

## בחינה בתוכנה 1

סמסטר ב' תש"ף, מועד א', 29 ביולי 2020  
לנה דנקין, שי גרשטיין, איתי יצחק

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 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.

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.

הבחינה תיערך בפורמט מקוון. אלה הדגשים למענה על הבחינה המקוונת:

- את התשובות הנכונות יש לסמן בגוף הבחינה במודל.
- בנוסף לסימון הבחינה יש לנמק בתמצות את כל התשובות על דפים נקיים שיוכנו מבעוד מועד.
- יש לנמק את השאלות לפי הסדר, ולסמן בבירור את מספר השאלה אליה מתייחס כל נימוק.

את דפי הנימוקים יש לסרוק בסיום הבחינה ולהעלות למודל. שם הקובץ יהיה answers. נימוק חסר או לא נכון עלול לגרום לאי קבלת נקודות על שאלה במקרים שבהם יוחלט לקבל יותר מתשובה אחת. נימוק לתשובה אחרת מזו שסומנה בטופס הבחינה לא יתקבל.

- בנוסף לדפי נימוקים, אתם יכולים להשתמש בדפי טיוטא שיהיו גם כן דפים לבנים חלקים שיוכנו מבעוד מועד. גם את הטיוטות עליכם לסרוק ולהגיש. שם הקובץ הסרוק יהיה draft.
- במהלך הבחינה לא נדרשת שום הקלדה על המחשב. כל מה שעליכם לעשות בזמן הבחינה הוא לסמן את התשובה הנכונה בכל שאלה ולדפדף בין דפי הבחינה.

בבחינה אסור השימוש בחומר עזר כלשהו, כולל מחשבוני, מחשבים או כל מכשיר אחר פרט לדפים נקיים ולמכשירי כתיבה.

© כל הזכויות שמורות למחברים. מבלי לפגוע באמור לעיל, אין להעתיק, לצלם, להקליט, לשדר, לאחסן במאגר מידע, בכל דרך שהיא, בין מכונית ובין אלקטרונית או בכל דרך אחרת כל חלק שהוא מטופס הבחינה.

בהצלחה!

שאלה 1:

שאלות 1-4 מתייחסות לקוד הבא:

המחלקה Fib מחשבת איברים בסדרת פיבונאצ'י בצורה יעילה, ע"י שמירת כל החישובים שכבר  
בוצעו. שימו לב, קוד זה דומה, אך לא זהה, לקוד שראיתם במהלך הקורס.

בשאלות 2-4 נעשים שינויים במחלקה. כל שינוי שנעשה תקף רק לאותה השאלה. כלומר, כל שאלה  
מתחילה בקוד שנתון בשאלה 1.

נזכיר שסדרת פיבונאצ'י מורכבת מהאיברים הבאים: 0 1 1 2 3 5 8 13

```

/*
 * @imp_inv: 1 <= lastCalculated < MAX_VAL
 * @imp_inv: for i in {0,1}, fibArr[i] = i
 * @imp_inv: for i>lastCalculated, fibArr[i] = 0
 * @imp_inv: for 1<i<=lastCalculated,
 *             fibArr[i] = fibArr[i-1] + fibArr[i-2]
 */
public class Fib{
    public final static int MAX_VAL = 50;
    private int lastCalculated;
    private long[] fibArr = new long[MAX_VAL];

    public Fib() {
        lastCalculated = 1;
        fibArr[1] = 1; // fibArr[0] = 0 and fibArr[1] = 1
    }

    /**
     * @pre n >= 1, n < MAX_VAL
     * @post lastCalculated = n
     */

    public Fib(int n) {
        this();
        if (n>=2) { calcFibForI(n); }
    }

    /**
     * @pre n >= 0, n < MAX_VAL
     * @post n <= 1 $implies $ret = n
     * @post n > 1 $implies $ret = getFib(n-1) + getFib(n-2)
     * @post_imp: lastCalculated >= n
     */
    public long getFib(int n) {
        if (n == 0) { return 0;}
        calcFibForI(n);
        return fibArr[n];
    }

    private void calcFibForI(int n) {
        for (int i = lastCalculated + 1; i <= n; i++) {
            fibArr[i] = fibArr[i-1] + fibArr[i-2];
        }
        lastCalculated = n;
    }

private long[] getFibArray(){
    return this.fibArr;
}

    public static void main(String[] args) {
        Fib fib = new Fib(5);
        //fib series: 0, 1, 1, 2, 3, 5, 8
        System.out.println(fib.getFib(1)); //1
        System.out.println(fib.getFib(6)); //8
    }
}

```

תוקן בזמן הבחינה: נא  
להתעלם מ getFibArray

שאלה זו מצריכה נימוק. עבור כל טענה שנבחרה כנכונה, עליכם להסביר בקצרה באיזה מקרה היא מתקיימת.

שאלה 1:

לפניך שלוש טענות המתייחסות לשירות `getFib`:

טענה 1: השירות `getFib` מפר את האינוריאנטה של המחלקה.

טענה 2: השירות `getFib` מפר את החוזה של עצמו.

טענה 3: יתכן מצב שבו `getFib` יחשב פעמיים את הערך של מספר פיבונאצ'י כלשהו.

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.

שאלה זו מצריכה נימוק. עבור כל טענה שנבחרה כנכונה, עליכם להסביר בקצרה באיזה מקרה היא מתקיימת.

שאלה 2:

נמיר את `lastCalculated` להיות סטטי. מה מהתופעות הבאות אפשרי עבור משתמשת של המחלקה `Fib`?

אופציה 1: `getFib` תחזיר 0 עבור קלט שאינו 0.

אופציה 2: `getFib` תחזיר ערך לא נכון שאינו 0 עבור קלט כלשהו שאינו 0.

אופציה 3: יתכן שנקבל שגיאת זמן ריצה על גישה לאינדקס לא חוקי במערך.

בחר/י בתשובה הטובה ביותר:

א. רק אופציה 1 אפשרית.

ב. רק אופציה 2 אפשרית.

ג. רק אופציה 3 אפשרית.

ד. רק אופציות 1+2 אפשריות.

ה. רק אופציות 1+3 אפשריות.

ו. רק אופציות 2+3 אפשריות.

ז. כל האופציות אפשריות.

ח. כל האופציות לא אפשריות.

שאלה זו מצריכה נימוק. במידה ויהיה שינוי מאופן הפעולה של המימוש המקורי, הסבירו כיצד השינוי יקרה.

שאלה 3:

נרצה לשנות את המימוש של הבנאי שמקבל את הפרמטר n באופן הבא: נשמיט את הקריאה ל this(), כך שהמימוש החדש יהיה:

```
public Fib(int n) {
    calcFibForI(n);
}
```

כיצד השינוי ישפיע על ההתנהגות של הקוד בהפעלת השורה הפקודה ?Fib(4)

- א. לא יהיה שינוי מאופן הפעולה של המימוש המקורי.
- ב. נקבל שגיאת **ArrayIndexOutOfBoundsException**
- ג. נקבל שגיאת **NullPointerException**
- ד. לא תיזרק שגיאה, אך התוכנית תחשב ערכים שונים מאלה שהמימוש המקורי מחשב (כלומר, עבור n כלשהו נקבל ערך שונה ממה שהיה מחושב במימוש המקורי).

שאלה זו מצריכה נימוק. במידה ונזרקת שגיאה בסוף הריצה, הסבירו את הסיבה לשגיאה.

שאלה 4:

נרצה להפוך את המחלקה Fib לאיטרטור שעובר על כל איברי הסדרה שחושבו עד כה. לצורך העניין, נעדכן את Fib באופן הבא (הקוד המקורי הושמט משיקולי מקום):

```
public class Fib implements Iterator<Long>{

    public final static int MAX_VAL = 50;
    private int lastCalculated;
    private long[] fibArr ;
    private int currIterIndex;

    @Override
    public boolean hasNext() {
        return currIterIndex <= lastCalculated ;
    }

    @Override
    public Long next() {
        return fibArr[currIterIndex++];
    }
    //the rest of the original code is here
}
```

מה יקרה בהרצת התוכנית הבאה?

```
Fib fib = new Fib(4);
while(fib.hasNext()) {
    long i = fib.next();
    System.out.print(i+ "-");
    while(fib.hasNext()) {
        long j = fib.next();
        System.out.print(j);
    }
    System.out.print("-");
}
```

- א. יודפס 0-01123-1-01123-2-01123-3-01123-5-01123 והתוכנית תסתיים בהצלחה.
- ב. יודפס 0-1123- והתוכנית תסתיים בהצלחה.
- ג. יודפס 0-1-1-2-3- והתוכנית תסתיים בהצלחה.
- ד. יודפס 0-1123-1-123-1-23-2-3-3- והתוכנית תסתיים בהצלחה.

- ה. יודפס 0-01123-1-01123-1-0123-2-01123-3-01123-0 ולאחר מכן תיזרק שגיאת זמן ריצה.  
 ו. יודפס 0-1123-0 ולאחר מכן תיזרק שגיאת זמן ריצה.  
 ז. יודפס 0-1-1-2-3-0 ולאחר מכן תיזרק שגיאת זמן ריצה.  
 ח. יודפס 0-1123-1-123-1-23-2-3-3-0 ולאחר מכן תיזרק שגיאת זמן ריצה.

שאלה זו מצריכה נימוק. במידה והתוכנית לא מסתיימת בהצלחה, הסבירו מדוע. אם כן, הסבירו כיצד נפסקת לולאת ה while.

שאלה 5:

התבוננו בתוכנית הבאה:

```
class Finalize {
    public static short finalizeCounter;

    @Override
    public void finalize() throws Throwable {
        super.finalize();
        finalizeCounter+=1;
    }

    public static void main(String[] args) {
        Finalize f = null;
        while (true) {
            f = new Finalize();
            if (finalizeCounter < -5000) {
                break;
            }
        }
        System.out.println("Done");
    }
}
```

נריץ את התוכנית כך שגודל המקסימלי של ה heap יאותחל לערך לא גדול, אך כזה שמאפשר הרבה איטרציות של לולאת ה while בתוכנית (ע"י הפקודה java -Xmx2M). מה יקרה בהרצת התוכנית?

בחר/י בתשובה הטובה ביותר:

- א. התוכנית תיכנס ללולאה אינסופית.
- ב. תיזרק שגיאת StackOverflowException
- ג. תיזרק שגיאת OutOfMemoryException
- ד. ריצת התוכנית תסתיים ויודפס Done

שאלה זו מצריכה נימוק. עבור כל פונקציה שלא מתקמפלת, הסבירו איזו בעית בטיחות טיפוסים יכולה להיווצר במידה והקוד כן היה מתקמפל.

שאלה 6:

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

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.

וריאציה נוספת:

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

בגירסה הזו f2 ו f2 מתקמפלות

שאלה זו מצריכה נימוק. עבור כל פלט אפשרי, הסבירו מהם הערכים שנוצרים, ואיך ניראים האיברים שעוברים בזרם.

שאלה 7:

```
public class Q7 {
    public static void main(String[] args) {
        Stream<Integer> s = Stream.generate(new RandomNumbers());
        System.out.println(s
            .filter(x->x%2 == 0)
            .peek(x->System.out.print(x + "_"))
            .map(x->x-3)
            .peek(x->System.out.print(x + "_"))
            .allMatch(x->x > 5));
    }
}

class RandomNumbers implements Supplier<Integer>{
    Random random = new Random();

    @Override
    public Integer get() {
        return random.nextInt(12)+1;
    }
}
```

המתודה nextInt(bound) מחזירה ערך אקראי בין 0 (כולל) לבין bound (לא כולל).

מכיוון שאנו עושים שימוש בערכים אקראיים, ריצות שונות יכולות לייצר פלטים שונים. איזה מבין הפלטים הבאים הוא אפשרי בריצת התוכנית? בחר\י בתשובה הטובה ביותר.

- א. true
- ב. false
- ג. 10 7 true
- ד. 9 6 true
- ה. 10 7 6 3 false
- ו. 9 6 3 0 false
- ז. 4 1 6 3 false
- ח. 3 0 6 3 false
- ט. 12 9 false
- י. 12 9 10 7 true
- יא. 12 10 9 6 true
- יב. מלבד תשובה זו יש שתי תשובות נכונות.
- יג. מלבד תשובה זו יש שלוש תשובות נכונות.

שאלה זו מצריכה נימוק. עבור כל טענה נכונה, הסבירו מדוע היא מתקיימת.

שאלה 8:

הסטודנטית שי לא הכירה את קיומו של ה enum ב Java ולכן החליטה לממש בעצמה מחלקה בעלת התנהגות דומה. לפניכם המימוש של שי:

```
class MyEnum{
    public static final MyEnum CLAUDE = new MyEnum("claude");
    public static final MyEnum AUGUSTE = new MyEnum("Auguste");
    public static final MyEnum EDGAR = new MyEnum("Edgar");

    String name;
    private MyEnum(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

להלן מספר טענות הקשורות למימוש:

טענה 1: אם נסיר את final מההגדרה של EDGAR, ניתן יהיה לייצר מצב שבו `MyEnum.EDGAR==MyEnum.CLAUDE` יחזיר True.

טענה 2: לא ניתן לייצר אובייקט שמקיים is-a עם MyEnum, ואינו אחד משלושת העצמים הסטטים שנוצרו במחלקה.

טענה 3: אם נסיר את final מההגדרה של EDGAR, ניתן לייצר מצב שבו `MyEnum.EDGAR.getName()` תחזיר את המחרוזת "Paul".

שימו לב – בכל אחת מהטענות, השינוי המתואר בטענה הוא השינוי היחיד שמבוצע במחלקה MyEnum.

בחר\י בתשובה הטובה ביותר:

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

שאלה זו מצריכה נימוק. הסבירו איזו פונק' מופעלת בכל אחת משתי ההדפסות בתוכנית.

עמוד 10 מתוך 18

שאלה 9:

```
public class Q9 {
    public static void main(String[] args) {
        AA a = new BB();
        BB b = new BB();
        System.out.print(a.getComp("abc").compare(1, 1) + " ");
        System.out.print(b.getComp("abc").compare(1, 1) + " ");
    }
}

class AA{
    public Comparator<Integer> getComp(Object a) {
        return (x,y) -> Integer.compare(x, y);
    }
}

class BB extends AA{
    public Comparator<Integer> getComp(String a) {
        return (x,y)-> super.getComp(a).compare(2*x, y);
    }

    public Comparator<Integer> getComp(Object a) {
        return (x,y)-> super.getComp(a).compare(x, 3*y);
    }
}
```

תזכורת: Integer.compare ערך של 1- כשהאיבר הראשון קטן מהאיבר השני.

מה יודפס בהרצת הקוד הבא?

- א. 1-1
- ב. 1 1
- ג. -1 1
- ד. -1-1
- ה. 0 1
- ו. 0 0
- ז. 1 0
- ח. -1 0
- ט. 0-1

עמוד 11 מתוך 18  
שאלה זו מצריכה נימוק. הסבירו בקצרה עבור כל מימוש אם הוא תקין או לא.

שאלה 10

נתון הקוד הבא:

```
public interface Decidable {
    int i = 0;
    public int decide(char a, char b);
}

public class PrintDecidable {

    public static void printDecision(Decidable item) {
        System.out.print(item.decide('x', 'y'));
        System.out.print(item.decide('y', 'x'));
    }

    public static void main(String[] args) {
        *****;
    }
}
```

נתונים 3 קטעי קוד שונים להחלפת המקטע המסומן ב\*\*\*\*\*:

– מימוש 1

```
printDecision((a,b) -> b-a);
```

– מימוש 2

```
printDecision((a,b) -> Decidable.i++);
```

– מימוש 3

```
printDecision((a,b) -> new Decidable(){
    int j;
    @Override
    public int decide(char a, char b) {
        return j++;
    }
});
```

עבור איזה מימוש הרצת התוכנית תייצר את הפלט 01?

בחר/י את התשובה הנכונה ביותר:

- א. רק מימוש 1.
- ב. רק מימוש 2.
- ג. רק מימוש 3.
- ד. רק מימושים 1+2.
- ה. רק מימושים 1+3.
- ו. רק מימושים 2+3.
- ז. כל אחד מהמימושים לא ייצר את הפלט הרצוי.
- ח. כל אחד מהמימושים ייצרו את הפלט המתאים.

שאלה 11:

```
public class A {
    int p = 8;
    /* some code here */
}
public class B extends A{
    int q = 9;
    public B() {
        q++;
    }
    /* some code here */
}
```

מהו סדר הפעולות שמתרחש ביצירת מופע חדש של המחלקה B שיורשת ממחלקה A?

יצירת מופע חדש - כלומר בהרצת הפקודה `new B()`.

פעולה 1 – אתחול השדה q במחלקה B לערך דיפולטי.

פעולה 2 – אתחול השדה p במחלקה A לערך דיפולטי.

פעולה 3 – השמת השדה q במחלקה B לערך 9.

פעולה 4 – השמת השדה p במחלקה A לערך 8.

פעולה 5 – קריאה לבנאי של המחלקה A.

פעולה 6 – ביצוע הפעולה `q++` מתוך הבנאי של המחלקה B.

בחר/י את סדר הפעולות הנכון ביותר, כאשר הפעולות מסודרות משמאל לימין. כלומר, 1-2 אומר שפעולה 1 מתרחשת קודם ולאחר מכן פעולה 2, ולא מתרחשת ביניהן פעולה אחרת שמוזכרת כאן.

- א. 1-5-2-4-3-6
- ב. 3-5-4-6
- ג. 5-2-4-1-3-6
- ד. 4-6-3
- ה. 5-4-3-6
- ו. 1-3-5-2-4-6
- ז. 1-3-2-4-6
- ח. 5-1-2-4-3-6

```

public class A {
    public static void main(String[] args) {
        MyList<String> list = new MyList<>();
        int res = 0;
        list.add("a");
        list.add("b");
        list.add("c");
        for (Iterator<String> iter = list.iterator(); iter.hasNext();) {
            String current = iter.next();
            if (current.equals("b"))
                res++;
        }
        try {
            for (String current : list) {
                if (current.equals("c"))
                    res++;
            }
        } catch (Exception exp) {
        }

        System.out.println(res);
    }
}

public class MyList<E> ***** {

    List<E> innerList;

    public MyList() {
        innerList = new ArrayList<E>();
    }

    public boolean add(E e) {
        return innerList.add(e);
    }

    public Iterator<E> iterator() {
        return innerList.iterator();
    }
}

```

בחר/י את התשובה הנכונה ביותר:

- א. אם נחליף את \*\*\*\*\* ב `implements Iterable<E>`, הקוד יתקמפל, ירוץ וידפיס 1.
- ב. אם נחליף את \*\*\*\*\* ב `implements Iterable<E>`, הקוד יתקמפל, ירוץ וידפיס 2.
- ג. אם נסיר את ה \*\*\*\*\* ולא נחליף אותן בשום דבר, הקוד יתקמפל אך נקבל שגיאה בזמן ריצה.
- ד. אם נסיר את ה \*\*\*\*\* ולא נחליף אותן בשום דבר, הקוד יתקמפל, ירוץ וידפיס 1.
- ה. אם נסיר את ה \*\*\*\*\* ולא נחליף אותן בשום דבר, הקוד יתקמפל, ירוץ וידפיס 2.

- א. הקוד לא יתקמפל גם אם נחליף את `****` ב `implements Iterable<E>` וגם אם נסיר אותו ללא החלפה.
- ז. מלבד תשובה זו, כל התשובות לא נכונות.
- ח. מלבד תשובה זו יש יותר מתשובה נכונה אחת.

שאלה 13:

שאלה זו עוסקת במחלקות סטטיות מקוננות.

```
class OuterClass {
    private int i;

    //some code
    public static class InnerStaticClass{
        public void func(OuterClass o) {
            System.out.println(o.i);
        }
    }
    //some code
}
```

בחר/י את התשובה הנכונה ביותר:

- א. לא ניתן לכתוב בנאי למחלקה `InnerStaticClass`.
- ב. המימוש של `func` לא מתקמפל.
- ג. `InnerStaticClass` יכולה לרשת ממחלקה אחרת שאינה סטטית.
- ד. לא ניתן ליצור מופע של `InnerStaticClass` ללא מופע של `OuterClass`.
- ה. בכל יצירת מופע של `OuterClass` נוצר גם מופע של `InnerStaticClass`.
- ו. מלבד תשובה זו, כל התשובות לא נכונות.
- ז. מלבד תשובה זו יש יותר מתשובה נכונה אחת.

שאלה 14:

בחר/י את התשובה הנכונה ביותר: הניחו כי כל החלפה נעשית רק על פי מה שמופיע בטענות, ללא קוד חישובי נוסף. למשל, אם נרצה להמיר בלוק `switch` בפקודות `if` בלבד, זה יהיה השינוי היחיד ולא יעשו שינויים נוספים בקוד.

- טענה 1 – ניתן להחליף כל בלוק `switch` במספר סופי של פקודות `if` ללא שימוש ב `else`.
- טענה 2 – ניתן להחליף כל בלוק של פקודות `if` ללא `else` בבלוק `switch` עם מספר `case`-ים סופי.
- טענה 3 – ניתן להחליף כל בלוק המכיל `if` אחד ו `else` באופרטור טרנטרי `( cond? X : Y )`.

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

שאלה זו מצריכה נימוק. הסבירו בקצרה עבור כל פונקציה שלא מתקמפלת מדוע זה קורה . עמוד 15 מתוך 18

שאלה 15

```
public class Box<T>{
    private T elem;

    public Box(T elem) {
        this.elem = elem;
    }

    public T getElem() {
        return this.elem;
    }
}

public class Test<T>{
    public <K extends Box<Integer>> K func1(K box, Integer newElem){
        return new Box<>(newElem);
    }

    public <K extends Box<Integer>> K func2(K box, Integer newElem){
        return new K<>(newElem);
    }

    public Box<T> func3(Box<T> box, Integer elem){
        return new Box<Integer>(elem);
    }
}
```

מי מבין המתודות func1, func2, func3 מתקמפלת?

בחר/י את התשובה הנכונה ביותר:

- א. רק func1 מתקמפלת.
- ב. רק func2 מתקמפלת.
- ג. רק func3 מתקמפלת.
- ד. רק func1 ו func2 מתקמפלות.
- ה. רק func1 ו func3 מתקמפלות.
- ו. רק func2 ו func3 מתקמפלות.
- ז. כל המתודות מתקמפלות.
- ח. כל המתודות לא מתקמפלות.

וריאציה נוספת: func3 מחזירה <?>Box. במקרה זה f3 תתמפל.

עמוד 16 מתוך 18 **שאלה זו מצריכה נימוק. עבור כל טענה נכונה הציגו מימוש מתאים.**

שאלה 16:

שאלה זו מתייחסת למימושן של מחלקות (ולא ממשקים) A, B, C שמימושן אינו נתון. ניתן להניח את קיומן של מחלקות נוספות ללא שום מגבלות.

```
public void func(B b1, B b2) {  
    A a1 = (A) b1; // line 1  
    A a2 = (C) b2; // line 2  
}
```

להלן מספר טענות לגבי קוד זה:

טענה 1: ניתן לממש את A, B ו-C כך ששתי השורות יתקמפלו ולכל ריצה של הקוד לא תיזרק שגיאת זמן ריצה.

טענה 2: ניתן לממש את A, B ו-C כך ששתי השורות יתקמפלו ויתכנו שתי ריצות שונות שבאחת תיזרק שגיאת זמן ריצה בשורה 1 ובשניה תיזרק שגיאת זמן ריצה בשורה 2.

טענה 3: ניתן לממש את A, B ו-C כך שרק שורה 2 תתקמפל.

בחרו את התשובה הנכונה ביותר:

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

```

class Exp {
    public static void main(String[] args) {
        int x = 0;
        int[] a = { 1, 2, 3 };
        try {
            for (int i : a) {
                if (i % 2 == 0)
                    throw new Exception();
                else
                    throw new RuntimeException();
            }
        }
        catch (RuntimeException e) {
            x = x - 4;
        }
        catch (Exception e) {
            x++;
        }
        finally {
            x++;
        }
        System.out.println(x);
    }
}

```

מה יקרה בהרצת הקוד הבא?

- א. תיזרק שגיאה לפני שהתוכנית תדפיס פלט כלשהו.
- ב. ריצת התוכנית תסיים בהצלחה ויודפס 0.
- ג. ריצת התוכנית תסתיים בהצלחה ויודפס -6.
- ד. ריצת התוכנית תסיים בהצלחה ויודפס -4.
- ה. ריצת התוכנית תסיים בהצלחה ויודפס -3.
- ו. ריצת התוכנית תסיים בהצלחה ויודפס -2.
- ז. ריצת התוכנית תסיים בהצלחה ויודפס 1.
- ח. ריצת התוכנית תסיים בהצלחה ויודפס -1.
- ט. ריצת התוכנית תסתיים בהצלחה ויודפס 2.
- י. ריצת התוכנית תסתיים בהצלחה ויודפס 3.
- יא. ריצת התוכנית תסתיים בהצלחה ויודפס 4.

```

public class A {
    protected String str;
    // some initialization of str
}

public class B extends A {
    protected String str;
}

public class C extends B {
    protected String str;
    public static void main(String[] args) { new C().printStrOfA(); }
    public void printStrOfA() { System.out.println("*****"); } //$
}

```

לפניכם הקוד החלקי של המחלקה A, הקוד המלא של המחלקה B אשר יורשת מ-A, והקוד של מחלקה C אשר יורשת מ-B כאשר מוסתר רק החלק שמופיע ב-\*\*\*\*\* בתוך פקודת ההדפסה בשורה \$.

ניתן לראות כי בכל המחלקות מוגדר שדה מופע מטיפוס String בשם str. ידוע כי הקוד המוסתר במחלקה A רק מאתחל את השדה str אך לא חושף את ערכו.

הסטודנטית שי מממשת את המחלקה C. במתודת ה-main היא יוצרת מופע של C וקוראת עליו למתודה printStrOfA ובה יש פקודת הדפסה בשורה \$.

שי חולמת להחליף את ה-\*\*\*\*\* בפקודה אשר תוביל להדפסה של הערך של השדה str שהוגדר במחלקה A ממנה היא יורשת בעקיפין (ולא של 2 השדות בעלי אותו השם שמופיעים ב-B או ב-C).

לפניכם מספר הצעות לקוד שניתן לשים במקום ה-\*\*\*\*\* להגשמת חלומה של שי.

הצעה 1: `super.super.str`

הצעה 2: `((B) this).super.str`

הצעה 3: `((A) this).str`

מבין ההצעות הללו, מהן ההצעות המתאימות? בחר/י בתשובה הטובה ביותר.

א. כל ההצעות לא מתאימות הצעה.

ב. הצעה 1 בלבד.

ג. הצעה 2 בלבד.

ד. הצעה 3 בלבד.

ה. הצעות 1+2 בלבד.

ו. הצעות 2+3 בלבד.

ז. הצעות 1+3 בלבד.

ח. כל ההצעות מתאימות.

שאלה 19:

לפניכם שלוש טענות כלליות על עבודה ב Java:

טענה 1: כאשר מחלקה X מבצעת import למחלקה Y, הקוד של Y מועתק לתוך X בזמן קומפילציה.

טענה 2: ה classpath משמש אותנו בשביל להגדיר את את המיקום אליו יכתבו קבצי ה class הנוצרים בתהליך הקומפילציה של קבצי java.

טענה 3: קוד שקומפל על מערכת הפעלה מסויימת יוכל לרוץ רק על מערכת הפעלה זו. אם נרצה להריץ אותו על מערכת הפעלה אחרת, עלינו לקמפל אותו מחדש ולהגדיר לאיזו מערכת הפעלה הוא מיועד.

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.

```
public class Test {
    public static void main(String[] args) {
        Base base = new Sub();
        System.out.println(base.func());
    }
}

public class Base{
    private int foo() {
        return 1+goo();
    }

    public int goo(){
        return 1;
    }
    public int func() {
        return 1+foo();
    }
}

public class Sub extends Base{
    private int foo() {
        return 2;
    }

    public int goo() {
        return 3*foo();
    }
    public int func() {
        return super.func() + foo();
    }
}
```

מה יודפס בהרצת התוכנית?

בחר\י בתשובה הטובה ביותר:

- א. 4
- ב. 5
- ג. 6
- ד. 7
- ה. 8
- ו. 9
- ז. 10