מבוא למדעי המחשב - סמסטר א' תש"פ
עבודת בית מספר 3

צוות התרגיל: עמיר רובין, עידן אטיאס ובר סימן טוב.
תאריך פרסום: 6.12.19
תאריך הגשה: 20.12.19 בשעה 12:00 בצהריים.

הקדמה

בעבודת בית זו נכיר את המחלקה BigInteger ואת המחלקה Random, נתרגל רקורסיה ותכנות מונחה עצמים.

בעבודה זו 16 משימות וסך הנקודות המקסימלי הוא 100. הניקוד לכל משימה מפורט במסמך.
בעבודה זו **מותר להשתמש בידע שנלמד עד הרצאה** 12**(כולל), וכן עד תרגול** 7**(כולל).**

הוראות מקדימות

הערות כלליות

1. **קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה.** ודאו שאתם מבינים את כל המשימות.
2. עבודה זו תוגש ביחידים. על מנת להגיש את העבודה יש להירשם למערכת ההגשות (Submission System). את הרישום למערכת ההגשות מומלץ לבצע כבר עכשיו, טרם הגשת העבודה (קחו בחשבון כי הגשה באיחור אינה מתקבלת). את הגשת העבודה ניתן לבצע רק לאחר הרישום למערכת.
3. בכל משימה מורכבת יש לשקול כיצד לחלק את המשימה לתתי-משימות ולהגדיר פונקציות עזר בהתאם.
4. בכל הסעיפים אפשר ומומלץ להשתמש בפונקציות מסעיפים קודמים.

קבצים

5. ישנם ארבעה קבצים מצורפים לתרגיל זה:
Part1.java, Part2.java, Bit.java, NumberAsBits.java
בקבצים אלו תערכו שינויים בהתאם למפורט בתרגיל. עליכם להגישם כפתרון, מקובצים כקובץ ZIP יחיד. שימו לב: עליכם להגיש רק את קובצי ה־Java. אין לשנות את שם הקבצים, ואין להגיש קבצים נוספים, בקובץ ה-ZIP אסור שתהיה תיקיה, אלא הקבצים בלבד. שם קובץ ה־ZIP יכול להיות כרצונכם, אך באנגלית בלבד. נוסף על כך, הקובץ שתגישו יכול להכיל טקסט המורכב מאותיות באנגלית, מספרים וסימני פיסוק בלבד. טקסט אשר יכיל תווים אחרים (אותיות בעברית, יוונית וכדומה) לא יתקבל. הקפידו לא להשאיר בהגשה חלקי קוד שאינם חלק מהתכנית (לדוגמה, בדיקות שכתבתם עבור עצמכם).
6. **קבצים שיוגשו שלא על פי הנחיות אלו לא ייבדקו.** את קובץ ה־ZIP יש להגיש ב־ Submission System. פרטים לגבי ההרשמה ואיך להגיש את העבודה תוכלו למצוא באתר.
7. בחלק מהקבצים נתונה לכם פונקצית main המכילה טסטים (בהערה) לשימושכם.

בדיקת עבודות הבית

8. עבודות הבית נבדקות גם באופן ידני וגם באופן אוטומטי. הבדיקה האוטומטית מתייחסת אך ורק לפלט המוחזר או מודפס מהפונקציות.
9. שימו לב במשימות בהם אתם קוראים לפונקציה שכתבתם במשימה אחרת (למשל כאשר הפונקציה f קוראת לפונקציה g): טעות בפונקציה הנקראת (g) תגרור טעות גם בפונקציה הקוראת (f). משמעות הדבר הוא שבמקרה כזה תהיה הפחתה בציון עבור פתרון הפונקציה f.
10. סגנון כתיבת הקוד ייבדק באופן ידני. יש להקפיד על כתיבת קוד יעיל וברור, על מתן שמות משמעותיים למשתנים, על הזחות (אינדנטציה), **ועל הוספת הערות בקוד** המסבירות את תפקידם של

מקטעי הקוד השונים. אין צורך למלא את הקוד בהערות סתמיות, אך חשוב לכתוב הערות בנקודות קריטיות המסבירות קטעים חשובים בקוד. הערות יש לרשום אך ורק באנגלית. יש לתכנן את הקוד בצורה נכונה כך שמשימות מורכבות יחולקו לתתי משימות המבוצעות על ידי פונקציות עזר. כתיבת קוד שאינה עומדת בדרישות אלו תגרור הפחתה בציון העבודה.

עזרה והנחיה

11. לכל עבודת בית בקורס יש צוות שאחראי לה. ניתן לפנות לצוות בשעות הקבלה. פירוט שמות האחראים לעבודה מופיע במסמך זה וכן באתר הקורס, כמו גם פירוט שעות הקבלה. כמו כן, אתם יכולים להיעזר בפורום ולפנות בשאלות לחבריכם לכיתה. צוות הקורס עובר על השאלות ונותן מענה במקרה הצורך.
12. **בתגבור** של השבוע (08.12 עד 11.12) נפתור באופן מודרך את משימות 1.1, 2.1, 3.1, 3.2.
13. בכל בעיה **אישית** הקשורה בעבודה (מילואים, אשפוז וכו'), אנא פנו אלינו דרך מערכת הפניות, כפי שמוסבר באתר הקורס.
14. אנחנו ממליצים בחום להעלות פתרון למערכת ההגשה לאחר כל סעיף שפתרתם. הבדיקה תתבצע על הגרסה האחרונה שהועלתה (בלבד!).

יושר אקדמי

הימנעו מהעתקות! ההגשה היא ביחידים. אם מוגשות שתי עבודות עם קוד זהה או אפילו דומה - זוהי העתקה, אשר תדווח לאלתר לוועדת משמעת. אם טרם עיינתם בסילבוס הקורס אנא עשו זאת כעת.

מומלץ לקרוא היטב את כל ההוראות המקדימות ורק לאחר מכן להתחיל בפתרון המשימות. **ודאו שאתם יודעים לפתוח קבוצת הגשה (עבור עצמכם) במערכת ההגשות.**

משימה 0: הצהרה (0 נקודות)

פתחו כל אחד מארבעת קבצי ה-java:
Part1.java, Part2.java, Bit.java, NumberAsBits.java
וכיתבו בראשם את שמכם ואת מספר תעודת הזהות שלכם.
משמעות פעולה זו היא שאתם מסכימים על הכתוב בו. דוגמה:

I, *Israel Israeli (123456789)*, assert that the work I submitted is entirely my own.
I have not received any part from any other person, nor did I give parts of it for use to others.
I realize that if my work is found to contain code that is not originally my own, a formal complaint will be opened against me with the BGU disciplinary committee.

_____

חלק 1: שימוש במחלקה BigInteger

מבוא

כחלק מהעבודה עם JAVA בפרט ובעולם פיתוח התוכנה בכלל, ישנם כלים רבים שפותחו על ידי מפתחי השפה או קהילת המפתחים.

בחלק זה נכיר שתי מחלקות שהן חלק מ-JAVA:

1. המחלקה BigInteger (תיעוד רשמי כאן).
2. המחלקה Random (תיעוד רשמי כאן).

עליכם לקרוא על המחלקות בתיעוד הרשמי המפורסם על ידי JAVA, בקישורים למעלה, להכיר את הפונקציות שלהן, ולהשתמש בהן.

(התיעוד הרשמי הוא חלק מה-API של JAVA (Application Programming Interface))

סעיף 1.1 : סכום הערכים הקטנים מ-n (5 נקודות)

השלימו בקובץ Part1.java את הפונקציה:

```
public static BigInteger sumSmaller(BigInteger n)
```

המקבלת משתנה n מטיפוס BigInteger ומחזירה משתנה מטיפוס BigInteger שערכו סכום המספרים החיוביים הקטנים מ-n.

הניחו כי הקלט אינו null. פונקציה זו לא זורקת חריגות.

עליכם לבצע חישובים (כגון סכימה ובדיקת שיוון) באמצעות הפונקציות של המחלקה BigInteger , כמתואר בAPI.

לדוגמא, קטע הקוד הבא:

```
public static void main(String[] args) {
 BigInteger biMinus = new BigInteger("-10");
 System.out.println(sumSmaller(biMinus)); // 0

 BigInteger bi0 = new BigInteger("0");
 System.out.println(sumSmaller(bi0)); // 0

 BigInteger bi7 = new BigInteger("7");
 System.out.println(sumSmaller(bi7)); // 21

 BigInteger biHigh = new BigInteger("99999");
 System.out.println(sumSmaller(biHigh)); // 4999850001
}
```

ידפיס:

```
0
0
21
4999850001
```

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

סעיף 1.2 : עבודה עם המחלקה Random (5 נקודות)

המחלקה Random מאפשרת לייצר משתנה (מטיפוס Random) אשר באמצעותו יכול המשתמש לבקש מספרים שנדגמים בצורה פסאודו-אקראית. כלומר, אוביקט מטיפוס Random הוא מחולל (Generator) של מספרים אקראיים.

השלימו בקובץ Part1.java את הפונקציה:

```
public static void printRandoms(int n)
```

המקבלת מספר n ומדפיסה למסך n מספרים מטיפוס int שנדגמו בצורה פסאודו-אקראית ואחידה מכל טווח הערכים האפשרים של int. על כל ערך להיות מודפס בשורה נפרדת.

הנחיה חובה: קראו על הפונקציה nextInt() בתיעוד הרשמי של Random. עליכם לייצר משתנה מטיפוס Random לביצוע המשימה.

הניחו כי n אינו שלילי, כלומר n>=0. פונקציה זו לא זורקת חריגות.

לדוגמאי, פלט אפשרי של הקריאה:

```
printRandoms(5);
```

הוא:

-269001551
1230791088
1983672709
-205636269
222626083

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

סעיף 1.3 : בדיקת ראשוניות של BigInteger (5 נקודות)

השלימו בקובץ Part1.java את הפונקציה:

```
public static boolean isPrime(BigInteger n)
```

המקבלת משתנה מסוג BigInteger שערכו אינו שלילי ומחזירה ערך בוליאני המשקף האם הקלט הוא ראשוני או לא, באופן ודאי לחלוטין, כלומר ללא שום סיכוי לשגיאה (שימו לב כי 0 ו-1 אינם ראשוניים).

הניחו כי הקלט תקין (כלומר הקלט אינו null וערכו אינו שלילי). פונקציה זו לא זורקת חריגות.

עליכם לפתור שאלה זו על בסיס האלגוריתם הנאיבי שהוצג בכיתה לבדיקת ראשוניות.

עליכם לבצע חישובים (כגון מודולו) באמצעות הפונקציות של המחלקה BigInteger , כמתואר ב-API.

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

סעיף 1.4 : הגרלת ראשוני קטן מ-2^n (5 נקודות)

השלימו בקובץ Part1.java את הפונקציה:

```
public static BigInteger randomPrime(int n)
```

המקבלת מספר n>1 שלם מסוג int, ומחזירה מספר ראשוני, **שנבחר באופן אקראי**, הקטן מ-2^n.

שימו לב- עליכם להחזיר מספר שהוא באופן ודאי לחלוטין ראשוני (בפרט המספר גדול ממש מ-1.)

הניחו כי הקלט תקין, כלומר ש n>1. פונקציה זו לא זורקת חריגות.

הנחיות:

1. אתם רשאים להשתמש בפונקציה מהסעיף הקודם.
2. שימו לב לבנאי של המחלקה BigInteger שחתימתו BigInteger(int numBits, Random rnd)

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

חלק 2: רקורסיה – בעיית העודף

שימו לב: אין קשר בין הסעיפים בחלק זה.

מבוא

בשאלה זו נעסוק בבעיית העודף. בהינתן רשימת ערכי מטבעות, וערך שאותו רוצים לפרוט, הבעיה עוסקת בפריטת הערך הנתון לפי ערכי המטבעות האפשרים.

סעיף 2.1 : בעית העודף (10 נקודות)

השלימו בקובץ Part2.java את הפונקציה:

```
public static boolean change(int[] coins, int n)
```

המקבלת מערך ממויין של מספרים חיוביים שונים זה מזה בשם coins, וערך שאינו שלילי n.

הפונקציה מחזירה ערך true אם ניתן לפרוט את n באמצעות מטבעות שערכיהן נמצאים במערך coins, כאשר מותר להשתמש בכמות בלתי מוגבלת של מטבעות, אחרת הפונקציה מחזירה ערך false.

הנחיה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הניחו שהקלט תקין, כלומר שהמערך הנתון אינו null, הוא מכיל אך ורק מספרים חיוביים שונים זה מזה, והוא ממוין. כמו כן n>=0. פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון שאינו רקורסיבי לא יקבל ניקוד.

דוגמאות:

בעבור הקלט

Coins = {1,5,10}

n = 7

הפונקציה תחזיר את הערך true , כיוון שניתן לפרוט את 7 על פי הנוסחא: $1 + 1 + 5 = 7$

בעבור הקלט

Coins = {2,10,20,100}

n = 15

הפונקציה תחזיר את הערך false , כיוון שלא ניתן לפרוט את 15 באמצעות מטבעות שערכיהן נמצאים במערך coins.

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

<u>סעיף 2.2</u>: בעיית העודף עם הגבלת מטבעות – האם קיים פתרון? (10 נקודות)

השלימו בקובץ Part2.java את הפונקציה:

```
public static boolean changeLimited(int[] coins, int n, int numOfCoinsToUse)
```

המקבלת מערך ממויין של מספרים חיוביים שונים זה מזה בשם coins, ערך שאינו שלילי n, וערך שאינו שלילי numOfCoinsToUse.

הפונקציה מחזירה ערך true אם ניתן לפרוט את n באמצעות **בדיוק** numOfCoinsToUse מטבעות שערכיהן נמצאים במערך coins, אחרת הפונקציה מחזירה ערך false.

הנחיה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הניחו שהקלט תקין (כלומר שהמערך הנתון אינו null, הוא מכיל אך ורק מספרים חיוביים שונים זה מזה, והוא ממוין. כמו כן n>=0 וגם numOfCoinsToUse>=0). פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון שאינו רקורסיבי לא יקבל ניקוד.

דוגמאות:

בעבור הקלט

Coins ={1,7,19,12}

N = 20

numOfCoinsToUse=2

הפונקציה תחזיר את הערך true , כיוון ש $19 + 1 = 20$ וגם השתמשנו בשני מטבעות בלבד.

בעבור הקלט

Coins ={5,7,12}

N = 8

numOfCoinsToUse=2

הפונקציה תחזיר את הערך false , כיוון שכלל לא ניתן להגיע לסכום 8 בעזרת המטבעות המופיעים במערך.

בעבור הקלט

Coins ={1,7,10,12}

N = 10

numOfCoinsToUse=5

הפונקציה תחזיר את הערך false , כיוון שלא קיים צירוף של 5 מטבעות **בדיוק** כך שסכומם יהיה 10.

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

<u>סעיף 2.3</u>: בעיית העודף עם הגבלת מטבעות – הדפסת פתרון יחיד (10 נקודות)

השלימו בקובץ Part2.java את הפונקציה:

```
public static void printChangeLimited(int[] coins, int n, int numOfCoinsToUse)
```

המקבלת מערך ממויין של מספרים חיוביים שונים זה מזה בשם coins, ערך שאינו שלילי n, וערך שאינו שלילי numOfCoinsToUse.

אם ניתן לפרוט את n באמצעות בדיוק numOfCoinsToUse מטבעות שערכיהן נמצאים במערך coins, הפונקציה מדפיסה למסך <u>הצעה **יחידה**</u>(כלשהי) לפריטה **ממוינת** לפי ערכי המטבעות, אחרת הפונקציה אינה מדפיסה דבר.

עליכם להדפיס את הערכים מופרדים על ידי פסיק, ללא רווחים, ראו דוגמא.

הנחיה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הניחו שהקלט תקין (כלומר שהמערך הנתון אינו null, הוא מכיל אך ורק מספרים חיוביים שונים זה מזה, והוא ממוין. כמו כן n>=0 וגם numOfCoinsToUse>=0). פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון שאינו רקורסיבי לא יקבל ניקוד.

דוגמאות:

עבור הקלט:

Coins = {1,2,3}

n=4

numOfCoinsToUse=2

יודפס:

"2,2"

או:

"1,3"

שימו לב שהמחרוזת ממוינת לפי ערכי המטבעות.

שימו לב שאין פסיק בסוף המחרוזת.

שימו לב שאין רווחים.

בעבור הקלט

Coins ={1,7,12}

N = 10

numOfCoinsToUse=5

הפונקציה לא תדפיס דבר.

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

<u>סעיף 2.4</u>: בעיית העודף עם הגבלת מטבעות – מציאת מספר האפשרויות (10 נקודות)

השלימו בקובץ Part2.java את הפונקציה:

```
public static int countChangeLimited(int[] coins, int n, int numOfCoinsToUse)
```

המקבלת מערך ממויין של מספרים חיוביים שונים זה מזה בשם coins, ערך שאינו שלילי n, וערך שאינו שלילי numOfCoinsToUse.

הפונקציה מחזירה את מספר האפשרויות השונות לפרוט את n באמצעות בדיוק numOfCoinsToUse מטבעות שערכיהן נמצאים במערך coins. אם לא ניתן לעשות זאת יוחזר הערך 0.

שימו לב – בספירת הפתרונות אין חשיבות לסדר המטבעות. ראו דוגמאות.

הנחיה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הניחו שהקלט תקין (כלומר שהמערך הנתון אינו null, הוא מכיל אך ורק מספרים חיוביים שונים זה מזה, והוא ממוין. כמו כן n>=0 וגם numOfCoinsToUse>=0). פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון שאינו רקורסיבי לא יקבל ניקוד.

דוגמאות:

עבור הקלט:

Coins = {1,2,3}

n=4

numOfCoinsToUse=2

יוחזר הערך 2, מכיוון שישנן שתי דרכים לפרוט 4 באמצעות 2 מטבעות שערכיהן הן 1,2 או 3:

דרך ראשונה: 2,2

דרך שניה: 1,3

(שימו לב: האפשרות להחזיר 3,1 שקולה לאפשרות להחזיר 1,3, ולכן אינה נספרת כדרך נוספת.)

עבור הקלט:

Coins = {5,10,20,50,100}

n=100

numOfCoinsToUse=5

יוחזר הערך 3 מכיוון שישנן שלוש דרכים לפרוט 100 באמצעות 5 מטבעות שערכיהן במערך הנתון:

דרך ראשונה: 20,20,20,20,20

דרך שניה: 10,10,10,20,50

דרך שלישית: 5,5,20,20,50

עבור הקלט:

Coins = {5,10,50}

n=65

numOfCoinsToUse=2

יוחזר הערך 0 מכיוון שלא קיימות דרכים לפרוט 65 באמצעות בדיוק 2 מטבעות שערכיהן במערך הנתון

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

<u>סעיף 2.5</u>: בעיית העודף עם הגבלת מטבעות – הדפסת כל האפשרויות (10 נקודות)

השלימו בקובץ Part2.java את הפונקציה:

```
public static void printAllChangeLimited(int[] coins, int n, int numOfCoinsToUse)
```

המקבלת מערך ממויין של מספרים חיוביים שונים זה מזה בשם coins, ערך שאינו שלילי n, וערך שאינו שלילי numOfCoinsToUse.

הפונקציה מדפיסה למסך את כל האפשרויות השונות לפרוט את n באמצעות בדיוק numOfCoinsToUse מטבעות שערכיהן נמצאים במערך coins.

כל אפשרות לפריטה מודפסת **ממוינת** לפי ערכי המטבעות, בשורה נפרדת. ערכי המטבעות מופרדים על ידי פסיק, ללא רווחים, ראו דוגמא.

הנחיה חובה: הגדירו פונקציה רקורסיבית משלכם, אשר לה תקראו מתוך הפונקציה הנתונה לעיל.

הניחו שהקלט תקין (כלומר שהמערך הנתון אינו null, הוא מכיל אך ורק מספרים חיוביים שונים זה מזה, והוא ממוין. כמו כן n>=0 וגם numOfCoinsToUse>=0). פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון שאינו רקורסיבי לא יקבל ניקוד.

דוגמאות:

עבור הקלט:

Coins = {1,2,3}

n=4

numOfCoinsToUse=2

יודפסו למסך שתי השורות הבאות:

"2,2"

"1,3"

או בסדר הפוך:

"1,3"

"2,2"

שימו לב שאין פסיק בסוף המחרוזות.

עבור הקלט:

Coins = {1,5,10,20}

n=13

numOfCoinsToUse=2

לא יודפס דבר

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

<u>חלק 2.6:</u> בעיית העודף הקובני (10 נקודות)

במדינה קובה ישנן שתי מערכות של מטבעות: CUP ו-CUC. (תוכלו לקרוא עליהן כאן)

בשאלה זו נניח כי בקובה שטרות בערכים 1,3,5,10,20,50,100 CUC, ואותם ערכים של CUP: 1,3,5,10,20,50,100. בעבודה זו נניח כי שער ההמרה הוא **3CUP = 1CUC** .

השלימו בקובץ Part2.java את הפונקציה:

```
public static int changeInCuba(int n)
```

המקבלת מספר שלם וחיובי n, המייצג ערך **ב-CUC**, ומחזירה int, המייצג את **מספר האפשרויות** להחזיר עודף בקובה בשווי n, ללא חשיבות לסדר המטבעות. שימו לב: בהחזרת העודף ניתן להשתמש הן ב CUP והן ב-CUC, ראו דוגמאות למטה.

חשבו כיצד להשתמש בפונקצית עזר או בסעיפים קודמים לפתרון הבעיה.

הניחו שהקלט תקין (כלומר n>0). פונקציה זו לא זורקת חריגות.

עליכם לפתור את השאלה בצורה רקורסיבית. פתרון שאינו רקורסיבי לא יקבל ניקוד.

דוגמאות:

- עבור הקלט 1, נקבל את הערך 3, כיוון שישנן 3 אפשרויות להחזיר CUC 1:

```
1. 1CUP,1CUP,1CUP
2. 3CUP
3. 1CUC
```

- עבור הקלט 2, נקבל את הערך 7, כיוון שישנן 7 אפשרויות להחזיר CUC 2:

```
1. 1CUP,1CUP,1CUP,1CUP,1CUP,1CUP
2. 1CUP,1CUP,1CUP,3CUP
3. 1CUP,1CUP,1CUP,1CUC
4. 1CUP,5CUP
5. 3CUP,3CUP
6. 3CUP,1CUC
7. 1CUC,1CUC
```

- עבור הקלט 20, נקבל את הערך 8689
- עבור הקלט 50, נקבל את הערך 1655256.

מסקנה: אין כמו בארץ.

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

חלק 3: המחלקות Bit ו NumberAsBits

מבוא

בתכנות מונחה עצמים (Object Oriented Programming) אנחנו מייצגים רעיונות באמצעות טיפוסי משתנים.
בחלק זה של העבודה נגדיר את הטיפוסים Bit וNumberAsBits.
Bit מייצג את הרעיון של ספרה בינארית, שהערך שלה אחד או אפס.
NumberAsBits מייצג את הרעיון של מספר חיובי, ללא הגבלה על גודלו (בשונה מint ובדומה לBigInteger).
לשם כך ניצור מחלקות חדשות.

המחלקה Bit (6 נקודות)

למחלקה Bit יהיה שדה פרטי אחד, value, מהטיפוס boolean . הספרה 0 תיוצג על ידי עצם שהערך בשדה שלו הוא false. הספרה 1 תיוצג על ידי עצם שהערך בשדה שלו הוא true.

במשימות להלן תכתבו בנאי ופונקציות פשוטות של המחלקה Bit. אתם מקבלים קובץ בשם Bit.java שבו תבניות שאותן עליכם למלא.

```
public class Bit {
    private boolean value;
    public Bit(boolean value) { /* 3.1 הקוד הזה יושלם בסעיף */}
    public int toInt() { /* 3.2 הקוד הזה יושלם בסעיף */}
    public String toString() { /* 3.3 הקוד הזה יושלם בסעיף */}
}
```

3.1. השלימו את הקוד של הבנאי כך שאם הוא מקבל כארגומנט את הערך false , הוא יוצר עצם המייצג את הספרה 0. אחרת, הוא יוצר עצם המייצג את הספרה 1. פונקציה זו לא זורקת חריגות.

3.2. השלימו את הקוד של השיטה toInt() כך שהיא תחזיר את הערך 1 אם העצם מייצג את הספרה 1. אחרת יוחזר הערך 0. פונקציה זו לא זורקת חריגות.

3.3. השלימו את הקוד של השיטה toString() כך שאם העצם מייצג את הספרה 0, השיטה תחזיר את המחרוזת "0", אחרת, היא תחזיר את המחרוזת "1". פונקציה זו לא זורקת חריגות.

לדוגמה, התוכנית

```
public static void main(String[] args) {
    Bit bit1 = new Bit(true);
    Bit bit0 = new Bit(false);
    System.out.println(bit0.toString()+" "+bit1.toString());
    Int sum = bit1.toInt() + bit0.toInt();
    System.out.println(sum);

}
```

תדפיס

```
0 1
1
```

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.

המחלקה NumberAsBits (14 נקודות)

למחלקה NumberAsBits יהיה שדה פרטי אחד, bits, מטיפוס מערך של Bit, שהוגדר למעלה.

כל מספר (חיובי) יהיה מיוצג על ידי המערך bits בייצוגו הבינארי. לדוגמא:

היצוג הבינארי של המספר 3 הוא 11, לכן היצוג ב bits יהיה בעזרת מערך באורך שתיים ששני ערכיו הם Bit המייצג 1, כלומר { new Bit(true), new Bit(true)}.

היצוג הבינארי של המספר 8 הוא 1000, לכן היצוג ב bits יהיה בעזרת מערך באורך ארבע שבתא הראשון ערכו הוא Bit המייצג 1, וכל השאר מייצגים 0, כלומר

{ new Bit(true) , new Bit(false) , new Bit(false) , new Bit(false)}

במשימות להלן תכתבו בנאי, ופונקציות פשוטות של המחלקה NumberAsBits. אתם מקבלים קובץ בשם NumberAsBits.java שבו תבניות שאותן עליכם למלא.

```
public class NumberAsBits {
    private Bit[] bits;
    public NumberAsBits (Bit[] bits) { /* 3.4 הקוד הזה יושלם בסעיף */}
    public String toString() { /* 3.5 הקוד הזה יושלם בסעיף */}
    public String base2() { /* 3.6 הקוד הזה יושלם בסעיף */}
}
```

3.4. השלימו את הקוד של הבנאי. אין להניח דבר על הקלט. על הבנאי לאתחל את הערכים במערך bits להיות זהים לערכים במערך הנתון. אסור שהמערך, שההפניה אליו התקבלה בפרמטר של הבנאי ,יהיה גם השדה של העצם. כלומר שינוי של המערך הנתון לא משפיע על ערכי השדה. ראו דוגמא למטה. שימו לב – הערכים בתוך המערך הם מטיפוס Bit. חשבו האם יש צורך לייצר העתקים שלהם בעת הגדרת השדה bits בבנאי של NumberAsBits.

3.5. השלימו את הקוד של השיטה toString() כך שתוחזר מחרוזת המכילה את היצוג **העשרוני** של המספר. פונקציה זו לא זורקת חריגות.

3.6. השלימו את הקוד של השיטה base2() כך שתוחזר מחרוזת המכילה את היצוג **הבינארי** של המספר. פונקציה זו לא זורקת חריגות.

אין להניח דבר על הקלט. קריאה לבנאי עם מערך null או עם מערך שאחד מתאיו null תגרור שגיאת IllegalArgumentExcption, הודעת השגיאה נתונה לשיקול דעתכם.

לדוגמה, התוכנית

```
public static void main(String[] args) {
        Bit bitT = new Bit(true);
        Bit[] bits = { bitT, new Bit(false), new Bit(false), new Bit(false)};
        NumberAsBits number = new NumberAsBits (bits);
        System.out.println(number.toString()); // 8
        System.out.println(number.base2()); // 1000
        bitT = new Bit(false);
        bits[1] = new Bit(true);
        System.out.println(number.toString()); // 8
        System.out.println(number.base2()); // 1000
}
```

תדפיס

```
8
1000
8
1000
```

סיימתם חלק זה? כל הכבוד! העלו את הגרסה האחרונה של עבודתם למערכת ההגשה.