

# מעבדה 7. נושא: עץ חיפוש בינארי

תאריך הגשה: 04/07/2024 בשעה 23:00 (בזוגות)

## יש לקרוא היטב לפני תחילת העבודה !

### מבוא:

במעבדה הנוכחית נממש עצי חיפוש בינאריים.

### רקע:

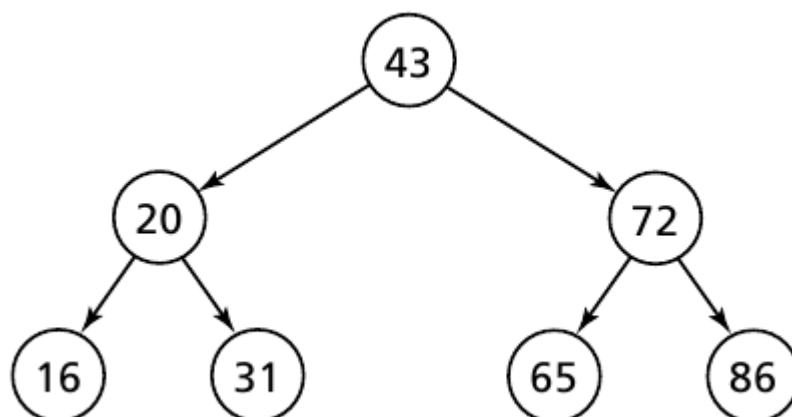
#### הגדרה 1 (עץ בינארי):

עץ שלכל אחד מהצמתים הפנימיים שלו לכל היותר שני בנים, נקרא עץ בינארי.

#### הגדרה 2 (עץ חיפוש בינארי):

עץ בינארי בו מפתחות של כל הצמתים הנמצאים בתת-עץ שמאלי קטנים ממפתח של שורש העץ וכל המפתחות של כל הצמתים הנמצאים בתת-עץ ימני גדולים ממפתח של שורש העץ, נקרא עץ חיפוש בינארי.

דוגמא לעץ חיפוש בינארי:



הממשק הגנרי Comparable (ממשק עם שיטה אחת בלבד)

<https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html>

### תאור התרגיל:

(1) כתבו מחלקה **Person** המכילה מידע על אנשים.

המחלקה תשב בחבילה: **package** il.ac.telhai.ds.misc

שדות המחלקה:

```
String id;  
String firstName;
```

String lastName;

(2) הוסיפו למחלקה בנאי המקבל ערכים לשלש השדות, והוסיפו getters ו setters לשדות, מלבד id-ל set.

(3) בנוסף, המחלקה Person תממש את הממשק Comparable<Person>, כאשר הסדר נקבע לקסיקוגרפית לפי id בלבד. שני מופעים של Person בעלי id זהים - יחשבו לאותו ה-Person. יש לשים לב שההשוואה ב-Person היא לפי id כמחזורת (סדר מילוני) ולא כמספר, למשל "111" קטן מ- "20" למרות ש- 20 קטן מ- 111.

(4) הוסיפו מתודות hashCode, equals כפי שניתן להוסיף באופן אוטומטי ב-eclipse (לחצו על source → generate hashCode and equals) ובדומה ב-intlij, לפי השוואת id בלבד. (שאלה: מדוע חשוב להוסיף מתודות אלו?)

(5) הוסיפו מתודת toString ל-Person.

"Person [id=" + #id + ", firstName=" + #firstName + ", lastName=" + #lastName + "]"  
כאשר במקום #field רושמים את ערך השדה field.  
ניתן להשתמש באפשרות האוטומטית של סביבת העבודה (ב-eclipse לחצו על source → generate toString).

(6) השלימו את המחלקה **BinarySearchTree** הנתונה לכם.

מחלקה זו נמצאת בחבילה: **package** il.ac.telhai.ds.trees

מחלקה זו מממשת עץ חיפוש, כאשר ההשוואה של האיברים מתבצעת לפי compareTo.  
בעץ החיפוש כל הערכים שונים.

הערה: יש לממש את המתודה size בזמן ריצה  $O(1)$ .

## סדר העבודה ופרטים טכניים

- שליפת הפרויקט DS-Lab07-SearchTree מתוך GITHUB בקישור:

<https://github.com/michalHorovitz/DSLlab2024Public>

○ אם אין לכם גישה לפרויקט שהורדתם מ GITHUB במעבדה הראשונה יש לבצע שליפה מחדש לפי ההוראות במעבדה הראשונה.

○ אם יש לכם גישה לפרויקט שהורדתם מ GITHUB במעבדה הראשונה אז בצעו:

■ קליק על שם הפרויקט.

■ עכבר ימני

■ Team-->Pull

■ File-->Import->Git->Projects From Git->Existing Local Repository

אם אתם עובדים ב VDI, מומלץ לשנות את המיקום המוצע לפרויקט בתיקייה כלשהי בכוון H.

### פורמט קובץ ההגשה ובדיקתו:

**פורמט:** יש להגיש קובץ ZIP בשם

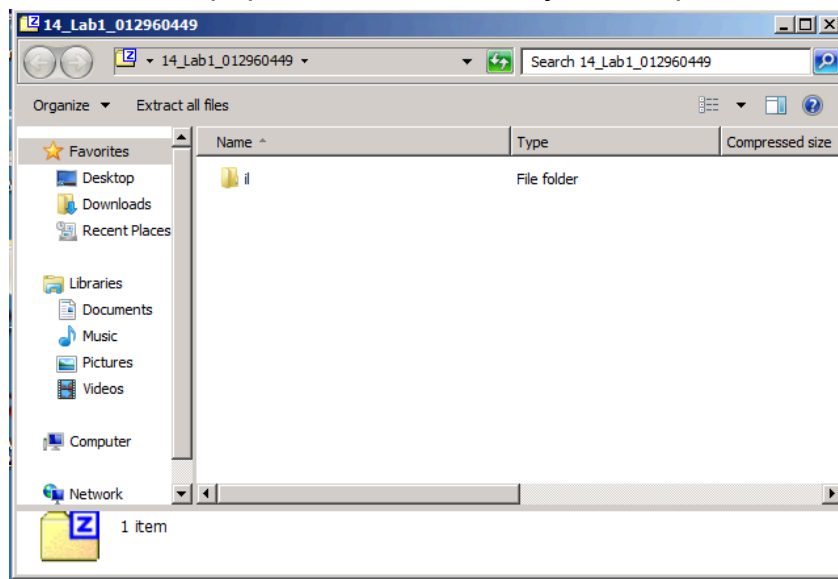
51\_lab07\_123456789\_987654321.zip

(כמובן, יש להחליף את המספרים עם מספרי ת.ז. של המגישים).

על הקובץ להכיל את כל קבצי ה JAVA שכתבתם כאשר הם נמצאים בתיקייה

il/ac/telhai/ds/trees

כלומר, השורש של קובץ ההגשה יכול רק תיקייה בשם il.  
ומכיל את כל קבצי - java . להמחשה תמונה של קובץ כזה שנפתח ב - WindowsExplorer



**בדיקת קובץ ההגשה:** בדקו את הקובץ שיצרתם בתוכנת הבדיקה בקישור:

<https://csweb.telhai.ac.il/>

ראו [סרטון הדגמה](#) של השימוש בתוכנת הבדיקה.

**חשוב !!!**

בדיקת ההגשות תבוצע ברובה ע"י תוכנית הבדיקה האוטומטית הנ"ל. תוצאת הבדיקה תהייה בעיקרון זהה לתוצאת הבדיקה הנ"ל שאתם אמורים לערוך בעצמכם. כלומר, אם ביצעתם את הבדיקה באתר החוג, לא תקבלו הפתעות בדיעבד. אחרת, ייתכן שתרגיל שעבדתם עליו קשה ייפסל בגלל פורמט הגשה שגוי וכו'. דבר שהיה ניתן לתקנו בקלות אם הייתם מבצעים את הבדיקה. היות ואין הפתעות בדיעבד, לא תינתן אפשרות של תיקונים, הגשות חוזרות וכד'.

הגשה שלא מגיעה לשלב הקומפילציה תקבל ציון 0.

הגשה שלא שמתקמפלת תקבל ציון נמוך מ- 40 לפי סוג הבעיה.

הגשה שמתקמפלת תקבל ציון 40 ומעלה בהתאם לתוצאות הריצה, ותוצאת הבדיקה הידנית של הקוד (חוץ ממקרה של העתקה).

תכנית הבדיקה האוטומטית מכילה תוכנה חכמה המגלה העתקות. מקרים של העתקות יטופלו

**בחומרה**