

# בחינה בתוכנה 1

מועד ב', סמסטר א' + ב', תשע"ג  
14.8.2013

דן הלפרין, ליאור וולף  
יעל אמסטרדמר, ניר אטיאס, דביר נתנאלי ולנה דנקין

## הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות**, חלקו את זמנכם ביעילות.
- אסור השימוש בחומר עזר כלשהו, כולל מחשבוניו או כל מכשיר אחר פרט לעט. בסוף הבחינה צורף לנוחותכם נספח ובו תיעוד מחלקות שימושיות.
- יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תיפסל. תשובות במחברת הבחינה לא תיבדקנה. במידת הצורך ניתן לכתוב בגב טופס הבחינה.
- יש למלא מספר סידורי (מס' מחברת) ומספר ת.ז על כל דף של טופס הבחינה.
- ניתן להניח לאורך השאלה שכל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import.
- במקומות בהם תתבקשו לכתוב מתודה (שירות), ניתן לכתוב גם מתודות עזר.
- ניתן להוסיף הנחות לגבי אופן השימוש בשירותים המופיעים בבחינה, ובלבד שאין הן סותרות את תנאי השאלה. יש לתעד הנחות אלו כחווה (תנאי קדם, תנאי בתר) בתחביר המקובל, שיכתב בתחילת השירות.
- יש להתייחס בכבוד רב לצוות המשגיחים.

לשימוש הבודקים בלבד:

שאלה	ניקוד	א	ב	ג	ד	ה	סה"כ
1	35						
2	35						
3	15						
4	15						

## בהצלחה!

כל הזכויות שמורות ©  
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.

## שאלה 1 (35 נקודות)

בשאלה זו נעסוק בעיבוד של טבלאות נתונים דו-מימדיות. כל עמודה בטבלה היא "שדה" המייצג מאפיין אחד וכל שורה היא יחידת מידע אחת (רשומה).

טבלאות הנתונים הן מלבניות, כלומר כל שורה מכילה אותו מספר של נתוני עמודות ולהפך, ומכילות **ערכים מספריים בלבד** (מסוג double). בנוסף, ניתן להניח כי הטבלה אינה ריקה, כלומר היא מכילה לפחות עמודה אחת ושורה אחת. בדוגמאות שניצג, נסיף את שמות העמודות לשם נוחות ההצגה, אולם שמות אלו אינם חלק ממבנה הנתונים.

לדוגמא, בטבלת הציונים הבאה רשומה (שורה) מכילה מידע על ציון של סטודנט מסוים באחד הקורסים בו הוא השתתף.

STUDENTID	COURSEID	GENDER	YEAROFBIRTH	GRADE
8	12	1	1980	95
9	12	-1	1992	87
9	13	-1	1992	92

### סעיף א' (20 נקודות)

ממשו את הפונקציה `groupBy`, המקבצת שורות בעלות מאפיין ספציפי זהה (כלומר, שורות שערך זהה בעמודה נתונה) ומסכמת אותן ברשומה אחת בה כל עמודה, מלבד זו ששימשה לקיבוץ, מחושבת ע"י פונקציית סיכום נתונה המוגדרת ע"י המנשק `ListOperator`:

```
public interface ListOperator {
    double apply(List<Double> values);
}
```

לדוגמא, הפעלת `groupBy` על הטבלה לעיל, עם קיבוץ לפי העמודה הראשונה וסיכום ע"י פעולת מיצוע, תייצר טבלה בעלת שתי רשומות (מספר הסטודנטים). בפרט, הפעולה מחשבת בעמודה האחרונה את ממוצע הציונים של כל סטודנט בקורסים בהם השתתף:

(בטבלה המקורית רשומות בעלות רקע זהה קובצו וסוכמו לרשומה בעלת אותו רקע בטבלה זו)

STUDENTID	COURSEID	GENDER	YEAROFBIRTH	GRADE
8	12	1	1980	95
9	12.5	-1	1992	89.5

לפניכם קוד המממש את הדוגמא ומחשב טבלת ממוצעים בה מקבצים לפי זהות הסטודנט.

```
public static void main(String[] args) {
    DataTable grades = ... ; // initialization omitted

    // use mean operator
    ListOperator mean = new MeanOperator();

    // group by the first column (student id) and apply
    // "mean" on the rest of the columns.
    DataTable meanTable = grades.groupBy(0, mean);

    // output the result
    System.out.println(meanTable);
}
```

5 נקודות) השלימו את המימוש של הפעולה MeanOperator המחשבת ממוצע:

- 15 נקודות) השלימו את מימוש מבנה הנתונים DataTable כך שהפונקציה main תפעל כנדרש:
1. לנוחיותכם, נתונה פונקציית העזר unique המקבלת מערך של double ומחזירה מערך של double בו כל ערך מהמערך המקורי מופיע פעם אחת בדיוק.
  2. ניתן להגדיר בנאים, שדות, ומתודות נוספות כרצונכם במידה ואתם משתמשים בהם.

```
public class DataTable {
```

```
// the number of rows in the table  
public int getNumberOfRows() {
```

```
}  
// the number of columns in the table  
public int getNumberOfColumns() {
```

```
}
```

```
// the value at the given cell  
public double getValue(int row, int column) {
```

```
}
```

```
public DataTable groupBy(int column, ListOperator operator) {
```

```
}
```

```
// returns a copy of "values" without duplicates  
private static double[] unique(double[] values) {...}
```

```
@Override
```

```
public String toString() {...} // do not implement
```

```
}
```

**סעיף ב' (8 נקודות)**

השלימו את הפונקציה `multipleYears`, ע"י שימוש בפונקציה `groupBy`, המוצאת את רשימת הסטודנטים עבורם נרשמה בטעות יותר משנת לידה אחת בקובץ הנתונים. הפונקציה מחזירה רשימה ובה המספר המזהה (STUDENTID) של כל סטודנט שעבורו נרשמה יותר משנת לידה אחת.

```
private static List<Double> multipleYears(DataTable grades) {
```

```
}
```

**רמז:** יתכן ותרצו להשתמש בשטח למטה כדי להגדיר `ListOperator` מתאים.

**סעיף ג' (7 נקודות)**

פעולת המרה של נתונים מוגדרת ע"י הממשק ConversionOperator:

```
public interface ConversionOperator {
    double convert(double value);
}
```

ממשו את הפעולה convertValues של המחלקה DataTable הממירה את הנתונים בעמודה המבוקשת בעזרת פונקציית המרה נתונה:

```
public class DataTable {
    ... previous code omitted ...
    public void convertValues(int column, ConversionOperator tx) {
```

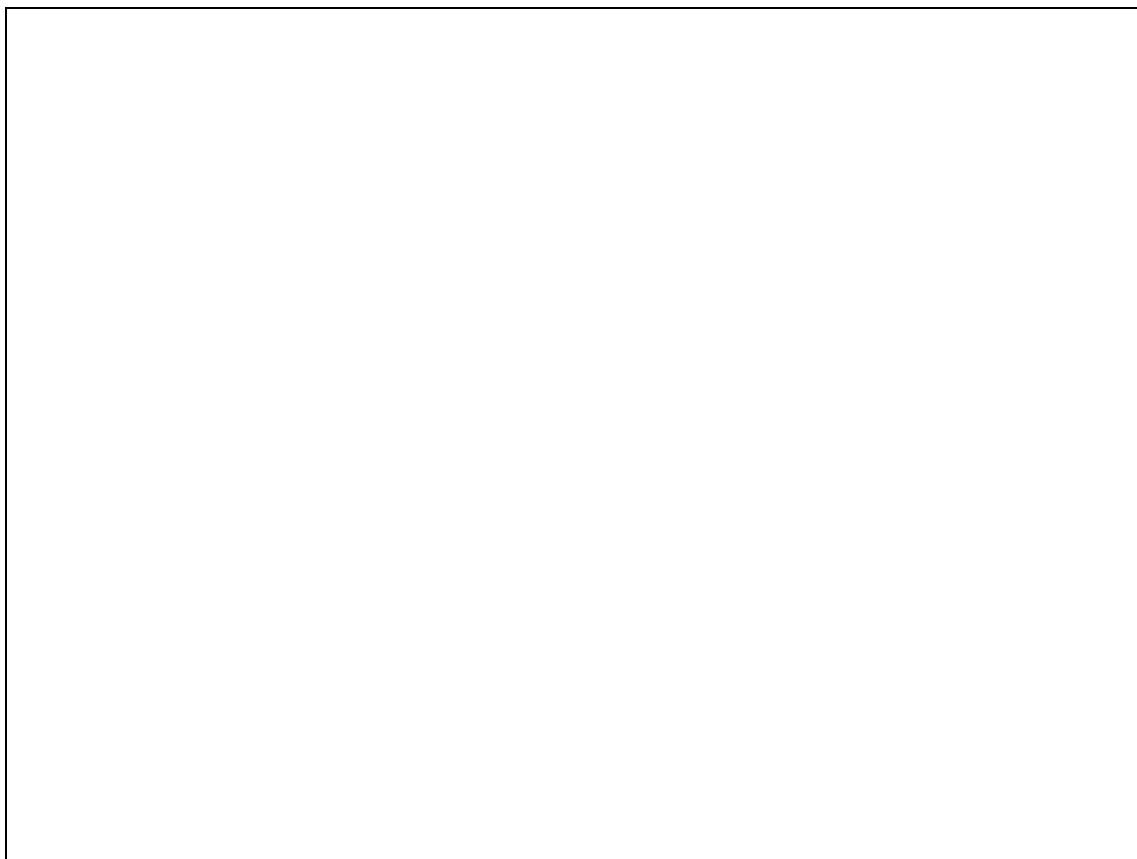
```
    }
}
```

היעזרו בפונקציית convertValues ובפעולה groupBy כדי להשלים את הפונקציית easyCourses המחשבת את אחוז הסטודנטים המצטיינים (שציונם גבוה מ-90) בכל קורס ומדפיסה ל-System.out את מספרי הקורסים שבהם אחוז הסטודנטים המצטיינים גבוה מ-30%.

```
private static void easyCourses(DataTable grades) {
```

```
// easy is a DataTable
for (int r = 0; r < easy.getNumberOfRows(); ++r)
    if (  )
        System.out.println(easy.getValue(r, 1));
}
```

קוד נוסף מחוץ למתודה easyCourses, במידה ונדרש, ניתן לרשום כאן:



## שאלה 2 (35 נקודות)

מספר רציונלי הוא מספר שניתן לבטא כמנה של שני מספרים שלמים, למשל  $5/6$  או  $175/199$ . בשאלה זו אתם מתבקשים לכתוב חלקים מהמחלקה `BigRational` שמאפשרת הצגה וביצוע פעולות על מספרים רציונליים בדיוק בלתי מוגבל (המונה והמכנה יכולים להיות מספרים גדולים כרצוננו).

כדי להקל על המשימה עומדת לרשותנו המחלקה `BigInteger` המיועדת למספרים שלמים כלשהם, ללא מגבלות הטווח הקיימות בטיפוסי המספרים הסטנדרטיים בשפה. מתודות של המחלקות `BigInteger` ו-`String` מופיעות לנוחותכם בנספח.

---

```
import java.math.BigInteger;

public class BigRational implements Comparable<BigRational> {

    public final static BigRational ZERO = new BigRational(0);

    private BigInteger num;    // the numerator
    private BigInteger den;    // the denominator

    // constructor from two standard int's

    public BigRational(int numerator, int denominator) {
        this(new BigInteger(Integer.toString(numerator)),
             new BigInteger(Integer.toString(denominator)));
    }

    // constructor from one standard int, the numerator
    public BigRational(int numerator) {
        this(numerator, 1);
    }
}
```

---

א. (9 נקודות) השלימו את שני הבנאים הבאים:

```
public BigRational(String s)
```

- הבנאי ממחרזת מניח (הנחת קדם) שהמחרוזת `s` מכילה שני מספרים שלמים אי-שליליים, ביניהם סימן חילוק, ואופציונאלית סימן מינוס בהתחלה. אין צורך לטפל במחרוזות קלט שאינן בפורמט הנ"ל. קלטים חוקיים לדוגמה:  
`"1/50000000000"`, `"-13/26"`, `"177/200"`

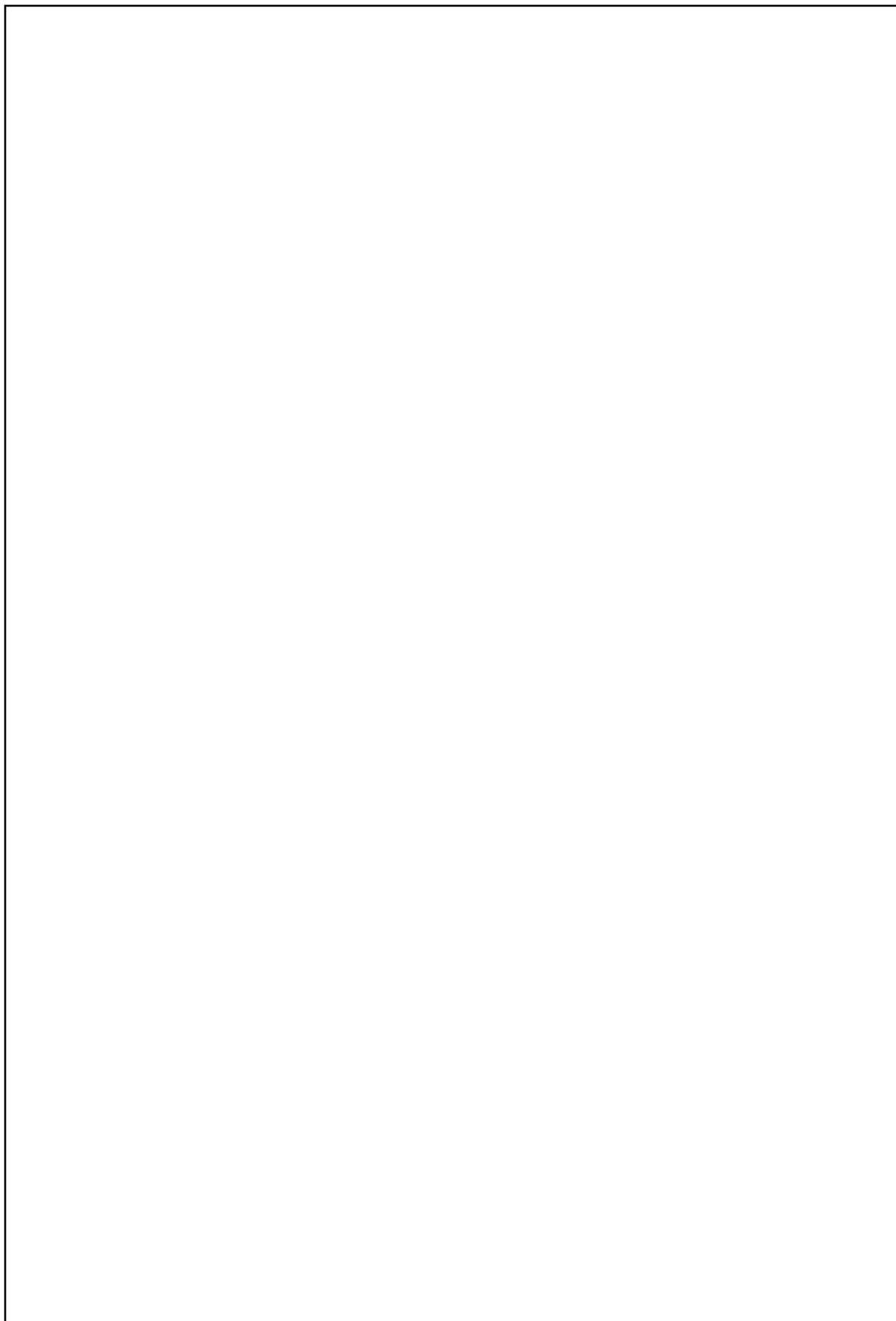
```
public BigRational(BigInteger numerator, BigInteger denominator)
```

- בנאי המקבל שני אובייקטים מסוג `BigInteger`

במימושם של שני הבנאים, בכל מקרה של תקלה, למשל במידה והקלט הוא `null` או מכיל מכנה 0, על הבנאים לזרוק חריג מסוג `RuntimeException` בצירוף הודעה מתאימה.

השתמשו במלבן הבא להשלמת שני הבנאים וכן פונקציות עזר לפי ראות עינכם.





ב. (10 נקודות) ממשו הפעולות הבאות (זיכרו שהמחלקה `BigInteger` זמינה לכם):

```
// return this*b
public BigInteger multiply(BigInteger b)

// return this+b
public BigInteger add(BigInteger b)

// return -this
public BigInteger negate()

// return this-b
public BigInteger subtract(BigInteger b)
```

ג. (8 נקודות) ממשו פונקציית ההשוואה בין שני רציונליים.

**רמז:** עבור שני רציונליים בעלי מכנה חיובי  $a/b$  ו-  $c/d$ , ידוע לנו כי  $a/b > c/d$  אם  $a \cdot d > c \cdot b$ . ניתן להוסיף פונק' עזר במקרה הצורך.

```
public int compareTo(BigRational b)
```

לאחר זמן מה של עבודה עם המחלקה גילינו שכדאי להציג מספרים רציונליים באופן קאנוני, אחיד. כלומר, בצורת הצגה זו המכנה יהיה תמיד חיובי, וכדי להציג רציונלי שלילי המונה יהיה שלילי. כמו כן ההצגה תהיה תמיד מנורמלת (מצומצמת), כלומר המחלק המשותף המקסימלי של המונה והמכנה יהיה 1. למשל  $4/12$  יוצג ע"י מונה 1 ומכנה 3.

ד. (8 נקודות) כתבו את פונקציית העזר לחישוב המחלק המשותף המקסימלי (ממ"מ בקיצור או gcd בלעז) של שני אובייקטי BigInteger (שימו לב שטיפוסי הקלט והפלט של פונקציית עזר זו הם BigInteger).

ניתן לממש את האלגוריתם של אוקלידס: בהינתן שני מספרים טבעיים (שלמים חיוביים) חלק את המספר הגדול שנסמנו ב-  $a$  במספר הקטן שנסמנו ב-  $b$ . נסמן את שארית החלוקה ב-  $r$ . אם  $r=0$  אז הממ"מ הוא  $b$ , אחרת הממ"מ שווה לממ"מ של  $b$  ו-  $r$ .

בסעיף זה לא ניתן להשתמש בפונקציית gcd קיימת.

```
public static BigInteger gcd(BigInteger a, BigInteger b)
```

## שאלה 3 (15 נקודות)

נתונות המחלקות Circle ו-Counter:

```

1  public class Circle{
2      int radius;
3      public Circle (int r){
4          this.radius = r;
5      }
6      public void setRadius(int r){
7          this.radius = r;
9      }
}

10 public class Counter {
11     public static void main(String[] str){
12         Integer obj1 = new Integer(1);
13         Integer obj2 = new Integer(2);
14         Integer obj3 = new Integer(1);
15         List<Integer> objects = new ArrayList<Integer>();
16         objects.add(obj1);
17         objects.add(obj2);
18         objects.add(obj3);
19         int numInCollection = countNumOfObjects(objects, 1);
20         System.out.println(numInCollection);
21     }
22
23     public static int countNumOfObjects(List<?> objectsList, Object query){
24         int cnt = 0;
25         for (Object object : objectsList){
26             if (object.equals(query)){
27                 cnt++;
28             }
30         }
31         return cnt;
32     }
}

```

בכל אחד מהסעיפים הבאים נבצע שינוי בקוד. עליכם לציין את הפלט של הרצת המתודה main של המחלקה Counter. אם לדעתכם אין פלט לתוכנית בשל בעיית קומפילציה או שגיאה בזמן ריצה (זריקת חריג) – הסבירו מה השגיאה וציינו את שורת הקוד הבעייתית. פתרון ללא הסבר לא יזכה בנקודות. בכל סעיף, השינויים מתייחסים לגירסא המקורית של המחלקות Circle ו-Counter.

**סעיף א:**  
אין שינוי בקוד.

תשובה:

**סעיף ב:**

```
12  int obj1 = 1;
13  int obj2 = 2;
14  int obj3 = 1;
15  List<int> objects = new ArrayList<int>();
```

תשובה:

**סעיף ג:**

```
19  int numInCollection = countNumOfObjects(objects, obj1);
26  if (object == query){
```

תשובה:

**סעיף ד:**

```
12  Circle obj1 = new Circle(1);
13  Circle obj2 = new Circle(2);
14  Circle obj3 = new Circle(1);
15  List<Circle> objects = new ArrayList<Circle>();

19  int numInCollection = countNumOfObjects(objects, new Circle(1));
```

תשובה:

**סעיף ה:**

```
12  Circle obj1 = new Circle(1);
13  Circle obj2 = new Circle(2);
14  Circle obj3 = obj1; obj3.setRadius(5);
15  List<Circle> objects = new ArrayList<Circle>();

19  int numInCollection = countNumOfObjects(objects, obj1);
```

תשובה:

## שאלה 4 (15 נקודות)

קראו את קוד המחלקות הבאות וענו על השאלות:

```

public abstract class A {

    public A() {
        System.out.print("A() ");
    }

    protected abstract A foo(A a);

    public A bar() {
        System.out.print("A.bar() ");
        return null;
    }
}

public class B extends A {

    public B() {
        System.out.print("B() ");
    }

    /* MISSING CODE!!! */

    public static void main(String[] args) {
        B b = new B();
        b.foo(null);
    }
}

```

בכל אחד מהסעיפים הבאים הוחלף הקוד החסר במחלקה B בפונקציה foo מסוימת. הנכם מתבקשים לציין מהו הפלט של הרצת פונקציית ה-main בכל אחד מהמקרים. אם לדעתכם אין פלט לתכנית מכיוון שאינה עוברת קומפילציה או מכיוון שקיימת שגיאה בזמן ריצה (זריקת חריג), הסבירו מה השגיאה. בכל מקרה (שגיאה או ריצה תקינה) יש לכתוב הסבר קצר.

סעיף א' (3 נק')

```

public A foo(A a) {
    System.out.print("foo1 ");
    return null;
}

```

סעיף ב' (3 נק')

```
protected A foo(A a) {  
    System.out.print("foo2 ");  
    return a.bar();  
}
```

סעיף ג' (3 נק')

```
protected A foo(B b) {  
    System.out.print("foo3 ");  
    return new B();  
}
```

סעיף ד' (3 נק')

```
protected A foo(A a) {  
    System.out.print("foo4 ");  
    return new A();  
}
```

סעיף ה' (3 נק')

```
protected A foo(A a) {  
    System.out.print("foo5 ");  
    return bar();  
}
```



# נספח

java.math

## Class BigInteger

Field Summary	
static <a href="#">BigInteger</a>	<a href="#">ONE</a> The BigInteger constant one.
static <a href="#">BigInteger</a>	<a href="#">ZERO</a> The BigInteger constant zero.

Constructor Summary	
<a href="#">BigInteger</a>	<a href="#">BigInteger</a> (String val) Translates the decimal String representation of a BigInteger into a BigInteger.

Method Summary	
<a href="#">BigInteger</a>	<a href="#">abs</a> () Returns a BigInteger whose value is the absolute value of this BigInteger.
<a href="#">BigInteger</a>	<a href="#">add</a> ( <a href="#">BigInteger</a> val) Returns a BigInteger whose value is (this + val).
int	<a href="#">compareTo</a> ( <a href="#">BigInteger</a> val) Compares this BigInteger with the specified BigInteger.
<a href="#">BigInteger</a>	<a href="#">divide</a> ( <a href="#">BigInteger</a> val) Returns a BigInteger whose value is (this / val).
boolean	<a href="#">equals</a> (Object x) Compares this BigInteger with the specified Object for equality.
int	<a href="#">hashCode</a> () Returns the hash code for this BigInteger.
<a href="#">BigInteger</a>	<a href="#">mod</a> ( <a href="#">BigInteger</a> m) Returns a BigInteger whose value is (this mod m). This method differs from <code>remainder</code> in that it always returns a <i>non-negative</i> BigInteger.
<a href="#">BigInteger</a>	<a href="#">multiply</a> ( <a href="#">BigInteger</a> val) Returns a BigInteger whose value is (this * val).
<a href="#">BigInteger</a>	<a href="#">negate</a> () Returns a BigInteger whose value is (-this).
<a href="#">BigInteger</a>	<a href="#">remainder</a> ( <a href="#">BigInteger</a> val) Returns a BigInteger whose value is (this % val).
<a href="#">String</a>	<a href="#">toString</a> () Returns the decimal String representation of this BigInteger.

java.lang

## Class String

### Constructor Summary

**String**([String](#) original)

Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

### Method Summary

char	<a href="#">charAt</a> (int index) Returns the <code>char</code> value at the specified index.
int	<a href="#">compareTo</a> ( <a href="#">String</a> anotherString) Compares two strings lexicographically.
boolean	<a href="#">contains</a> ( <a href="#">CharSequence</a> s) Returns true if and only if this string contains the specified sequence of char values.
boolean	<a href="#">endsWith</a> ( <a href="#">String</a> suffix) Tests if this string ends with the specified suffix.
boolean	<a href="#">equals</a> ( <a href="#">Object</a> anObject) Compares this string to the specified object.
int	<a href="#">indexOf</a> (int ch) Returns the index within this string of the first occurrence of the specified character.
int	<a href="#">indexOf</a> ( <a href="#">String</a> str) Returns the index within this string of the first occurrence of the specified substring.
int	<a href="#">lastIndexOf</a> ( <a href="#">String</a> str) Returns the index within this string of the rightmost occurrence of the specified substring.
int	<a href="#">length</a> () Returns the length of this string.
<a href="#">String</a> []	<a href="#">split</a> ( <a href="#">String</a> regex) Splits this string around matches of the given <a href="#">regular expression</a> .
boolean	<a href="#">startsWith</a> ( <a href="#">String</a> prefix) Tests if this string starts with the specified prefix.
<a href="#">String</a>	<a href="#">substring</a> (int beginIndex, int endIndex) Returns a new string that is a substring of this string.