עבודת בית מספר 4

ייצוג מספרים גדולים

מבוא למדעי המחשב, סמסטר א' תשע"ט

המחלקה למדעי המחשב, אוניברסיטת בן-גוריון בנגב

מידע כללי

צוות העבודה

מרצים אחראיים: חן קיסר, יוחאי טוויטו

מתרגלים אחראיים: גליה צימרמן, יערה שובל

תאריכים קשורים

תאריך פרסום: 6.12.2018

מועד אחרון להגשה: 20.12.2018 עד השעה 13:00

מטרות ויעדים

במהלך עבודה זו תתנסו במרכיבים שונים של תכנות מונחה-עצמים ושימוש במבני נתונים. בין היתר תחשפו לנושאים הבאים:

1. ייצוג (בינארי) של מספרים גדולים, וביצוע פעולות אריתמטיות עליהם.
2. דריסה ושימוש בשיטות המתקבלות בירושה מהמחלקה Object.
3. שימוש ברשימות מקושרות ומעבר עליהן באמצעות איטרטורים.
4. ישום הממשק Comparable ושימוש בו.
5. מימוש בנאים מסוגים שונים.
6. שימוש בחריגות.
7. כתיבת בדיקות תוכנה.

הנחיות כלליות

הגשת עבודות בית

1. **קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה.** ודאו שאתם מבינים את כל המשימות. רמת הקושי של המשימות אינה אחידה: הפתרון של חלק מהמשימות קל יותר, ואחרות מצריכות חקירה מתמטית - שאותה תוכלו לבצע בספרייה או בעזרת מקורות דרך רשת האינטרנט. בתשובות שבהן אתם מסתמכים על עובדות מתמטיות שלא הוצגו בשיעורים, יש להוסיף כהערה במקום המתאים בקוד את ציטוט העובדה המתמטית ואת המקור (כגון ספר או אתר).
2. עבודה זו תוגש ביחידים. כדי 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. **לעבודה מצורפים חמישה קבצים:**
   3.1. **Number**.java – אותו תערכו כדי להשלים את משימות 1-14.
   3.2. **NumberTester**.java – אותו תערכו כדי להשלים את משימה 15.
   3.3. **Bit**.java – המכיל את המחלקה Bit כפי שהוגדרה בעבודה מספר 3 (בתוספת מספר שיטות כמפורט מטה). השתמשו בה כפי שהיא ואל תשנו את הקובץ.
   3.4. **LinkedList**.java – המכיל את המחלקה LinkedList בדומה לאופן בו הוגדרה בכתה.
   3.5. **List**.java – המכיל את הממשק List.

   את חמשת הקבצים הנ"ל עליכם לכווץ כקובץ ZIP יחיד ואותו להגיש. שימו לב: עליכם להגיש רק את קבצי ה-Java. אין לשנות את שמות הקבצים, ואין להגיש קבצים נוספים. שם קובץ ה-ZIP יכול להיות כרצונכם, אך באנגלית בלבד. בנוסף, הקבצים שתגישו יכולים להכיל טקסט המורכב מאותיות באנגלית, מספרים וסימני פיסוק בלבד. טקסט אשר יכיל תווים אחרים (אותיות בעברית, יוונית וכד'..) לא יתקבל.
4. קבצים שיוגשו שלא על פי הנחיות אלו לא ייבדקו. את קובץ ה-ZIP יש להגיש ב-Submission System. פרטים בעניין ההרשמה ואופן הגשת העבודה תוכלו למצוא באתר.

בדיקת עבודות הבית

5. עבודות הבית נבדקות גם באופן ידני וגם באופן אוטומטי.
6. סגנון כתיבת הקוד ייבדק באופן ידני. יש להקפיד על כתיבת קוד ברור, 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. כתיבת קוד אשר אינה עומדת בדרישות אלו תגרור הפחתה בציון העבודה.

עזרה והנחיה

7. 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. בשאלות טכניות אפשר גם לגשת לשעות התגבורים, שבהן ניתנת עזרה במעבדה. כמו כן, אתם יכולים להיעזר בפורום ולפנות בשאלות לחבריכם לכיתה. צוות הקורס עובר על השאלות ונותן מענה במקרה הצורך.
8. בכל בעיה אישית הקשורה בעבודה (מילואים, אשפוז וכו'), אנא פנו אלינו **דרך מערכת הפניות**, כפי שמוסבר באתר הקורס.

הערות ספציפיות לעבודת בית זו

9. בעבודה זו 15 משימות וסך הנקודות המקסימלי הוא 100. הניקוד לכל משימה מצויין בגוף העבודה.
10. בעבודה זו מותר להשתמש בכל הידע שנלמד עד למועד הגשתה.
11. **בכל המשימות בעבודה אין להניח קלט תקין, אלא אם כן צויין אחרת.** אם הקלט אינו תקין עליכם לזרוק את

החריגה **IllegalArgumentException** בלבד. אין להשתמש ב- NullPointerException.
כל חריגה שתזרק בקוד שאיננה IllegalArgumentException תגרור הורדה בציון.

12. בעבודה זו אתם יוצרים את השיטה הציבורית ()toString. **אין לקרוא לה** מאף שיטה אחרת שאתם כותבים.
13. בכל אחת מהמשימות מותר להוסיף שיטות עזר כראות עיניכם.
14. עבודה זו משתמשת בשלושה ממשקים מובנים של ג'אווה: Comparable, Iterator, List.
15. בעבודה זו אתם מקבלים 3 מחלקות עזר אותן **אסור לשנות**:
    א. Bit - שאותה פגשתם בעבודה 3. שימו לב שנוספו לה בנאי ומספר שיטות:
    - הבנאי הריק: לא מקבל דבר ומחזיר מופע של המחלקה Bit שערכו 0.
    - השיטה ()getValue אשר לא מקבלת דבר ומחזירה את הערך הבוליאני של הביט (true עבור 1 ו- false עבור 0.
    - השיטה ()toInt אשר לא מקבלת דבר ומחזירה את ערכו של הביט (1 או 0)
    - השיטה (equals(Object הדורסת את השיטה של המחלקה Object. השיטה מקבלת פרמטר מטיפוס Object ומחזירה true אם ורק אם הוא ביט בעל ערך זהה ל- this.

    ב. LinkedList - שימו לב שמחלקה זו מממשת את הממשק List אבל רבות מהשיטות שלו לא נתמכות ואינן ניתנות לשימוש בעבודה זו. לצורך הבהרה: יש להשתמש רק בשיטות הממומשות במחלקה LinkedList שניתנת לכם כאן.

    ג. List – הממשק List.

    <u>אין לשנות מחלקות אלו.</u>
16. חלק מההערות המובאות בקוד הינן בסטנדרט doc, אתם מוזמנים לבצע חיפוש של המילה Javadoc ברשת האינטרנט ולקרוא על פורמט התיעוד בסטנדרט זה (עשוי לסייע לכם בהבנת התיעוד, ובעבודה הבאה).
17. שימו לב לקבצים שעליכם להגיש המופיעים בסעיף 3 לעיל. <u>יש להגיש רק את 5 הקבצים הללו</u> ללא חבילות (packages). גם במהלך כתיבת הקוד בסביבת הפיתוח, יש לדאוג כי כל הקבצים נמצאים תחת חבילת ברירת המחדל (default package), מה שקורה באופן אוטומטי ב- eclipse, כל עוד לא הגדרתם חבילה משלכם.

<u>יושר אקדמי</u>

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.

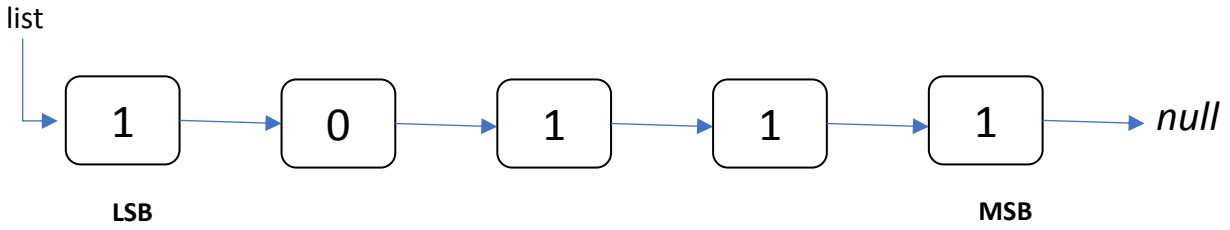
מומלץ לקרוא היטב את כל ההוראות וההוראות המקדימות ורק לאחר מכן להתחיל בפתרון המשימות.

ודאו שאתם יודעים לפתוח קבוצת הגשה (עבור עצמכם) במערכת ההגשות.

העבודה

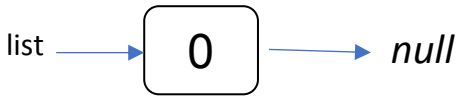
בשפה ג'אווה, לטיפוסים הפרימיטיביים המייצגים מספרים שלמים (int ו- long) יש מגבלות מובנות על הגודל המקסימאלי של המספרים שהם יכולים לייצג. מספרים גדולים יותר צריכים להיות מיוצגים על ידי טיפוסים מורכבים. בעבודה זו נעסוק בייצוג מספרים גדולים ע"י רשימות מקושרות וע"י המחלקה Bit אותה מימשתם בעבודת בית מספר 3. המספרים אותם נייצג ובהם נעסוק יהיו **מספרים טבעיים בלבד** – כלומר מספרים שלמים אי-שליליים.

את המספרים נייצג בבסיס בינארי ע"י רשימה של איברים מהטיפוס Bit, שפגשתם בעבודת הבית מספר 3. האיבר הראשון ברשימה יהיה הספרה הפחות משמעותית (Least Significant Bit, LSB) והאיבר האחרון ברשימה יהיה הספרה המשמעותית ביותר (Most significant Bit, MSB). לדוגמה, את המספר הבינארי 11101 (29 בבסיס 10) נייצג על ידי הרשימה המקושרת הבאה:



ברשימה זו, האיבר הראשון הוא הספרה האחרונה של המספר הבינארי 11101, האיבר השני הוא הספרה הלפני אחרונה, האיבר השלישי הוא הספרה שלפני הספרה שלפני הספרה האחרונה, וכך הלאה. האיבר האחרון ברשימה הוא הספרה הראשונה במספר הבינארי.

המספר אפס ייוצג על ידי רשימה שבה איבר אחד, ביט שערכו 0.



המחלקה המרכזית אותה תכתבו היא המחלקה Number המייצגת מספר גדול, באופן המתואר לעיל. מעבר לייצוג המספר הגדול, המחלקה תספק גם פונקציונליות לבניית מספר, בדיקת חוקיות של מספר, חיבור וכפל של מספרים גדולים, ועוד, כמתואר בסעיפים הבאים.

תבנית של המחלקה Number אותה עליכם לממש ניתן למצוא בקובץ Number.java (בקובץ זה יש להשלים את משימות 1-14). שדות המחלקה והבנאי הריק כבר נכתבו. למחלקה Number שדה מטיפוס List<Bit> שנועד לייצג את המספר הגדול כרשימה כפי שהוגדר לעיל. בנוסף יש במחלקה שלושה קבועים בהם תוכלו להשתמש כראות עיניכם. **אין לשנות או להוסיף שדות למחלקה, ואין לשנות את הבנאי הריק, את כותרת המחלקה ואת החתימות של הפונקציות**. אל המחלקה Number כבר יובאו כל המחלקות והממשקים שיש לייבא, **אין לייבא מחלקות וממשקים נוספים**.

עליכם <u>להשלים את כל השיטות</u> המוגדרות במסמך זה ואשר מופיעות בקוד שקיבלתם ללא מימוש. שימו לב, בתבנית הקוד של כל הפונקציות הללו מופיעה השורה:

```
throw new UnsupportedOperationException("Delete this line and implement
the method.");
```

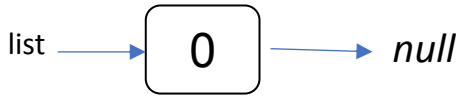
בכל פונקציה בה מופיעה שורה זו, יש למחוק את השורה **כולה** (החל מהמילה throw ועד הנקודה ופסיק) ולכתוב מימוש מלא לפונקציה. אין להוסיף קבצים חדשים מעבר לקבצים שקיבלתם, אך ניתן להוסיף פונקציות עזר פרטיות ללא מגבלה, **באותו הקובץ**.

בנאים (15 נקודות)

למחלקה Number מספר בנאים אשר את חלקם עליכם לממש.

הבנאי הריק:

הבנאי הריק אינו מקבל פרמרטים והוא בונה את המספר אפס. המספר אפס מיוצג על-ידי רשימה באורך 1 שבה ביט אחד שערכו 0.



בנאי זה כבר מומש ואין לשנותו. ניתן לראות את המימוש בין קבצי הקוד שניתנו לכם.

```
/**
 * Constructs a new Number defaulted to the value zero.
 */
public Number(){…}
```

משימה 1: מימוש בנאי פרמטרי (5 נקודות)

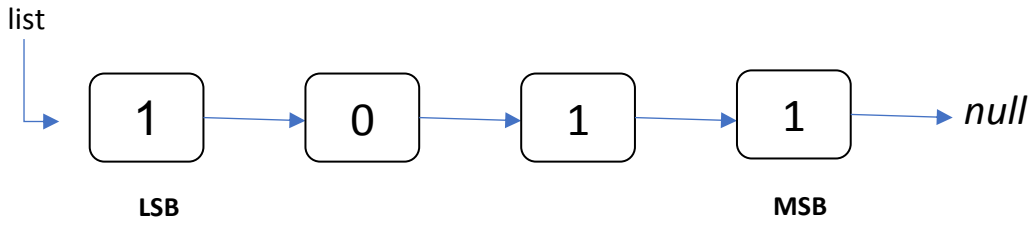
במשימה זו עליכם לממש בנאי המקבל ערך מטיפוס int. הקלט הוא מספר בבסיס 10. על הבנאי להמיר את המספר לבסיס 2 ולייצג אותו כאובייקט מטיפוס Number כנזכר לעיל (רשימה של ביטים).

זיכרו כי האיבר הראשון ברשימה יהיה הספרה הפחות משמעותית (Least Significant Bit, LSB) והאיבר האחרון ברשימה יהיה הספרה המשמעותית ביותר (Most significant Bit, MSB). כלומר, אם נעבור על הרשימה מתחילתה עד סופה נקבל את הייצוג הבינארי של המספר בסדר הפוך.

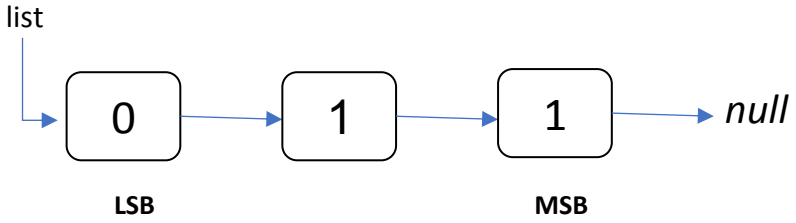
יש לוודא שהקלט תקין. בהקשר זה, הקלט הוא מספר בבסיס 10 (מתקבל כ int), אך אינו בהכרח מספר טבעי, יתכן שהוא שלילי. המחלקה Number אינה תומכת במספרים שליליים והם נחשבים קלט חריג.

```
/**
 * Constructs a new Number from an int.
 * @param number an int representing a decimal number
 */
public Number(int number) { // assignment #1
    // TODO: implement!
}
```

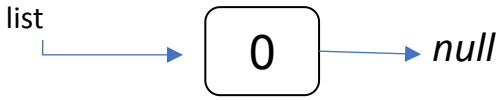
למשל עבור קלט של המספר 13, המספר בייצוג בינארי (בסיס 2) הינו 1101, ועל כן הרשימה במחלקה Number תראה כך:



עבור קלט של המספר 6, המספר בייצוג בינארי הינו 110, ועל כן הרשימה במחלקה Number תראה כך:



עבור קלט של המספר 0, המספר בייצוג בינארי הינו 0, ועל כן הרשימה במחלקה Number תראה כך:



עבור קלט של המספר -5, תזרק חריגה מסוג IllegalArgumentException.

משימה 2: מימוש בנאי פרמטרי (5 נקודות)

במשימה זו עליכם לממש בנאי המקבל ערך מטיפוס מחרוזת. מחרוזת הקלט אמורה לייצג מספר בבסיס 10 אך דבר זה אינו מובטח (כזכור, בכל סעיפי העבודה הזו, לא ניתן להניח את תקינות הקלט, אלא אם כן נאמר במפורש אחרת בגוף הסעיף). אם המחרוזת אינה מייצגת מספר חוקי בבסיס 10, על הבנאי לזרוק חריגה. כמו כן יש לבדוק מקרים חריגים נוספים (לדוגמה, מספר שלילי). אחרת, אם הקלט תקין, על הבנאי להמיר את המספר המיוצג במחרוזת הקלט מבסיס 10 לבסיס 2 ולייצג אותו כטיפוס מסוג Number כנזכר לעיל (רשימה של ביטים). **ניתן להניח שאם המחרוזת אכן מייצגת מספר בבסיס 10, המספר אינו גדול מ- Integer.MAX_VALUE.** שימו לב כי למרות שאנו מגבילים את גודל המספר שאותו בונים, מספר זה עשוי לגדול (ולהיות מספר גדול מאוד) ע"י פעולות כמו increment אשר הגדרתה מופיעה בהמשך העבודה.

דוגמאות למחרוזות תקינות:	דוגמאות למחרוזות לא תקינות:
"123"	""
"0"	"023"
"556291"	"23$@ 56"
"10"	"-56"

```
/**
 * Constructs a new Number from a String.
 * @param s a String (possibly) representing a decimal number.
 */
public Number(String s) { // assignment #2
    // TODO: implement!
}
```

משימה 3: מימוש בנאי מעתיק (5 נקודות)

במשימה זו עליכם לממש בנאי מעתיק (העתקה עמוקה). בנאי זה מקבל ערך מטיפוס מספר גדול (Number) כקלט ובונה ממנו מספר גדול חדש. על המספר הגדול החדש לייצג מספר **השווה בערכו** למספר הקלט. כמו כן, לאחר בניית המספר הגדול החדש, שני המספרים הללו (מספר הקלט והמספר החדש) אינם קשורים זה לזה – שינוי באחד אינו גורם לשינוי בשני. למשל הגדלה של אחד מהמספרים באחד לא תגרור שינוי במספר השני (שימו לב לדוגמה מטה).

```
/**
 * Constructs a new Number which is a deep copy of the provided Number.
 * @param number a Number to be copied
```

```
 */
public Number(Number number) { // assignment #3
    // TODO: implement!
}
```

למשל עבור קטע הקוד הבא הנמצא מחוץ למחלקה Number, יודפס true, אחר כך יודפס false ולבסוף יודפס (1100, 1101):

```
Number num1 = new Number(12);
Number num2 = new Number(num1);
System.out.println(num1.equals(num2)); // true
num1.increment(); //See below (assignment 6) the description of the method
System.out.println(num1.equals(num2)); // false
System.out.println("(" + num2 + ", " + num1 + ")"); //(1100, 1101)
```

השיטות equals ו- increment של Number המופיעות בקטע הקוד הקודם מוגדרות בהמשך מסמך זה (משימה 8 ו-6, בהתאמה).

שיטות כלליות (20 נקודות)

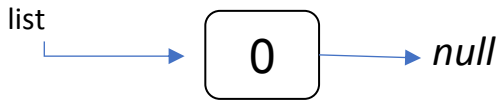
למחלקה Number מספר שיטות כלליות אותן עליכם לממש.

משימה 4: מימוש השיטה isZero() (5 נקודות)

במשימה זו עליכם לממש את השיטה isZero(), אשר אינה מקבלת פרמטרים ומחזירה ערך בוליאני. שיטה זו מחזירה ערך true אם ורק אם המספר המיוצג על ידי האובייקט (this) הוא המספר אפס. למשל בקטע הקוד הבא הנמצא מחוץ למחלקה, יודפס תחילה true ואחר כך false.

```
Number num0 = new Number(0);
System.out.println(num0.isZero());  //true
Number num1 = new Number(1);
System.out.println(num1.isZero());  //false
```

תזכורת: אובייקט מהטיפוס Number המייצג את המספר אפס מכיל רשימה באורך 1 ובה ביט יחיד שערכו 0.

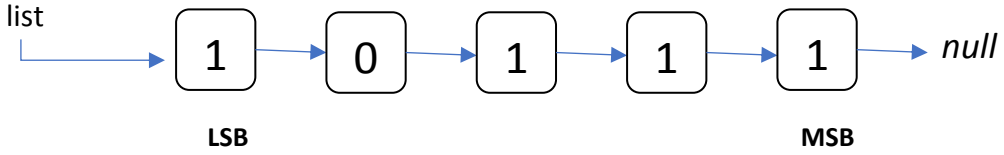



```
/**
 * Checks if this Number is zero.
 * @return true if and only if this object representing the number zero.
 */
public boolean isZero() { // assignment #4
    // TODO: implement!
}
```

משימה 5: מימוש השיטה bitIterator() (5 נקודות)

במשימה זו עליכם לממש את השיטה bitIterator() של המחלקה Number. השיטה אינה מקבלת פרמטרים ומחזירה איטרטור (Iterator) על הביטים (Bit) שברשימה המייצגת את המספר. על האיטרטור המוחזר להיות כזה שעובר על הביטים של המספר החל מה- LSB ועד ל- MSB, כלומר הביט הראשון באיטרטור הוא הביט הראשון בשדה הרשימה של המחלקה.

למשל, עבור המספר 11101 (29 בעשרוני) והרשימה הבאה המייצגת אותו, הביט הראשון באיטרטור שיחזור מהשיטה bitIterator() יהיה 1, השני 0 ושלושת הביטים הבאים אחר כך יהיו 1.



כדי להבהיר יותר את הדוגמה התבוננו בקטע הקוד הבא הנמצא מחוץ למחלקה. ההערות בקוד מציינות את שיודפס בעקבות כל שורה.

```
Number num = new Number(29);
Iterator<Bit> iter = num.bitIterator();
System.out.print(iter.next());      //1
System.out.print(iter.next());      //0
System.out.print(iter.next());      //1
System.out.print(iter.next());      //1
System.out.print(iter.next());      //1
System.out.print(iter.next());      // Exception in thread "main"
                                    // java.util.NoSuchElementException
```

```
/**
 * Returns an iterator over the Bit objects in the representation of this
 * number, which iterates over the Bit objects from LSB (first) to MSB (last).
 * @return a LSB-first iterator over the Bit objects in the representation of
 * this number.
 */
public Iterator<Bit> bitIterator() { // assignment #5
    // TODO: implement!
}
```

משימה 6: מימוש השיטה increment() (5 נקודות)

במשימה זו עליכם לממש את השיטה increment() של המחלקה Number. השיטה מגדילה את ערכו של המספר (ששיטתו זו הופעלה) ב 1.

```
/**
 * Adds 1 to the number
 */
public void increment() {      // assignment #6
   // TODO: implement!
}
```

לדוגמה התבוננו בקטע הקוד הבא הנמצא באיזשהי פונקציית main מחוץ למחלקה Number. ההערות בקוד מציינות את שיודפס בעקבות כל שורה.

```
Number num = new Number();
System.out.print(num);  //0
num.increment();
System.out.print(num);  //1
num.increment();
System.out.print(num);  //10
num.increment();
System.out.print(num);  //11
num.increment();
num.increment();
System.out.print(num);  //101
```

משימה 7: מימוש השיטה isLegal(String) (5 נקודות)

זוהי שיטה סטטית של המחלקה המקבלת מחרוזת המייצגת מספר ומחזירה true אם"ם המחרוזת מייצגת מספר חוקי בבסיס 10. חישבו מהו מספר חוקי. האם כל אורך של מחרוזת יכול לייצג מספר חוקי? האם כל תו שיכול להיות במחרוזת יכול להיות גם במספר? האם כל ספרה יכולה להיות בתחילת מספר? חיזרו אל משימה 2 כדי לראות דוגמאות למחרוזות המייצגות מספרים חוקיים ולא חוקיים. שימו לב שכאן אין הגבלה על הערך המקסימלי אותו המחרוזת יכולה לייצג.

```
/**
 * Checks if a provided String represent a legal decimal number.
 * @param s a String that possibly represents a decimal number, whose legality
 * to be checked.
 * @return true if and only if the provided String is a legal decimal number
 */
public static boolean isLegal(String s) { // assignment #7
   // TODO: implement!
}
```

שיטות הדורסות שיטות של המחלקה Object (10 נקודות)

המחלקה Number דורסת שתי שיטות של המחלקה Object אותן עליכם לממש.

משימה 8: מימוש השיטה equals(Object) (5 נקודות)

במשימה זו עליכם לממש את השיטה equals(Object) של המחלקה Number, אשר דורסת את השיטה equals(Object) של המחלקה Object. השיטה מקבלת אובייקט ומחזירה ערך בוליאני. השיטה מחזירה ערך true אם ורק אם אובייקט הקלט הוא מסוג מספר גדול (Number) אשר ערכו שווה לערך המספר המיוצג על ידי האובייקט הנוכחי (this).

שימו לב כי כל קלט לשיטה זה הוא קלט תקין ועל השיטה לדעת להחזיר true או false לכל קלט שהיא מקבלת בלי יוצא מן הכלל. כלומר, **שיטה זו <u>לא</u> זורקת חריגה.**

```
/**
 * Compares the specified object with this Number for equality.
 * Returns true if and only if the specified object is also an
 * object of type Number, which represents the same number.
 * @param obj the object to be compared for equality with this Number
 * @return true if and only if the specified object is equal to this object
 */
public boolean equals(Object obj) { // assignment #8
    // TODO: implement!
}
```

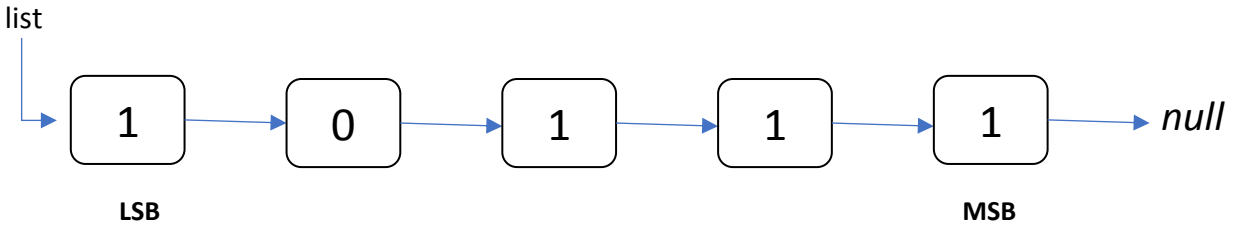
לדוגמה נסתכל בקטע הקוד הבא הנמצא מחוץ למחלקה. גם כאן בהערות מופיע מה שיודפס בכל שורה (true ואחר כך 3 פעמים false):

```
Number num1 = new Number(29);
Number num2 = new Number(29);
Number num3 = new Number(5);
System.out.println(num1.equals(num2));        //true
System.out.println(num1.equals(num3));        //false
System.out.println(num1.equals(null));        //false
System.out.println(num1.equals("29"));        //false
```

משימה 9: מימוש השיטה toString() (5 נקודות)

במשימה זו עליכם לממש את השיטה toString() של המחלקה Number, אשר דורסת את השיטה toString() של המחלקה Object. על השיטה להחזיר מחרוזת המכילה את הייצוג הבינארי (הרגיל, לא הפוך) של המספר המיוצג על ידי המחלקה Number.

לדוגמה, אם המספר המיוצג על ידי עצם מטיפוס Number הינו המספר 29, כלומר העצם מטיפוס Number מכיל את הרשימה הבאה:



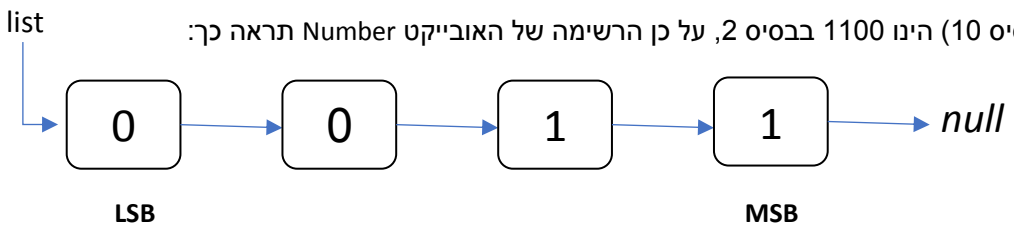
עליכם להחזיר את המחרוזת **"11101"**. שימו לב כי המחרוזת שמוחזרת מכילה את הביטים המובאים ברשימה בסדר הפוך. כלומר, היא מחזירה את הייצוג הבינארי הרגיל של המספר 29. יש לבנות את מחרוזת הייצוג הרגיל של המספר על ידי מעבר על הרשימה ואגירת הביטים הנמצאים בה בסדר הנכון (**הפוך מסדר הרשימה**). יש לעבור על הרשימה רק פעם אחת.

```
/**
 * Returns a string representation of this Number
 * as a binary number.
 * @return a string representation of this Number
 */
public String toString() { // assignment #9
    // TODO: implement!
}
```

דוגמה נוספת: לאחר ביצוע קטע הקוד הבא יודפס 1100.

```
Number num = new Number(12);
System.out.println(num.toString());        //1100
```

המספר 12 (בבסיס 10) הינו 1100 בבסיס 2, על כן הרשימה של האובייקט Number תראה כך:



והשיטה toString() תחזיר 1100 **ולא** 0011.

שיטות השוואת גודל (25 נקודות)

המחלקה Number מממשת את הממשק Comparable. כלומר, עליכם לממש את השיטה compareTo(Number). לפני שתממשו שיטה זו נבקשכם לממש שתי שיטות סטטיות פומביות השייכות למחלקה ומשוות בין שני אובייקטים מסוג Number: שיטה אחת lessEq(Number,Number) המבצעת את הפעולה החשבונית "קטן-שווה" (≤) ומחזירה ערך בוליאני מתאים, שיטה שנייה lessThan(Number,Number) המבצעת את הפעולה החשבונית "קטן מ-" (<) ומחזירה ערך בוליאני מתאים. שתי השיטות קשורות זו בזו, **אנא קיראו את ההוראות של שתיהן לפני מימושן.**

בכתיבת שתי השיטות הבאות בחנו את זוגות המספרים (הבינאריים) הבאים וחישבו איזו תשובה כל שיטה שכתבתם תחזיר. כלומר, החליפו את סימן השאלה בסימן "קטן מ-" או "קטן-שווה" בהתאם לשיטה וחישבו האם ההצהרה היא true או false. נסו להבין איך מתוך מעבר על רשימת הביטים של המספר תוכלו להחזיר את הערך הנכון (true/false). זיכרו כי השיטות צריכות לפעול גם עבור מספרים גדולים מאוד, גדולים יותר מ- Long.MAX_VALUE.

i.	10 ? 11	iv.	10 ? 10
ii.	10 ? 1	v.	1011 ? 1001
iii.	1 ? 10	vi.	1000 ? 111

לדוגמה, במקרה iv, ההצהרה 10<10 היא false, ולעומת זאת ההצהרה 10≤10 היא true. לכן במקרה זה lessThan תחזיר false, וlessEq תחזיר true.

משימה 10: מימוש השיטה lessEq(Number, Number) (10 נקודות)

במשימה זו עליכם לממש את השיטה הסטטית lessEq(Number, Number) של המחלקה Number. שיטה סטטית זו מקבלת שני פרמטרים מטיפוס Number ומחזירה true אם ורק אם הפרמטר הראשון קטן או שווה לפרמטר השני, כמספר. שיטה זו תזרוק חריגה (IllegalArgumentException) במידה והקלט לא תקין.

למשל קטע הקוד הבא הנמצא מחוץ למחלקה ידפיס 3 פעמים true ואחר כך false:

```
Number num1 = new Number(7);                    //in binary form: 111
Number num2 = new Number(7);                    //in binary form: 111
Number num3 = new Number(2);                    //in binary form: 10
Number num4 = new Number(3);                    //in binary form: 11
System.out.println(Number.lessEq(num1, num2));  //true
System.out.println(Number.lessEq(num2, num1));  //true
System.out.println(Number.lessEq(num3, num4));  //true
System.out.println(Number.lessEq(num4, num3));  //false
```

הנחיות:

א. ניקוד מלא יינתן למשימה זו רק אם במקרה הגרוע ביותר היא תעבור לכל היותר **פעם אחת** על כל אחד מהביטים ברשימה של כל אחד מהמספרים. פתרון שעובר על הביטים יותר מפעם אחת ייחשב בלתי יעיל ויקבל ציון חלקי.
שימו לב למימוש של השיטות size() ו- get(int) במחלקה LinkedList. שיטות אלה אינן יעילות בהקשר של סעיף זה מאחר וקריאה יחידה לאחת מהן עלולה לגרור מעבר על כל (או על רוב) הביטים ברשימה.

ב. בפרט אין להשתמש בקריאה לשיטה toString(). פתרון שישתמש בשיטה toString() לא יקבל נקודות כלל.

```
/**
 * Compares the two provided numbers, and returns true if
```

```
 * the first parameter is less than or equal to the second
 * parameter, as numbers.
 * @param num1 the first number to compare
 * @param num2 the second number to compare
 * @return true if and only if the first number is less than
 * or equal to the second parameter, as numbers.
 */
public static boolean lessEq(Number num1, Number num2) { // assignment #10
    // TODO: implement!
}
```

משימה 11: מימוש השיטה lessThan(Number, Number) (10 נקודות)

במשימה זו עליכם לממש את השיטה הסטטית lessThan(Number, Number) של המחלקה Number. שיטה סטטית זו מקבלת שני פרמטרים מטיפוס Number ומחזירה true אם ורק אם הפרמטר הראשון קטן ממש מהפרמטר השני, כמספר. שיטה זו תזרוק חריגה (IllegalArgumentException) במידה והקלט לא תקין.

הנחיות:

א. ניקוד מלא יינתן למשימה זו רק אם במקרה הגרוע ביותר היא תעבור לכל היותר **פעם אחת** על כל אחד מהביטים ברשימה של כל אחד מהמספרים. פתרון שעובר על הביטים יותר מפעם אחת ייחשב בלתי יעיל ויקבל ציון חלקי.
שימו לב למימוש של השיטות size() ו- get(int) במחלקה LinkedList. שיטות אלה אינן יעילות בהקשר של סעיף זה מאחר וקריאה יחידה לאחת מהן עלולה לגרור מעבר על כל (או על רוב) הביטים ברשימה.

ב. בפרט אין להשתמש בקריאה לשיטה toString(). פתרון שישתמש בשיטה toString() לא יקבל נקודות כלל.

למשל קטע הקוד הבא הנמצא מחוץ למחלקה ידפיס פעמיים false ואחר כך true:

```
Number num1 = new Number(7);                    //in binary form: 111
Number num2 = new Number(2);                    //in binary form: 10
Number num3 = new Number(2);                    //in binary form: 11
System.out.println(Number.lessThan(num1, num2));      // false
System.out.println(Number.lessThan(num2, num3));      //false
System.out.println(Number.lessThan(num3, num1));      //true

/**
 * Compares the two provided numbers, and returns true if
 * the first parameter is strictly less than the second
 * parameter, as numbers.
 * @param num1 the first number to compare
 * @param num2 the second number to compare
 * @return true if and only if the first number is strictly
 * less than the second parameter, as numbers.
 */
public static boolean lessThan(Number num1, Number num2) { // assignment #11
    // TODO: implement!
}
```

משימה 12: מימוש השיטה compareTo(Number) (5 נקודות)

במשימה זו עליכם לממש את השיטה compareTo(Number) של המחלקה Number. שיטה זו מקבלת פרמטר מטיפוס Number ומחזירה מספר שלם. הערך המוחזר צריך להיות שלילי אם האובייקט הנוכחי (this) קטן מאובייקט הקלט (כמספר), אפס אם שני האובייקטים שווים, וחיובי אם האובייקט הנוכחי גדול מאובייקט הקלט. שיטה זו תזרוק חריגה (IllegalArgumentException) במידה והקלט לא תקין.

למשל קטע הקוד הבא הנמצא מחוץ למחלקה ידפיס אפס, אחר כך מספר שלילי ולבסוף מספר חיובי:

```
Number num1 = new Number(7);                          //in binary form: 111
Number num2 = new Number(2);                          //in binary form: 10
Number num3 = new Number(3);                          //in binary form: 11
System.out.println(num1.compareTo(num1));             // 0
System.out.println(num2.compareTo(num3));             // negative number
System.out.println(num3.compareTo(num2));             // positive number

/**
 * Compares this object with the specified object for order.  Returns a
 * negative integer, zero, or a positive integer as this object is less
 * than, equal to, or greater than the specified object.
 *
 * @param o the object to be compared.
 * @return a negative integer, zero, or a positive integer as this object
 * is less than, equal to, or greater than the specified object.
 */
public int compareTo(Number o) { // assignment #12
    // TODO: implement!
}
```

שיטות אריתמטיות (20 נקודות)

בחלק זה תממשו חלק מפעולות החשבון הבסיסיות בין מספרים, כשיטות סטטיות של המחלקה Number.

משימה 13: מימוש השיטה add(Number, Number) (10 נקודות)

במשימה זו עליכם לממש את השיטה add(Number num1, Number num2) המקבלת 2 מספרים ומחזירה את סכומם: num1 + num2. למשל, עבור *num1* המייצג את המספר הבינארי **11**, *num2* המייצג את המספר הבינארי **10** יוחזר הערך **101**.

<u>הערה:</u> אין להשתמש בקריאה לשיטה increment(). פתרון שישתמש בשיטה increment() לא יקבל נקודות כלל.

העזרו בשיטות לחיבור ביטים מהמחלקה Bit שכתבתם בעבודה 3. על מנת לחשוב על האופן בו מתבצעת פעולת החיבור במחלקה, באפשרותכם לחשוב על האופן שבו מתצע חיבור במאונך:

```
  1101
+  101
 10010
```

```
/**
 * Adds the specified Number objects, and returns their sum.
 * @param num1 the first addend
 * @param num2 the second addend
 * @return the sum of the specified Number objects.
 */
public static Number add(Number num1, Number num2) { // assignment #13
    // TODO: implement!
}
```

משימה 14: מימוש השיטה multiply(Number, Number) (10 נקודות)

במשימה זו עליכם לממש את השיטה multiply(Number num1, Number num2) המקבלת 2 מספרים ומחזירה את מכפלתם: num1 * num2. למשל, עבור *num1* המייצג את המספר הבינארי **11** (3 בבסיס 10), *num2* המייצג את המספר הבינארי **10** – יוחזר הערך **110**.

תוכלו להעזר באופן בו מבוצע כפל מאונך:

```
    1010
*    101
    1010
+  00000
+ 101000
  110010
```

```
/**
 * Multiply the specified Number objects, and returns their product.
 * @param num1 the first factor
 * @param num2 the second factor
 * @return the product of the specified Number objects.
 */
public static Number multiply(Number num1, Number num2) { // assignment #14
      // TODO: implement!
}
```

מחלקת בדיקה (10 נקודות)

התבוננו בקובץ NumberTester.java אותו קיבלתם יחד עם קבצי העבודה. קובץ זה מכיל את מחלקת הבדיקה של המחלקה Number. פונקציית ה main של מחלקת הבדיקה מפעילה פונקציות בדיקה שונות – אחת לכל שיטה <u>ציבורית</u> של המחלקה Number (למעט הבנאים שנבדקים כולם בפונקציית בדיקה אחת).

משימה 15: כתיבת מחלקת בדיקות (10 נקודות)

עליכם להשלים את כל פונקציות הבדיקה המובאות בקובץ. כל פונקציה צריכה לבדוק את השיטה הרלוונטית במחלקה Number על פי שמה. בכל פונקציה יש לכתוב לפחות 3 בדיקות הבודקות את הפונקציה הרלוונטית במחלקה Number. בפונקציה שבודקת את הבנאים, יש לכתוב לפחות 3 בדיקות, לפחות בדיקה אחת לכל בנאי.

לכל שיטה שאתם בודקים חישבו על המקרה הטיפוסי ועל מקרי הקצה הרלוונטים, ובדקו את נכונות השיטות שכתבתם במקרים אלה.

במשימה זאת בלבד, בשונה מהמשימות האחרות, ניתן להשתמש בשיטה toString() של המחלקה Number.

קבצי העבודה

3 קבצים נספחים לשימושכם אותם אין לשנות:

1. Bit.java
2. LinkedList.java

3. List.java

שני קבצים בהם יש להשלים את המשימות:

1. Number.java
2. NumberTester.java

בהצלחה!