מבוא למדעי המחשב – סמסטר א' תשע"ז

עבודת בית מספר 5 (מבני נתונים)

צוות העבודה:

מרצה אחראית: מיכל שמש

מתרגלים אחראים: אור עמי, שקד מטר

תאריך פרסום: 5.1.17

תאריך הגשה: 19.1.17

הוראות מקדימות

הגשת עבודות בית

1. **קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה**. ודאו שאתם מבינים את כל השאלות. רמת הקושי של המשימות אינה אחידה.
2. אנו ממליצים לפתור ולהגיש את העבודות בזוגות. במקרה זה, עליכם להירשם כזוג להגשת העבודה במערכת ההגשות (Submission System) המחלקתית. מותר גם להגיש לבד – גם כן דרך מערכת ההגשות. אין לפתור את תרגילי הבית עם כל אדם אחר, אלא אם כן נרשמתם כזוג להגשה.
3. אין לשנות את שמות הקבצים או את חתימות הפונקציות כפי שהן מופיעות בקובצי העבודה.
4. אין להגיש קבצים נוספים.
5. 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.
6. קבצים שיוגשו שלא על פי הנחיות אלו לא ייבדקו.
7. את קובץ ה-ZIP יש להגיש ב-Submission System.
8. אין להשתמש ב- **packages**. אם תעשו בהן שימוש עבודתכם לא תתקבל על ידי מערכת ההגשות. בידקו כי המילה **package** אינה מופיעה בקובצי ההגשה שלכם.

בדיקת עבודות הבית

9. עבודות הבית נבדקות באופן ידני וכן באופן אוטומטי. הבדיקה האוטומטית מתייחסת לערכי ההחזרה של הפונקציות או לפעולות אשר הן מבצעות וכן לפלט התכנית המודפס למסך (אם קיים). לכן, יש להקפיד על ההוראות ולבצע אותן במדוייק. כל הדפסה אשר אינה עונה בדיוק על הדרישות המופיעות בעבודה (כולל שורות, רווחים, סימני פיסוק או כל תו אחר - מיותרים, חסרים או מופיעים בסדר שונה מהנדרש), לא תעבור את הבדיקה האוטומטית ולכן תגרור פגיעה בציון.
10. 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.

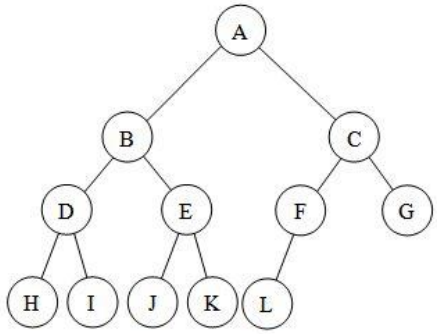
עזרה והנחיה

11. לכל עבודת בית בקורס יש צוות שאחראי לה (מרצה/ים ומתרגלים). ניתן לפנות לצוות בשעות הקבלה. פירוט שמות האחראים לעבודה מופיע באתר הקורס, כמו גם פירוט שעות הקבלה. בשאלות טכניות אפשר גם לגשת לשעות "עזרה במעבדה". כמו כן, אתם יכולים להיעזר בפורום ולפנות בשאלות לחבריכם לכיתה. צוות הקורס עובר על השאלות ונותן מענה במקרה הצורך.
12. בכל בעיה אישית הקשורה בעבודה (מילואים, אשפוז וכו'), אנא צרו את הפנייה המתאימה במערכת הגשת העבודות, כפי שמוסבר באתר הקורס.

הערות ספציפיות לעבודת בית זו

13. לעבודה זו מצורפים קבצי java עם השמות הנדרשים כמפורט בכל משימה. צרו תיקייה חדשה והעתיקו את קובצי ה- java לתוכה. עליכם לערוך את הקבצים האלו בהתאם למפורט בתרגיל ולהגישם כפתרון, מכווצים כקובץ ZIP יחיד. שימו לב: עליכם להגיש רק את קובצי ה-java הנדרשים.
14. בעבודה זו ניתן להגדיר פונקציות (עזר) נוספות, לפי שיקולכם. פונקציות אלו ייכתבו בתוך קובצי המשימה הרלוונטיים.
15. בעבודה זו יהיה עליכם לכתוב קובצי בדיקה משלכם על מנת לוודא את נכונות הקוד. (יכולת) כתיבה נכונה של קובצי בדיקה היא מדד עצמי מצויין לצורך הבנת המשימה ומהווה חלק חשוב בפתרון נכון של המשימות בעבודה.
16. הגדרה: עץ בינארי שלם (complete binary tree) הינו עץ בינארי בו כל הרמות בעץ מלאות, פרט אולי לתחתונה, בה הקדקודים נמצאים בצד השמאלי.

דוגמה:



יושר אקדמי

הימנעו מהעתקות! ההגשה היא בזוגות או ביחידים. אם תוגשנה שתי עבודות עם קוד זהה או אפילו דומה - זוהי העתקה, אשר תדווח לאלתר לוועדת משמעת. בפרט אם פתרתם עבודה בזוג - יש להגיש עותק יחיד של העבודה. אם טרם עיינתם בסילבוס הקורס אנא עשו זאת כעת.

ניהול ספר טלפונים

בעבודה זו נממש מערכת ספר טלפונים. במערכת אוסף של רשומות מהסוג <שם, מספר>. המערכת תומכת בפעולות הבאות: יצירת ספר טלפונים חדש (ריק), הוספת רשומה לספר, מחיקת רשומה מהספר, חיפוש מספר טלפון לפי שם, וחיפוש שם לפי מספר טלפון. כדי לתמוך בחיפוש יעיל לפי שם ולפי מספר טלפון במערכת יתוחזקו שני עצי חיפוש בינאריים. בעץ אחד הרשומות יהיו ממויינות לפי שם ובעץ השני הרשמות יהיו ממויינות לפי מספר הטלפון. מכיוון שפעולות ההוספה והמחיקה עלולות להוציא את העצים מאיזון המערכת תומכת גם בפעולה המאזנת את העצים.

בספר הטלפונים שנממש שמות ומספרי טלפונים צריכים להיות יחודיים. לא ייתכנו שתי רשומות עם אותו השם וגם לא ייתכנו שתי רשומות עם אותו מספר הטלפון.

משימה 1: מבנה הרשומה (0 נקודות)

הרשומות בספר הטלפונים מתוארות על ידי הקובץ PhoneEntry.java. במשימה זו תבצעו הכרות עם המחלקה הנתונה לכם בקובץ זה ושבה תשתמשו בהמשך העבודה. אין לשנות את תוכן הקובץ.

במחלקה PhoneEntry בנאי יחיד

- public PhoneEntry (String name, Integer number)

השיטות הציבוריות במחלקה הן:

- public String getName()
- public Integer getNumber()
- public String toString()

קראו היטב את הקוד שבקובץ PhoneEntry.java. עליכם להכיר את כל פרטי המחלקה, את השדות, הבנאים והשיטות שלה. כפי שתראו בקוד, שמות מיוצגים על ידי מחרוזות לא ריקות ומספרי טלפון על ידי מספרים חיוביים.

משימה 2: השוואת רשומות (10 נקודות)

במשימה זו תשלימו את הגדרת שתי המחלקות הבאות בקבצים שקיבלתם.

- public class EntryComparatorByName implements Comparator
- public class EntryComparatorByNumber implements Comparator

מחלקות אלו מממשות את השיטה public int compare(Object o1, Object o2) המוגדרת בממשק Comparator. במחלקה EntryComparatorByName שיטה זו משווה בין רשומות (מסוג PhoneEntry) לפי שם (לפי הסדר הלקסיקוגרפי על מחרוזות) ובמחלקה EntryComparatorByNumber לפי מספר (לפי יחס הסדר הטבעי על מספרים). עליכם לממש את השיטה בשתי המחלקות.

שימו לב שבקבצים EntryComparatorByName.java ו- EntryComparatorByNumber.java מופיעה השורה import java.util.Comparator. זהו הממשק Comparator כפי שמוגדר ב- java. מומלץ להיזכר בפירטי הממשק Comparator כפי שמתואר ב API של java.

משימה 3: ממשקים נתונים / מחלקות נתונות (0 נקודות)

במשימה זו תבצעו הכרות עם הממשקים והמחלקות הבאים הנתונים לכם ושבהם תשתמשו בהמשך העבודה. אין לשנות את הקבצים הנתונים. שימו לב שהממשק List הנתון הוא חלקי ותואם את מטרות העבודה.

- public interface Stack
- public interface Queue
- public interface List
- public class StackAsDynamicArray implements Stack
- public class QueueAsLinkedList implements Queue
- public class DynamicArray implements List
- public class LinkedList implements List

קראו היטב את הקוד בקבצים המתאימים. עליכם להכיר את כל פרטי המחלקות, את השדות, הבנאים והשיטות שלהן.

משימה 4: עצים בינאריים (10 נקודות)

במשימה זו נתונות לכם המחלקות BinaryNode, BinaryTree. מחלקות אלו זהות למחלקות שנלמדו בהרצאה.
במשימה זו תשלימו במחלקה BinaryNode את הגדרת השיטה:

- public String toString()

השיטה toString() במחלקה BinaryTree נתונה לכם. אם העץ אינו ריק היא קוראת לשיטה toString() שבמחלקה BinaryNode. במחלקה BinaryNode השיטה פועלת כך שאם נדפיס את המחרוזת שהיא מחזירה נקבל שורת הדפסה אחת לכל קודקוד בעץ. בשורה זו יופיעו 2*d רווחים, כאשר d עומק הקודקוד בעץ ואח"כ יודפס (באותה השורה) המידע שבקודקוד. הקודקודים יודפסו בסדר inorder.

הדרכה: במשימה זו יש להשתמש בשיטות עזר פרטיות בהתאם לצורך. אין צורך לחשב מהו העומק של קודקוד בעץ באופן ישיר.

לאחר שתשלימו את הגדרת השיטה toString במחלקה BinaryNode ולאחר שתשלימו את (השיטה insert ב-) משימה 5ב, הקוד בקובץ TestToString.java ידפיס למסך את הפלטים הבאים:

```
----------t1:----------
    1
  2
    3
4
    5
  6
    7
      8
```

```
----------t2:----------
    1
  2
      3
    4
5
  6
    7
      8
```

```
----------t3:----------
  1
2
  3
    4
      5
        6
          7
            8
```

```
----------t4:----------
            1
          2
        3
      4
    5
  6
7
8
```

משימה 5: עצי חיפוש בינאריים, איטראטור

משימה 5א: (0 נקודות)

נתונה לכם המחלקה BinarySearchTree בשלמותה. אין לשנות בה דבר. קראו היטב את הקוד שבקובץ BinarySearchTree.java. עליכם להכיר את כל פרטי המחלקה, את השדות, הבנאים, והשיטות שלה.

המחלקה public class BinarySearchTree extends BinaryTree implements Iterable{…} יורשת את המחלקה BinaryTree וממשת את הממשק Iterable.

במחלקה שדה יחיד

Comparator treeComparator

בעזרתו המידע בעץ נשמר ממויין ומסודר על פי ה- Comparator המתקבל בעת יצירת העץ.

למחלקה שני בנאים:

- public BinarySearchTree(Comparator myComparator)

בנאי זה מקבל כפרמטר Comparator ובונה עץ חיפוש ריק.

- public BinarySearchTree(BinarySearchTree otherTree)

בנאי זה הוא הבנאי המעתיק. הוא מקבל כפרמטר עץ חיפוש אחר otherTree ובונה עץ חיפוש <u>שלם</u> המכיל את אותו המידע כמו בעץ otherTree (מכיל בדיוק את אותן ההפניות השמורות בקודקודי העץ otherTree). בקוד השיטה ישנה קריאה לבנאי המעתיק של המחלקה BinarySearchNode המתוארת בהמשך.

לאחר שתשלימו את הגדרת הבנאי המעתיק במחלקה BinarySearchNode ולאחר שתשלימו את משימה 4 הקוד בקובץ TestBalance.java ידפיס למסך את הפלטים הבאים:

```
----------t1:----------
    1
  2
    3
4
    5
  6
    7
      8

----------t2:----------
    1
  2
      3
    4
5
  6
    7
      8

----------t3:----------
  1
2
  3
    4
      5
        6
          7
            8

----------t4:----------
              1
            2
          3
        4
      5
    6
  7
8

----------b1:----------
      1
    2
  3
    4
5
    6
  7
    8

----------b2:----------
      1
    2
  3
    4
5
    6
  7
    8

----------b3:----------
      1
    2
  3
    4
5
    6
  7
    8

----------b4:----------
      1
    2
  3
    4
5
    6
  7
    8
```

<u>נתונות השיטות הבאות:</u>

- public Object findData(Object element)

כאשר נחפש איבר בעץ חיפוש בינארי, נעשה זאת בעזרת ה-Comparator. יתכן שהאיבר שנחפש לא יהיה זהה לזה שנמצא בעץ (שדה ה-data שבאחד הקודקודים של העץ) אך יהיה שווה לו לפי ה-Comparator.

שיטה זו מקבלת אובייקט element. השיטה מחפשת ומחזירה את ה- data השווה ל- element (על פי ה-Comparator) הנמצא בעץ המפעיל את השיטה, במידה וקיים. במידה ולא קיים בעץ קודקוד עם שדה data השווה ל-element (על פי ה- Comparator), השיטה מחזירה ערך null.

דוגמאות:

1. בקריאה לשיטה זו כאשר ה- Comparator של העץ הוא מטיפוס EntryComparatorByName, element מפנה אל הרשומה <"Dan", 1> , והעץ מכיל את הרשומה <"Dan", 86471234>, תוחזר הפניה לאותה הרשומה <"Dan", 86471234> .
2. בקריאה לשיטה זו כאשר ה- Comparator של העץ הוא מטיפוס EntryComparatorByNumber, element מפנה אל הרשומה <"dummy", 86471234> , והעץ מכיל את הרשומה <Dan, 86471234>, תוחזר הפניה לאותה הרשומה <Dan, 86471234> .

- public Comparator getComparator()

שיטה זו מחזירה את ה- Comparator של העץ.

- public void insert(Object toInsert)

שיטה זו מקבלת אובייקט toInsert ומכניסה אותו לעץ. זיכרו כי בספר הטלפונים שנממש שמות ומספרי טלפונים צריכים להיות יחודיים. לא ייתכנו שתי רשומות עם אותו השם וגם לא ייתכנו שתי רשומות עם אותו מספר הטלפון. במידה ו- toInsert מתנגש עם דרישה זו השיטה לא תשנה את העץ.

- public void remove(Object toRemove)

שיטה זו מקבלת אובייקט toRemove ומסירה אותו מהעץ, במידה והוא קיים בו.

- public Iterator iterator()

שיטה זו מחזירה Iterator של העץ מטיפוס BinaryTreeInOrderIterator המתואר בהמשך.

משימה 5ב:

המחלקה public class BinarySearchNode extends BinaryNode {…} יורשת את המחלקה BinaryNode.

במחלקה שדה יחיד

Comparator treeComparator

בעזרתו המידע בעץ נשמר ממויין ומסודר על פי טיפוס ה- Comparator המתקבל בעת יצירת קודקוד.

למחלקה שני בנאים:

- public BinarySearchNode(Object data, Comparator myComparator)
  בנאי זה מקבל אוביקט data ו- Comparator ובונה קודקוד חיפוש.
- public BinarySearchNode(BinarySearchNode otherTreeRoot, Iterator otherTreeIterator)

בנאי זה הוא הבנאי המעתיק. השלימו את הגדרתו בקובץ BinarySearchNode.java. הבנאי מקבל קודקוד חיפוש בינארי otherTreeRoot המהווה שורש של עץ חיפוש אחר, נסמנו t1, ואיטראטור otherTreeIterator של t1. הבנאי יוצר עץ חיפוש שלם t2 המכיל את אותו המידע כמו בעץ t1. בקוד השיטה ישנה קריאה לשתי שיטות עזר פרטיות אותן יהיה עליכם להשלים כמתואר בהמשך.

תהליך בניית העץ בבנאי המעתיק מתבצע כך:

1. ראשית יש קריאה לשיטה הפרטית buildPerfectTree(otherTreeRoot.size()) נבנה עץ שלם t2 עם מספר קודקודים השווה ל-otherTreeRoot.size(). שיטה זו דואגת לכל ששדה ה-data בכל קודקוד בעץ t2 יפנה למחרוזת "dummy".
2. לאחר מכן יש קריאה לשיטה הפרטית fillTheNodes(this, otherTreeIterator). קריאה זו מציבה את תוכן קוקודי העץ t1 (תת העץ המושרש ב- otherTreeRoot) בקודקודי העץ השלם t2. הצבה זו נעשית בעזרת otherTreeIterator של העץ t1 כך שסריקות ה-inorder של שני העצים תיהינה זהות.

נתונות השיטות הבאות (אין לשנות את הגדרתן):

- public Object findData(Object element)
  שיטה זו מקבלת אובייקט element מחפשת ומחזירה את ה- data השווה ל- element (על פי ה- Comparator) הנמצא בתת העץ המושרש בקודקוד המפעיל את השיטה, במידה וקיים. במידה ו- element לא קיים בתת עץ זה על השיטה להחזיר את הערך null. ראו דוגמאות לשיטה findData במחלקה BinarySearchTree.

- public Object findMin()
  השיטה מחזירה את שדה ה- data של הקודקוד המכיל את ה- data ה"קטן ביותר" בתת העץ המושרש בקודקוד המפעיל את השיטה.

עליכם להשלים את השיטות הבאות במחלקה:

- private void buildPerfectTree(int size) **(10 נקודות)**
  שיטה זו בונה עץ בינארי שלם עם מספר קודקודים השוה ל- size אשר השורש שלו הוא this – הוא הקודקוד המפעיל את השיטה.

הדרכת חובה (ראו מימוש חלקי בקובץ המחלקה): בעץ זה שדה ה- data של כל הקודקודים מפנה אל המחרוזת "dummy". הבנייה משתמשת בתור. יש ליצור תור חדש ולהוסיף לראש התור את (הקודקוד) this. כעת העץ מכיל קודקוד יחיד. בקטע הקוד שתשלימו הקודקודים יתווספו לעץ באופן הבא.

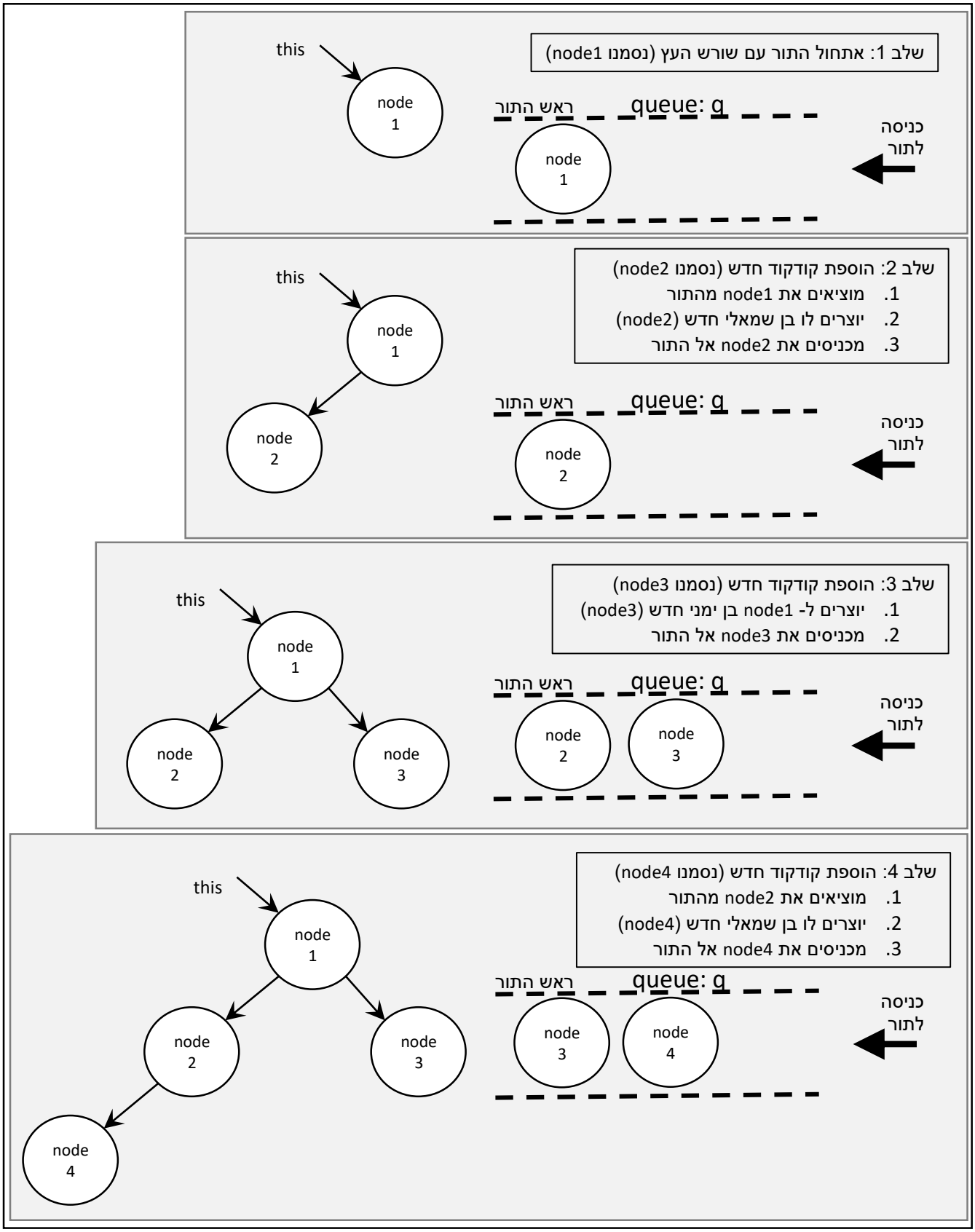
נסמן ב- n את מספר הקודקודים בעץ בכל רגע נתון.

על התהליך הבא נחזור כל עוד n<size:

אם n<size מוציאים קודקוד מהתור (נקרא לו nextNode) יוצרים לו בן שמאלי (חדש, נקרא לו leftDummyNode) ומכניסים את leftDummyNode לסוף התור. כעת מספר הקודקודים בעץ גדל באחד (n=n+1). אם n<size יוצרים לקודקוד nextNode בן ימני (חדש, נקרא לו rightDummyNode) ומכניסים גם את rightDummyNode לסוף התור.

בסוף התהליך הזה הקודקוד this הוא השורש של עץ שלם עם size קודקודים.

האיור הבא ממחיש בארבעה שלבים את פעולת השיטה buildPerfectTree(size) כאשר size=4:



- private void fillTheNodes(BinarySearchNode root, Iterator treeIterator) **(10 נקודות)**

שיטה זו מקבלת עץ חיפוש בינארי המושרש בקודקוד root (נסמנו t1) ואיטראטור treeIterator של t1. בזמן הקריאה לשיטה זו, this פונה לשורש של עץ בינארי שלם שכל קודקודיו מכילים את המידע "dummy", נסמנו t2. בשני העצים אותו מספר קודקודים. השיטה מבצעת סריקת inorder ב-t2 ובכל קודקוד לפי סדר הסריקה מציבה בשדה ה-data את האובייקט המתקבל מ treeIterator.next() שהוא האיבר הבא בעץ t1 לפי סריקת inorder . לאחר סיום הסריקה העץ t2 מכיל את אותו המידע שבעץ t1 ולשני העצים סריקת inorder זהה.

- public Comparator getComparator() **(5 נקודות)**

שיטה זו מחזירה את ה- Comparator של העץ.

- public void insert(Object toInsert) **(10 נקודות)**

שיטה זו מקבלת אובייקט toInsert ומכניסה אותו לקודקוד חדש במקום המתאים לו בתת העץ המושרש בקודקוד המפעיל את השיטה. זיכרו כי אם תת העץ המושרש בקודקוד מכיל את toInsert אז אובייקט זה לא ייכנס לעץ.

- public boolean contains(Object element) **(5 נקודות)**

שיטה זו מקבלת אובייקט element ומחזירה true אם תת העץ המושרש בקודקוד המפעיל את השיטה מכיל את element.

- public BinaryNode remove(Object toRemove) **(10 נקודות)**

שיטה זו מקבלת אובייקט toRemove ומסירה אותו מהעץ המושרש בקודקוד המפעיל את השיטה, במידה והוא קיים בעץ (על פי ה- Comparator של העץ). השיטה מחזירה מצביע לשורש העץ המושרש בקודקוד המפעיל את השיטה לאחר ההסרה.

משימה 5ב: (15 נקודות)

נתונה לכם המחלקה BinaryTreeInOrderIterator המממשת את הממשק Iterator של java. שימו לב כי בקובץ המחלקה מופיעה השורה import java.util.Iterator;.

איטראטור זה עובר על המידע השמור בעץ החיפוש לפי סדר inorder. בעת כתיבת השיטות במחלקה זו מומלץ להיעזר בשיטת עזר פרטית.

במחלקה שדה יחיד (אין להוסיף שדות נוספים)

Stack stack;

עליכם להשלים את השיטות הבאות במחלקה:

- public BinaryTreeInOrderIterator(BinaryNode root)

בנאי האיטראטור מקבל כפרמטר קודקוד המהווה שורש של עץ חיפוש בינארי ומאתחל את שדה המחלקה.

- public boolean hasNext()
- public Object next()

השיטות hasNext, next הן השיטות המפורטות בממשק Iterator המובנה ב-java.

משימה 6: מערכת ספר הטלפונים (15 נקודות)

במשימה זו תשלימו את הגדרת המחלקה Phonebook בקובץ Phonebook.java. למחלקה שני שדות

```
private BinarySearchTree namesTree;
private BinarySearchTree numbersTree;
```

שהינם עצי חיפוש בינארי. עצים אלו מכילים את אוסף הרשומות (מסוג PhoneEntry) הקיים בספר הטלפונים. בעץ הראשון הרשומות ממויינות לפי שמות ובעץ השני לפי מספרי טלפון. נדגיש כי כל רשומה קיימת בספר הטלפונים רק פעם אחת, ובכל עץ קיים לה קודקוד ובו שדה Object data המפנה אליה.

בנאי המחלקה public Phonebook() מגדיר ספר טלפונים ריק (עם שני עצי חיפוש ריקים).

נתונות השיטות הבאות (אין לשנות את הגדרתן):

- public PhoneEntry lookUp(String name)

שיטה זו מקבלת שם name ומחזירה את הרשומה בספר הטלפונים עם השם name במידה וקיימת כזו. אחרת השיטה תחזיר את הערך null.

- public PhoneEntry lookUp(int number)

שיטה זו מקבלת מספר number ומחזירה את הרשומה בספר הטלפונים עם המספר number במידה וקיימת כזו. אחרת השיטה תחזיר את הערך null.

- public void balance()

שיטה זו מיודעת לשמירה על יעילות השימוש בספר הטלפונים.
שיטה זו בונה מחדש את שני עצי החיפוש כך שתכולתם תישאר זהה אך מבנה העץ יהיה מבנה של עץ שלם. פעולה זו מתבצעת על ידי שתי קריאות לבנאי המעתיק של המחלקה BinarySearchTree.

עליכם להשלים את השיטות הבאות במחלקה:

- public boolean add(PhoneEntry newEntry)

שיטה זו מקבלת רשומה חדשה newEntry ומוסיפה אותה לספר הטלפונים במידה והרשומה עונה על התנאים הבאים:

- אין בספר הטלפונים רשומה קיימת עם אותו השם שב-newEntry .
- אין בספר הטלפונים רשומה קיימת עם אותו המספר שב-newEntry .

הפונקציה מחזירה true אם ההוספה התבצעה בהצלחה ומחזירה false אחרת.
יש להוסיף את אותה הרשומה לשני העצים המוגדרים בשדות המחלקה.

- public boolean delete(String name)

שיטה זו מקבלת שם name ומוחקת את הרשומה בספר הטלפונים עם השם name במידה וקיימת כזו. זיכרו כי במידה והרשומה קיימת יש להסיר את ההפניה אליה משני העצים. השיטה מחזירה true אם התבצעה מחיקה ו- false אחרת.

- public boolean delete(int number)

שיטה זו מקבלת מספר number ומוחקת את הרשומה בספר הטלפונים עם המספר number במידה וקיימת כזו. זיכרו כי במידה והרשומה קיימת יש להסיר את ההפניה אליה משני העצים. השיטה מחזירה true אם התבצעה מחיקה ו- false אחרת.

בהצלחה!