשחזר בזכרון את כל האוסף שאיטרטור הקלט מגדיר. בפרט, אסור לייצר את הרשימה המקורית בזכרון, לבנות רשימה של `RepeatedElement`-ים, ואז להחזיר אותם אחד אחד. עליכם להתקדם באיטרטור המקורי רק לפי הצורך.

ניתן להניח שאיטרטור הקלט שמקבל הבנאי לא יחזיר איבר שהוא `null`.

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

```
public class CompIterator < T > implements
    Iterator <Iterator<RepeatedElement<T> > > {
```

```
    private Iterator<T> iter;
    private T toReturn = null;
```

```
    public CompactIterator(Iterator<T> iter) {
        this.iter = iter;
        if (iter.hasNext()) {
            toReturn = iter.next();
        }
    }
```

```
    public boolean hasNext() {
        return toReturn != null;
    }
```

```

public RepeatedElement<T> next() {
    int cnt = 1;
    T curr = null;
    while (iter.hasNext()) {
        curr = iter.next();
        if (curr == toReturn) {
            cnt++;
        }
        else {
            break;
        }
    }
    RepeatedElement<T> retValue = new
        RepeatedElement<>(toReturn, cnt);
    toReturn = curr;
    return retValue;
}
}

```

סעיף ב' (9 נק):

ממשו את הפונקציה analyze אשר מקבלת רשימה של רשימות, כל רשימה פנימית מייצגת משפט, והיא מחזירה את ה k המקסימלי כך שקיימות לפחות k מילים שונות המופיעות בלפחות k משפטים שונים. אם אף מילה לא מופיעה ביותר ממשפט אחד הפונקציה תחזיר 0.

לדוגמא: עבור הקלט:

```

List<List<String>> sentences = new ArrayList<>();
sentences.add(Arrays.asList("I", "love", "java"));
sentences.add(Arrays.asList("python", "is", "fun"));
sentences.add(Arrays.asList("java", "is", "awesome"));
sentences.add(Arrays.asList("java", "is", "love", "is", "java"));
System.out.println(analyze(sentences)); //2

```

הפונקציה תחזיר את הערך 2. יש לפחות 2 מילים שמופיעות ביותר מ 2 משפטים (למשל, love, java). זהו ה k המקסימלי כיוון שיש רק 2 מילים שמופיעות בלפחות 3 משפטים (is, java).

```

/* @ret >= 0 */
public static int analyze (List<List<String>> sentences) {
    Map<String, Integer> wordsCnt = new HashMap<>();
    for (List<String> sentence : sentences) {
        Set<String> uniques = new HashSet<>();
        uniques.addAll(sentence);
        for (String u : uniques) {
            wordsCnt.put(u, wordsCnt.getOrDefault(u,0)+1);
        }
    }
    Map<Integer, Integer> counts = new HashMap<>();
    for (int i : wordsCnt.values()) {
        counts.put(i, counts.getOrDefault(i,0)+1);
    }
}

```

```

    /*counts hold the mapping between x and the number of words that
    appeared in x sentences*/

    int k = Collections.max(counts.keySet());
    int numOfWordsThatAppearInKOrMoreSentences = 0;
    while (k > 0) {
        numOfWordsThatAppearInKOrMoreSentences +=
            counts.getOrDefault(k,0);
        if (numOfWordsThatAppearInKOrMoreSentences >= k) {
            return k;
        }
        k--;
    }
    return 0;
}
}

```

מסגרת החירום מופיעה בעמוד הבא.

שאלה 3 (5 נק' ):

לפניכם קוד המבצע פעולת casting ושלוש טענות שמתייחסות אליו:

```

A a = /***/
B b = (B)a;

```

- טענה 1: אם B היא מחלקה אבסטרקטית, הקוד יתקמפל אך תמיד יזרוק שגיאה בזמן ריצה.
- טענה 2: אם B הוא הטיפוס הפרימיטיבי short ו A הוא הטיפוס הפרימיטיבי int, יתכן ש a יכיל ערך מספרי שונה מ b (כלומר, לא רק ייצוג שונה בזכרון אלא ממש ערך שונה).
- טענה 3: אם A ו B הם טיפוסים הפניה ופעולת ה casting מתבצעת ללא שגיאות זמן ריצה, מתקיים a==b.

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: \_\_\_\_\_

מסגרת חירום

שאלה 4 (5 נק'): \_\_\_\_\_

```
public class Base{
    private String privStr;
    protected Object proObj;
    public final int finInt = 9;

    /*public class Inner{
        public void func() {
            System.out.println(privStr);
        }
    } */
}
```

טענה 1: לא ניתן להגדיר שדה בשם finInt במחלקה שירשת מ Base.

טענה 2: אם נוציא את המחלקה Inner מהערה תהיה שגיאת קומפילציה בפונקציה func.

טענה 3: השירות foo שמוגדר כך:

```
public void foo(Base b) { System.out.println(b.proObj); }
```

יתקמפל אם הוא מוגדר במחלקה אשר לא יורשת מ Base אך נמצאת באותה החבילה בה Base

נמצאת.

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 5: \_\_\_\_\_

נניח את קיומה של מחלקה אבסטרקטית AbsClass. בחר/י בתשובה הטובה ביותר:

- א. יתכן שקיים אובייקט כך שטיפוס זמן הריצה שלו (הטיפוס הדינמי) הוא AbsClass.
- ב. מחלקה אבסטרקטית לא יכולה לרשת ממחלקה אבסטרקטית אחרת.
- ג. מנשק יכול לרשת ממחלקה אבסטרקטית.
- ד. מחלקה אבסטרקטית יכולה להכיל שירותים בניראות private ו protected.
- ה. ניתן להגדיר שירות abstract גם במחלקה רגילה.
- ו. מלבד תשובה זו כל התשובות לא נכונות.
- ז. מלבד תשובה זו יש יותר מתשובה נכונה אחת.

נימוק:

## שאלה 6:

לפניכם מימוש המחלקה MyClass.

```

public final class MyClass{
    private final int[] arr;
    private final String str;
    private final int i;

    public MyClass(int[] arr) {
        this.arr = arr;
        this.str = "abc";
        this.i = arr.length;
    }

    public String getStr() {
        return str;
    }

    public int func(int j) {
        int res = this.i+j;
        return res;
    }
}

```

המתכנת אביב מעוניינת ש MyClass יהיה immutable. איזה מהצעדים הבאים נדרש על מנת להבטיח זאת?

שינוי 1: צריך לשנות את מימוש הבנאי.

שינוי 2: צריך לשנות את מימוש הפונקציה getStr.

שינוי 3: צריך לשנות את מימוש הפונקציה func.

בחר/י בתשובה הטובה ביותר:

- א. המחלקה היא כבר immutable ולכן לא נדרש שום שינוי.
- ב. מספיק לבצע את שינוי 1 על מנת שהמחלקה תהיה immutable.
- ג. מספיק לבצע את שינוי 2 על מנת שהמחלקה תהיה immutable.
- ד. מספיק לבצע את שינוי 3 על מנת שהמחלקה תהיה immutable.
- ה. מספיק לבצע את שינויים 1+2 על מנת שהמחלקה תהיה immutable.
- ו. מספיק לבצע את שינויים 1+3 על מנת שהמחלקה תהיה immutable.
- ז. מספיק לבצע את שינויים 2+3 על מנת שהמחלקה תהיה immutable.
- ח. צריך לבצע את שלושת השינויים על מנת שהמחלקה תהיה immutable.

נימוק:

## שאלה 7:

לפניכם קוד של שתי מחלקות שלכל אחת יש חוזה משלה.

```

/* @inv: func() > 0 */
public class A{

    public int func() { /* code here */ }

    /* @pre: (x > 0) && (x % 4 == 0)
     * @post: $ret % 2 == 0 */
    public int foo(int x) { /* code here */ }
}

/* @inv: func() % 2 == 0 */
public class B /* extends A */ {

    public int func() { /* code here */ }

    /* @pre: x % 2 == 0
     * @post: ($ret > 0) && ($ret % 2 == 0) */
    public int foo(int x) { /* code here */ }
}

```

נרצה שמחלקה B תירש ממחלקה A (כלומר, להוציא מההערה את הקוד בשורה שבה B מוגדרת). בהנחה שהחוזה של A לא ישתנה, אילו סעיפים בחוזה B יש לשנות על מנת לקיים את חוקי הירושה של חוזים? הניחו כי בכל מקרה שבו חוזה מתעדכן, גם המימוש יתעדכן.

בחרו בתשובה הטובה ביותר:

- א. יש לשכתב רק את האינוריאנטה של B.
- ב. יש לשכתב רק את תנאי ה pre של foo ב B.
- ג. יש לשכתב רק את תנאי ה post של foo ב B.
- ד. יש לשכתב גם את האינוריאנטה של B וגם את תנאי ה pre של foo ב B.
- ה. יש לשכתב גם את האינוריאנטה של B וגם את תנאי ה post של foo ב B.
- ו. יש לשכתב גם את תנאי ה pre וגם את תנאי ה post של foo ב B.
- ז. החוזה של B כמו שהוא מאפשר ירושה בין B ל A ללא שינויים.
- ח. יש לשכתב את כל סעיפי החוזה שמופיעים ב B.

נימוק:

שאלה 8:

```
public class Outer<T>{
    public Outer() {}
    public class Inner{
        public Inner() { /* some code here */ }
    }
    public static class InnerStatic{ /* some code here */ }
}
```

לפניכם מספר טענות המתייחסות לקוד הנתון. בחר/י בתשובה הטובה ביותר:

טענה 1: הקוד בתוך Inner יכול לעשות שימוש ב T.

טענה 2: הקוד בתוך InnerStatic יכול לעשות שימוש ב T.

טענה 3: ניתן לקרוא לבנאי של Inner רק בתוך קוד המופיע במחלקה Outer או במחלקות הפנימיות שלה.

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 9:

```
public class Q9 {
    public <T> T f1(Collection<T> c, List<? extends T> l ) {
        return l.get(0);
    }
    public void f2(List<? extends Number> l1, List<Integer> l2) {
        l1 = l2;
    }
    public void f3(List<? extends Number> l1,
                  List<? super Number> l2) {
        l2.add(l1.get(0));
    }
}
```

אילו מהפונקציות הנתונות מתקמפלת? בחר/י בתשובה הטובה ביותר:



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

נימוק:

שאלה 10:

```
public class Base{
    public int i = 3;

    public int func(Object s) {return i; }

    public int foo(String s) {return this.func(s) + this.goo(s);}

    public int goo(String s) {return 1;}
}

public class Sub extends Base{
    public int i = 1;

    public int func(Object o) { return 2*i; }

    public int func(String o) { return 4*i;}

    public int goo(String s) {return func(s);}

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

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

- א. 6
- ב. 7
- ג. 8
- ד. 9
- ה. 10
- ו. 11
- ז. 12
- ח. 13

נימוק:

שאלה 11:

לפניכם פסאודו קוד המתאר מבנה של זרם עם 3 פעולות ביניים ופעולה סופנית. הפעולות intermediate1-3 הן פעולות ביניים כלשהן, ובכל התייחסות אליהן מתייחסים באופן כללי לפעולות ביניים.

```
Stream<Integer> s = ... ;
s.intermediate1(...).intermediate2(...).intermediate3(...).allMatch(...);
```

בחרו בתשובה הטובה ביותר:

- א. אם `allMatch` מופעלת על זרם אינסופי, היא תחזיר `False` או שתמשיך לרוץ לנצח.
- ב. על מנת ש `intermediate2` תעביר איבר יחיד ל `intermediate3`, היא צריכה לצרוך רק איבר אחד מ `intermediate1`.
- ג. הפעולה `intermediate3` מופעלת בהכרח על זרם של `Integer`-ים.
- ד. מלבד תשובה זו כל התשובות לא נכונות.
- ה. שתיים מבין התשובות א,ב,ג הן נכונות.
- ו. תשובות א,ב,ג נכונות.

נימוק:

שאלה 12:

```
try {}
catch (IOException exp) {}
catch (Exception exp) {} /**
finally {}
```

טענה 1: אם תיזרק שגיאת `NullPointerException` בתוך בלוק ה `try`, התוכנית תבצע את הבלוק של ה `catch` המופיע בשורה שמומנת ב `**`.

טענה 2: בלוק `finally` יתבצע גם אם הקוד נכנס לאחד מבלוקי ה `catch` ובתוכו נזרק חריג.

טענה 3: אם תיזרק שגיאת `IOException` בבלוק ה `try`, התוכנית תבצע את שני בלוקי ה `catch`.

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	<code>containsKey(Object key)</code> Returns true if this map contains a mapping for the specified key.
<code>Set&lt;Map.Entry&lt;K,V&gt;&gt;</code>	<code>entrySet()</code> Returns a Set 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 null 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 <code>defaultValue</code> 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&lt;K&gt;</code>	<code>keySet()</code> Returns a Set 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 <code>key</code> , or null if there was no mapping for <code>key</code> .
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 <code>key</code> , or null if there was no mapping for <code>key</code> .
int	<code>size()</code> Returns the number of key-value mappings in this map.
<code>Collection&lt;V&gt;</code>	<code>values()</code> Returns a Collection 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.
boolean	<code>addAll(Collection&lt;? Extends E&gt; c)</code> 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.
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&lt;E&gt;</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 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.
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
E	remove(int index) Removes the element at the specified position in this list. Returns the element previously at the specified position
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 char value at the specified index.
static String	join(CharSequence delimiter, CharSequence... elements) Returns a new String composed of copies of the CharSequence elements joined together with a copy of the specified delimiter.
int	length() Returns the length of this string.
String[]	split(String regex) 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.

**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.
-------------	--