

מספר סידורי: \_\_\_\_\_ מספר ת"ז: \_\_\_\_\_

## בחינה בתוכנה 1

סמסטר ב' תשע"ח, מועד א', 20 ביולי 2018  
לנה דנקין, ברית יונגמן, שי גרשטיין

משך הבחינה שלוש שעות.

סך הניקוד על השאלות בבחינה הוא 105, אך הציין 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.

יש לסמן את התשובה הטובה ביותר בתשובון. לא יינתן ניקוד על סימון תשובה בטופס הבחינה, במחברת הבחינה, או בטופס הנימוקים.

יש לנמק בתמצות את כל התשובות בטופס הנימוקים המצורף בלבד. נימוק חסר או לא נכון עלול לגרום לאי קבלת נקודות על שאלה במקרים שבהם יוחלט לקבל יותר מתשובה אחת. נימוק לתשובה אחרת מזו שסומנה על גבי טופס התשובות לא יתקבל.

יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תפסל. תשובות ונימוקים במחברת הבחינה לא יבדקו.

יש למלא מספר סידורי (מספר מחברת) ומספר ת"ז על כל דף של טופס הבחינה.

אסור השימוש בחומר עזר כלשהו, כולל מחשבוניו או כל מכשיר אחר פרט לעט.

לטופס הבחינה מצורף דף לשאלות הסטודנטים. יש לכתוב שאלות שמתעוררות במהלך הבחינה בדף זה ולהעביר לסגל הקורס. שאלות ענייניות תענינה על ידי סגל הקורס בפני כל הנבחנות/ים.

© 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!

1. שאלה 1:

```
public class Box<V>{  
  
    public <T> void func1(Set<T> s1, T item){  
        s1.add(item);  
    }  
  
    public void func2(Set<?> s2, Object item){  
        s2.add(item);  
    }  
  
    public void func3(Set<? extends Exception> s3, IOException exp){  
        s3.add(exp);  
    }  
  
    public void func4(Set s4, Object item){  
        s4.add(item);  
    }  
}
```

אילו מבין הפונקציות func המופיעות במחלקה Box מתקמפלות? בחר/י את התשובה הטובה ביותר.

- א. רק מימושים func1+func2 מתקמפלים.
- ב. רק מימושים func1+func3 מתקמפלים.
- ג. רק מימושים func1+func4 מתקמפלים.
- ד. רק מימושים func2+func3 מתקמפלים.
- ה. רק מימושים func2+func4 מתקמפלים.
- ו. רק מימושים func3+func4 מתקמפלים.
- ז. רק פונקציה אחת מתקמפלת.
- ח. כל הפונקציות לא מתקמפלות.

2. שאלה 2

```
public class Top {
    public Top (int id) { this.id = id; }
    private int id;
    private Inner in;

    public class Inner {
        private int id = 3; // X
        public void f() {
            System.out.println(this.id * Top.this.id);
        }
    }

    public Inner create() {
        return new Inner();
    }

    public static void main(String[] args) {
        Top t1 = new Top(1);
        Top t2 = new Top(2);
        t2.in = t1.create();
        t2.in.f();
    }
}
```

- א. התוכנית לא מתקמפלת, בגלל שגיאת קומפילציה בשורה X.
- ב. התוכנית מתקמפלת, אך יש שגיאת זמן ריצה.
- ג. אין שגיאת קומפילציה או ריצה, והתוכנית מדפיסה 1.
- ד. אין שגיאת קומפילציה או ריצה, והתוכנית מדפיסה 2.
- ה. אין שגיאת קומפילציה או ריצה, והתוכנית מדפיסה 3.
- ו. אין שגיאת קומפילציה או ריצה, והתוכנית מדפיסה 4.
- ז. אין שגיאת קומפילציה או ריצה, והתוכנית מדפיסה 6.
- ח. אין שגיאת קומפילציה או ריצה, והתוכנית מדפיסה 9.

שאלות 3 ו 4 מתייחסות לקטע הקוד הבא.

```
public class Granny{
    public void func(){ System.out.print("Granny's func"); }
}

public class Mom extends Granny{
    private void f(){ System.out.print("Mom's f ; "); }

    public void foo(){
        f();
        func(); //$$$
    }
}

public class Daughter extends Mom{
    public void f(){ System.out.print("Daughter's f ; ");} //***

    public void func(){ System.out.print("Daughter's func");}

    /*
    public void foo(){
        f();
        super.func(); //###
    }*/

    public static void main(String[] args){
        Mom mom = new Daughter();
        mom.foo();
    }
}
```

### 3. שאלה 3:

יש להתעלם בשאלה זו, מהקטע קוד המופיע בהערה.

מה יקרה בהרצת התוכנית Daughter? בחר/י את התשובה הטובה ביותר:

- א. לא ניתן להריץ את התוכנית. ישנה שגיאת קומפילציה בשורה המסומנת ב \$\$\$
- ב. לא ניתן להריץ את התוכנית. ישנה שגיאת קומפילציה בשורה המסומנת ב \*\*\*

ג. יודפס Mom's f ; Daughter's func

ד. יודפס Daughter's f ; Daughter's func

ה. יודפס Mom's f ; Granny's func

ו. יודפס Daughter's f ; Granny's func

### 4. שאלה 4:

כעת, נוריד את ההערה מהמימוש של foo במחלקה daughter. מה יקרה בהרצת התוכנית Daughter?

בחר/י את התשובה הטובה ביותר:

א. התוכנית לא התקמפלה ותמשיך לא להתקמפל.

ב. התוכנית התקמפלה אך לאחר הוספת המימוש של foo ל daughter יש שגיאת קומפילציה

בשורה המסומנת ב ###

ג. יודפס Mom's f ; Daughter's func

ד. יודפס Daughter's f ; Daughter's func

ה. יודפס Mom's f ; Granny's func

ו. יודפס Daughter's f ; Granny's func

5. שאלה 5:

להלן מספר טענות על מבנה הזכרון של Java. מבין הטענות הבאות, איזו טענה אינה נכונה? במידה וכל הטענות נכונות, בחרו בתשובה: כל הטענות נכונות.

- א. שדות סטטיים נשמרים על ה heap.
- ב. משתנים מקומיים של פונקציות נשמרים על ה stack.
- ג. אובייקטים נשמרים על ה heap.
- ד. במצב של קריאה רקורסיבית אינסופית יזרק חריג מסוג StackOverflowError.
- ה. **שדות מופע נשמרים על ה stack.**
- ו. כל הטענות נכונות.

6. שאלה 6:

```
public class Test<T extends Comparable>{  
  
    public static void main(String[] args){  
        List<String> strList = Arrays.asList("abc", "def");  
        System.out.println(func(strList));  
    }  
  
    public static boolean func(List<*****> lst){  
        return lst.get(0).compareTo(lst.get(1)) == 0;  
    }  
}
```

לפניכם מספר אופציות בהן נוכל להחליף את \*\*\*\*\*

אופציה 1: extends Comparable ?

אופציה 2: T

אופציה 3: Comparable

עבור אילו מבין האופציות הקוד של Test יתקמפל?

- א. **אופציה 1 בלבד.**
- ב. אופציה 2 בלבד.
- ג. אופציה 3 בלבד.
- ד. אופציות 1+2 בלבד.
- ה. אופציות 1+3 בלבד.
- ו. אופציות 2+3 בלבד.
- ז. עבור כל האופציות.
- ח. עבור אף אחד מבין האופציות.

7. שאלה 7:  
התבונני בקוד הבא:

```
public class Test{
    public static void main(String[] args){
        Stream<Integer> s= Stream.generate(new NaturalNumbers());
        Optional<Integer> mV =
            s.filter(x-> {System.out.println(x);
                        return x <= 10;})
            .max((x,y)-> Integer.compare(x, y));
        System.out.println(mV.isPresent() ? mV.get() : "no max value");
    }
}

class NaturalNumbers implements Supplier<Integer> {
    private int i;

    @Override
    public Integer get() {
        return ++i;
    }
}
```

התיעוד של השירות max:

```
Optional<T> max(Comparator<? super T> comparator)
Returns the maximum element of this stream according to the provided Comparator.
```

- א. התוכנית תסתיים ובסיומה יודפס 10
- ב. התוכנית תסתיים ובסיומה יודפס הערך של Integer.MAX\_VALUE (הערך המקסימלי ל int).
- ג. התוכנית תסתיים ובסיומה יודפס no max value
- ד. התוכנית תיכנס ללולאה אינסופית ולא יודפס שום פלט במהלך ריצתה.
- ה. התוכנית תסתיים ולא יודפס שום פלט במהלך ריצתה.
- ו. מלבד תשובה זו יש יותר מתשובה אחת נכונה.
- ז. **מלבד תשובה זו, כל התשובות לא נכונות.**



8. שאלה 8:

```
class A{
    public A(){
    }

    public A(int i){
    }
}

class B extends A{
    public B(){ //c1
    }

    public B(int i){ //c2
        this();
    }

    public B(int i, int j){ //c3
        super();
    }
}
```

בחר/י בתשובה הטובה ביותר:

- א. הקוד של B אינו מתקמפל. אם נוריד את c1, הקוד יתקמפל.
- ב. הקוד של B אינו מתקמפל. אם נוריד את c2, הקוד יתקמפל.
- ג. הקוד של B אינו מתקמפל. אם נוריד את c3, הקוד יתקמפל.
- ד. הקוד של B אינו מתקמפל. אם נוריד את c1 וגם את c2, הקוד יתקמפל.
- ה. הקוד של B אינו מתקמפל. אם נוריד את c1 וגם את c3, הקוד יתקמפל.
- ו. הקוד של B אינו מתקמפל. אם נוריד את c2 וגם את c3, הקוד יתקמפל.
- ז. **הקוד של B מתקמפל. אם נוריד את כל הבנאים של B, הקוד ימשיך להתקמפל.**
- ח. מלבד תשובה זו, כל התשובות לא נכונות.

9. שאלה 9:

```
public class Test {
    public static void main(String[] args) throws Exception {
        class OddLengthException extends Exception{
        }

        List<String> lst = Arrays.asList("a", "ab", "abc", null, "abcd");
        for (String s : lst) {
            try {
                if (s.length() % 2 == 1) {
                    throw new OddLengthException();
                }
            }
            catch (OddLengthException exp){
                if (s.length() > 2) {
                    throw new Exception();
                }
            }
            finally {
                System.out.print(s + " ");
            }
        }
    }
}
```

- מה יקרה בהרצת התוכנית? בחר/י את התשובה הטובה ביותר
- א. יודפס a ab abc null abcd והתוכנית תסתיים ללא חריג (Exception).
  - ב. יודפס a ab abc null abcd והתוכנית תעוף על חריג (Exception).
  - ג. יודפס a ab abc null abcd והתוכנית תסתיים ללא חריג (Exception).
  - ד. **יודפס a ab abc null abcd והתוכנית תעוף על חריג (Exception).**
  - ה. יודפס a ab abc null abcd והתוכנית תסתיים ללא חריג (Exception).
  - ו. יודפס a ab abc null abcd והתוכנית תעוף על חריג (Exception).

10. שאלה 10

- להלן ארבע טענות הקשורות לטיפוסי מניה (enum):
- טענה 1: לטיפוס המניה ניתן להגדיר מתודות אבסטרקטיות
  - טענה 2: לטיפוס המניה ניתן להגדיר שדות מופע.
  - טענה 3: לטיפוס המניה ניתן להגדיר שירותי מופע, אך הם חייבים להיות מוגדרים כפרטיים (private)
  - טענה 4: לטיפוס המניה ניתן להגדיר בנאים.
- בחר/י את התשובה הטובה ביותר:
- א. כל הטענות נכונות
  - ב. אף טענה אינה נכונה
  - ג. **ישנן בדיוק 3 טענות נכונות**
  - ד. ישנן בדיוק שתי טענות נכונות
  - ה. ישנה בדיוק אחת נכונה



11. שאלה 11

```
public class A {  
    public int foo(int i, int j) throws IOException { . . . }  
}  
  
public class B extends A {  
    /* public double foo(int i, int j) throws IOException { . . . } */  
    /* public int foo(double i, int j) throws Exception { . . . } */  
    /* public double foo(double i, int j) { . . . } */  
    /* protected int foo(int i, int j) { . . . } */  
}
```

- הניחו כי בכל פעם נוסף מתודה אחת בלבד למחלקה B. בחרו בתשובה הנכונה ביותר.
- עבור כל מתודה שנוסף הקוד יתקמפל.
  - עבור כל מתודה שנוסף תיווצר שגיאת קומפילציה.
  - ישנן בדיוק שתי מתודות אשר הוספת כל אחת מהן תגרור שגיאת קומפילציה
  - ישנה בדיוק אחת אשר הוספתה תגרור שגיאת קומפילציה
  - ישנן בדיוק שלוש מתודות אשר הוספת כל אחת מהן תגרור שגיאת קומפילציה

12. שאלה 12

```
public class A {  
    protected String str = "A";  
    public static int i = 1;  
  
    public A() {foo();}  
    public void foo() {System.out.println(str + " "+i);}  
  
public class B extends A {  
    protected String str = "B"; // *  
    public static int i = 2; // **  
  
    public B() {foo();}  
    public void foo() {System.out.println(str + " "+i);}  
  
public class C {  
    public static void main(String[] args) {  
        A a = new B();  
        B b = new B();  
    }  
}}
```

- ישנה שגיאת קומפילציה בשורה המסומנת ב-\*
- ישנה שגיאת קומפילציה בשורה המסומנת ב-\*\*
- יודפס: null 2 B 2 null 2 B 2
- יודפס: A 2 B 2 A 2 B 2
- יודפס: A 1 B 2
- יודפס: A 1 B 2 A 1 B 2
- יודפס: A 1 B 2 null 2 B 2
- יודפס: null 2 B 2

13. שאלה 13

```
public /*final*/ class A {  
  
    public final int j = 1;  
    public static int k = 2;  
  
    public void foo() {System.out.println(j+k);}  
  
    public static void main(String[] args)  
    {  
        A a = new A() { /*  
  
            public final int j = 3;    /**  
            public void foo() {  
                System.out.println(j+k);} /****  
  
        };  
        a.foo();  
    }  
}
```

בחר/י את התשובה הנכונה ביותר:

- א. ישנה שגיאת קומפילציה בשורה המסומנת ב-\*\*, וזה נכון גם אם A הייתה מוגדרת להיות final
- ב. ישנה שגיאת קומפילציה בשורה המסומנת ב-\*\*, וזה נכון גם אם A הייתה מוגדרת להיות final
- ג. יודפס 5 וזה נכון גם אם A הייתה מוגדרת להיות final
- ד. יודפס 3 וזה נכון גם אם A הייתה מוגדרת להיות final
- ה. יודפס 5. אם A הייתה מוגדרת final היה מודפס 3
- ו. יודפס 5. אם A הייתה מוגדרת final הייתה שגיאת קומפילציה בשורה המסומנת ב-\*
- ז. יודפס 3. אם A הייתה מוגדרת final הייתה שגיאת קומפילציה בשורה המסומנת ב-\*

14. שאלה 14

```
public class ShellWithbutton {
    private static int i = 0;

    public static void main(String[] args) {
        Display display = Display.getDefault();
        Shell shell = new Shell(display);
        shell.setLayout(new FillLayout(SWT.VERTICAL));
        shell.setText("example1");
        Button ok = new Button(shell, SWT.PUSH);
        ok.setText("i = "+i);
        ok.addSelectionListener(*****); //missing code goes here
        shell.pack();
        shell.open();
        while (!shell.isDisposed()) {
            if (!display.readAndDispatch())
                display.sleep();
        }
        display.dispose();
    }
}
```

- נרצה כי בכל לחיצה על הכפתור, הערך של המשתנה i יגדל ב-1 וערכו החדש יופיע על הכפתור
- טענה 1: ניתן לממש התנהגות זאת על ידי כתיבת מאזין שמוגדר כמחלקה אנונימית אשר מממשת את המנשק SelectionListener בתוך ShellWithButton.
- טענה 2: ניתן לממש התנהגות זאת על ידי כתיבת מאזין שמממש את המנשק SelectionListener, ומוגדר כמחלקה פנימית בתוך המחלקה ShellWithButton
- טענה 3: ניתן לממש התנהגות זאת על ידי כתיבת מאזין שמממש את המנשק SelectionListener, ומוגדר כמחלקה חדשה בקובץ נפרד.
- בחר/י בתשובה הטובה ביותר:
- רק טענה 1 נכונה
  - רק טענות 1 ו-2 נכונות
  - כל הטענות נכונות
  - טענה 1 נכונה וטענה 2 תהיה נכונה רק אם i יוגדר להיות public
  - כל הטענות לא נכונות. אם i יוגדר להיות public רק טענה 1 תהיה נכונה
  - רק טענה 1 נכונה. אם i יוגדר להיות לא סטטי, גם טענה 2 תהיה נכונה
  - רק טענה 3 נכונה

15. שאלה 15:

```
public class A{
    private int i;
    public A(int i) {
        this.i = i;
    }
    public int hashCode() {
        return i;
    }
    public boolean equals(Object obj) {
        return new Random().nextBoolean();
    }

    public static void main(String[] args) {
        Set<A> aSet = new HashSet<>();
        aSet.add(new A(1));
        aSet.add(new A(1));
        aSet.add(new A(2));
        System.out.println(aSet.contains(new A(1)));
        System.out.println(aSet.size());
    }
}
```

המימוש של המחלקה A אינו דטרמיניסטי (ולא מקיים את דרישות החוזה של equals), ולכן בריצות שונות יכולים להתקבל פלטים שונים. להלן מספר טענות הנוגעות לפלטים האפשריים בהרצת התוכנית טענה 1: ניתן לקבל פלט שבו יש true בשורה הראשונה, וניתן לקבל פלט שבו מופיע false בשורה הראשונה.

טענה 2: ניתן לקבל פלט שבו מופיע 2 בשורה השניה.  
טענה 3: ניתן לקבל פלט שבו מופיע 3 בשורה השניה.

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.

16. שאלה 16

```
public class ScannerQ {  
  
    public static void main(String[] args) {  
        Scanner s = new Scanner("A#B#C D# E");  
        s.useDelimiter("#");  
        s.next();  
        s.next();  
        String text = "";  
        while (s.hasNextLine()) {  
            text += s.nextLine() + ' ';  
        }  
        System.out.println(text);  
        s.close();  
    }  
}
```

בחר/י את התשובה הנכונה ביותר.

- א. התוכנית תדפיס: #C D# E
- ב. התוכנית תדפיס: C D# E
- ג. התוכנית תדפיס: C D E
- ד. התוכנית תדפיס: #B#C D# E
- ה. התוכנית תדפיס: B#C D# E
- ו. התוכנית תדפיס מחרוזת ריקה.

17. שאלה 17

בחנו את הטענות הבאות וקבעו אילו מהן נכונות.

- טענה 1: במחלקה אבסטרקטית (abstract) מתודה המכילה מימוש אינה יכולה לקרוא למתודה אבסטרקטית של אותה המחלקה.
  - טענה 2: אם מחלקה A מממשת את ממשק I, אז ביצוע פעולת downcasting ממשתנה מטיפוס I למשתנה מטיפוס A תמיד תתקמפל.
  - טענה 3: אם מחלקה אבסטרקטית מממשת ממשק, אז היא מוכרחה לספק מימוש לכל המתודות האבסטרקטיות של אותו הממשק.
- 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

18. שאלה 18

```
public class A {
    public void f() throws Exception {
        throw new RuntimeException();
    }
}

public class B extends A {
    public void f() throws RuntimeException { // *
        throw new RuntimeException();}

    public static void main(String[] args) throws RuntimeException {
        A a = new B();
        a.f();// **
    }
}
```

בחר/י בתשובה הטובה ביותר.

- א. ישנה שגיאת קומפילציה בשורה המסומנת ב- \*.
- ב. ישנה שגיאת קומפילציה בשורה המסומנת ב- \*\*.
- ג. הקוד מתקמפל. אם נסיר את ההצהרה על החריג מהחתימה של main (כלומר: נסיר מהחתימה את צמד המילים throws RuntimeException), נקבל שגיאת קומפילציה בשורה המסומנת ב- \*\*.
- ד. הקוד מתקמפל. אם נסיר את ההצהרה על החריג מהחתימה של main (כלומר אם נסיר מהחתימה את צמד המילים throws RuntimeException), הקוד ימשיך להתקמפל.
- ה. מלבד תשובה זו כל התשובות לא נכונות.

19. שאלה 19:

```
public class Test{
    public static void main(String[] args) {
        String[] arr1 = {new String("a"), new String("b")};
        String[] arr2 = {new String("a"), arr1[1]};
        String[] arr3 = arr2;
        int res = 0;
        if (arr1 == arr2)
            res += 1;
        if (arr1[1] == arr2[1])
            res += 2;
        if (arr3 == arr2)
            res += 4;
        System.out.println(res);
    }
}
```

מה יודפס בהרצת הקוד הבא? בחר/י בתשובה הטובה ביותר:

- א. 0
- ב. 1
- ג. 2
- ד. 3
- ה. 4
- ו. 5
- ז. 6
- ח. 7

שאלות 20 ו 21 מתייחסות למחלקה MultiMap:

```
public class MultiMap<K, V>{
    private Map<K, List<V>> rep;
    private Map<K, Iterator<V>> iters;

    public MultiMap() {
        rep = new HashMap<>();
        iters = new HashMap<>();
    }

    /*
     * @pre iters.get(key) != null //pre1
     */
    public boolean keyHasNext(K key) {
        return iters.get(key).hasNext();
    }

    /*
     * @pre: iters.get(key) != null && keyHasNext(key) == true; //pre2
     */
    public V nextForKey(K key) {
        return iters.get(key).next();
    }

    /*
     * @pre: rep.get(key) != null //pre3
     */
    public void initIterForKey(K key) {
        iters.put(key, rep.get(key).iterator());
    }

    public void put(K key, V value) {
        List<V> values = rep.get(key);
        if (values == null) {
            values = new ArrayList<>();
            rep.put(key, values);
        }
        values.add(value);
    }
}
```

המחלקה מגדירה מבנה נתונים דמוי מילון המאפשר שמירת יותר מערך (value) אחד עבור כל מפתח (key). בנוסף, המחלקה מאפשרת מעבר באיטרטור על רשימת הערכים עבור כל מפתח. הקוד כתוב כך שקיימת דרך נכונה לשמור ערכים ולעבור עליהם באיטרציות מבלי לקבל חריגי זמן ריצה.

20. שאלה 20:

- למחלקה הוגדרו שלושה תנאי קדם (pre). בחנ'י אותם ובחר'י בתשובה הטובה ביותר.
- תנאי ה pre המוגדרים מספיקים על מנת להבטיח שימוש נכון ובטוח. אם נסיר אותם, המשתמש יוכל לקבל חריג מסוג NullPointerException בזמן הקריאה לשירותי המחלקה.
  - תנאי ה pre המוגדרים מספיקים על מנת להבטיח שימוש נכון ובטוח. המחלקה תהיה בטוחה לשימוש ללא חריגי זמן ריצה גם אם נסיר את תנאי ה pre.
  - תנאי ה pre אינם מתאימים. אם נוסיף תנאי pre לשירות put המחלקה תהיה בטוחה לשימוש ללא חריגי זמן ריצה.
  - תנאי ה pre אינם מתאימים. ניתן לנסח את שלושת תנאי ה pre באופן שונה, מבלי לבצע שינויים במימוש המחלקה, על מנת להבטיח שהמחלקה תהיה בטוחה לשימוש ללא חריגים.

ה. מלבד תשובה זו כל התשובות לא נכונות.



21. שאלה 21:

נרצה להרחיב את המחלקה כך שתתמוך בשמירת הערכים באוספים (Collections) שונים מלבד רשימות. לדוגמא, נרצה ש MultiMap מסויים ישמור את הערכים שלו ב Set במקום ב List.


הסטודנטית שי הציעה לעדכן את המחלקה MultiMap באופן הבא על ידי ארבעה שינויים המסומנים בהערות.

```
class MultiMap<T, V, K extends Collection<V>>{ //1
    private Map<T, K> rep; //2
    private Map<T, Iterator<V>> iters;

    /* unchanged code here */

    public void put(T key, V value) {
        K values = rep.get(key); //3
        if (values == null) {
            values = new K(); //4
            rep.put(key, values);
        }
        values.add(value);
    }
}
```

בחר/י את התשובה הטובה ביותר:

- א. השינויים ששי הציעה אכן מגדירים MultiMap התומך בשמירת הערכים באוספים מטיפוסים שונים.
- ב. השינויים ששי הציעה אינם טובים כיוון ששינוי 1 לא מתקמפל.
- ג. השינויים ששי הציעה לא מתאימים כיוון ששינוי 4 לא מתקמפל. 
- ד. השינויים ששי הציעה מתקמפלים וניתן להריץ את הקוד ללא חריגים, אך הקוד לא עובד כמצופה ולא מאפשר שמירה של הערכים באוספים מטיפוסים שונים.
- ה. השינויים ששי הציעה מתקמפלים, אך בזמן ריצה אנו עשויים לקבל חריג מסוג ClassCastException בשורה שבה בוצע שינוי 3.