הטכניון – מכון טכנולוגי לישראל
הפקולטה למדעי הנתונים וההחלטות
הנדסת תוכנה אביב תשפ"ג

הנדסת תוכנה – תרגיל בית 4

דגשים להגשת המטלה

1. תאריך הגשה: יום חמישי 06.07.2023 בשעה 23:59.
2. **הגשה בזוגות בלבד!**
3. הקוד חייב להיכתב בהתאם למוסכמות כתיבת הקוד בקורס כולל תיעוד כנדרש. קוד שלא עומד בדרישות יגרור הורדת ניקוד. ניתן למצוא את קובץ מוסכמות הקידוד באתר הקורס תחת הלשונית "קבצי עזר".
4. ההגשה מתבצעת ב-Moodle באזור המיועד על ידי אחד מהשותפים, לאחר יצירת קבוצה.
5. ניתן להגיש את התרגיל לכל היותר עד 48 שעות לאחר מועד ההגשה ללא הורדת ניקוד. לאחר 48 שעות תיבת ההגשה תיסגר ולא יהיה ניתן להגיש את התרגיל כלל.
6. פורמט הגשת התרגיל נמצא בקובץ ההנחיות ב-Moodle. **כל חריגה מפורמט זה תגרור ציון אפס.**

מטרת התרגיל

עבודה עם רקורסיה, עצים בינאריים ותכנות מקבילי.

הכנות טרם תחילת התרגיל

1. פתיחת פרויקט ג'אווה חדש.
2. הורדת קבצי התרגיל, והעתקתם אל תוך תיקיית ה-src **פרט** לקובץ הפלט.

הוראות כלליות

1. מומלץ להריץ את התוכנית עם מספר קלטים שונים ולחשוב על מקרי קצה אפשריים.
2. מומלץ לחזור על התרגולים וההרצאות וכן להיעזר באינטרנט.
3. יש להשתמש בגרסה 9.0.4 של ג'אווה בעת פתרון התרגיל.
4. מומלץ להשתמש ב-Git במהלך כתיבת התרגיל.

הוראות הגשה

1. יש למלא אחר הוראות ההגשה בהתאם למסמך הדרישות "הנחיות כלליות לפתרון והגשת תרגילי הבית" אשר מופיע באתר הקורס.
2. הגשה אלקטרונית **בלבד** דרך אתר הקורס ב-moodle. ההגשה תכלול קובץ ה-zip בלבד בפורמט HW4_<id1>_<id2> כאשר <id1> ו-<id2> הם תעודות הזהות של המגישים. על הקובץ לכלול תיקייה בודדת בשם src בה יימצאו כל קבצי הקוד.
3. ההגשה מתבצעת על ידי אחד מבני הזוג לאחר שיצר קבוצה ובן הזוג השני הצטרף אליה.
4. תרגיל בית שלא יוגש על פי הוראות ההגשה – **לא ייבדק**.
5. יש להקפיד על יושרת הכנת התרגיל וההגשה.
6. יש לוודא כי הקוד מתקמפל – קוד אשר לא יעבור הידור יקבל ציון אפס.
7. אין צורך להגיש את קובץ הפלט אשר ניתן כחלק מתרגיל זה.

חלק א – רקורסיה ועצים בינאריים

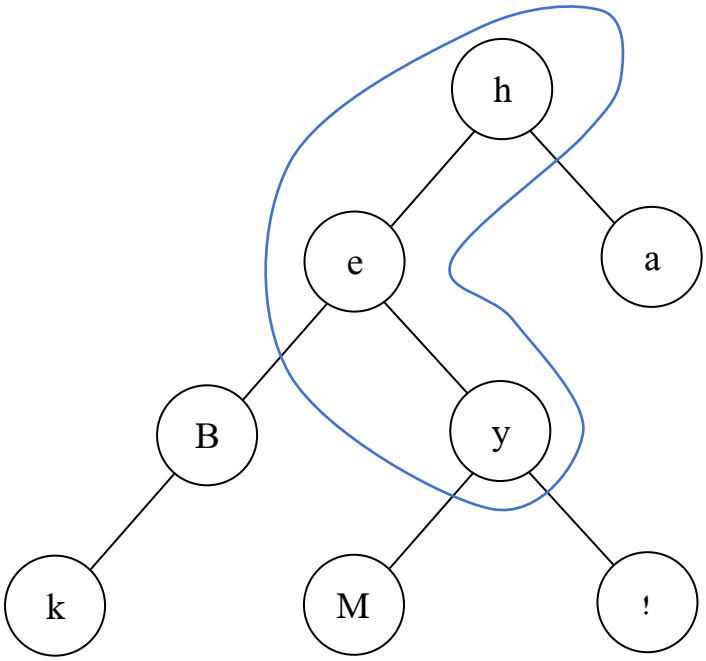
בקבצים שקיבלתם כחלק מן התרגיל מופיעה המחלקה הגנרית BinNode אותה פגשנו בתרגול. להזכירכם, מחלקה זו מייצגת צומת בעץ בינארי, והיא תשמש אתכם לשתי השאלות בחלק זה של התרגיל. **אין לשנות את תוכן המחלקה.**

שאלה 1

בקבצים מוגדרת מחלקה בשם PathFromRoot בה מוגדרת פעולה סטטית בודדת בשם doesPathExist. פעולה זו מקבלת שני פרמטרים: שורש של עץ בינארי של תווים ומחרוזת. על הפעולה למצוא האם קיים מסלול המתחיל בשורש העץ שבו רצף התווים זהה למחרוזת שהתקבלה. במידה וכן, הפעולה תחזיר אמת, ואחרת תחזיר שקר.

במידה והמחרוזת שהתקבלה ריקה, על הפעולה להחזיר אמת, שכן קיים מסלול באורך אפס.

לדוגמה, התבוננו בעץ הבא:



במידה ונעביר לפעולה את עץ זה ואת המחרוזת "hey" יוחזר אמת, שכן המסלול שמסומן בכחול מקיים את התנאי.

במידה ונעביר לפעולה את עץ זה ואת המחרוזת "Hey" יוחזר שקר, שכן לא קיים מסלול שמקיים את התנאי.

במידה ונעביר לפעולה את עץ זה ואת המחרוזת "hex" יוחזר שקר, שכן לא קיים מסלול שמקיים את התנאי.

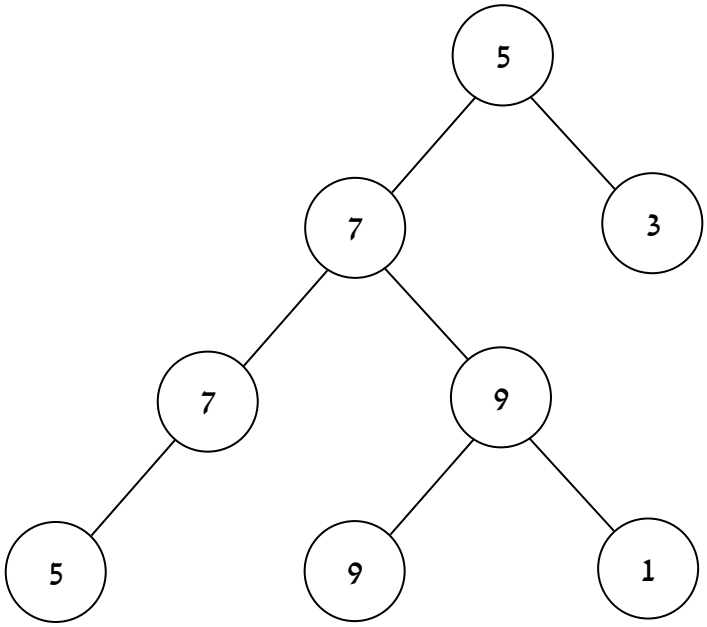
הנחיות לשאלה

- בשאלה זו יש להשתמש ברקורסיה.
- אין להשתמש בלולאות כלל.
- **לא** ניתן להגדיר פעולות עזר, משתנים סטטיים במחלקה, או מחלקות נוספות.
- אין לייבא אף מחלקה ואין להשתמש באף פעולה חיצונית, גם לא בפעולות שמוגדרות במחלקות אשר אינן דורשות ייבוא, פרט לפעולות המוגדרות במחלקה BinNode ובמחלקה String.

שאלה 2

בקבצים שקיבלתם מוגדרת מחלקה בשם `LevelLargestSum` בה מוגדרת פעולה סטטית בודדת בשם `getLevelWithLargestSum`. פעולה זו מקבלת שורש של עץ בינארי של מספרים שלמים, מוצאת את הרמה בעלת הסכום הגבוה ביותר ומחזירה אותה. במידה ויש מספר רמות שסכומן הוא הגבוה ביותר, יש להחזיר את מספרה של הרמה הראשונה מבין אותן רמות. במידה והעץ ריק, יש להחזיר 1-.

לדוגמה, התבוננו בעץ הבא:



במידה ונעביר לפעולה את עץ זה יוחזר 2, מאחר וסכום האיברים הנמצאים ברמה 2 הוא 16, וסכום זה גדול מן הסכום של כל רמה אחרת.

הנחיות לשאלה

- בשאלה זו אין חובה להשתמש ברקורסיה, ומותר להשתמש בלולאות.
- בשאלה זו **לא** ניתן להגדיר פעולות עזר, משתנים סטטיים ומחלקות נוספות.
- אין לייבא אף מחלקה ואין להשתמש באף פעולה חיצונית, גם לא בפעולות שמוגדרות במחלקות אשר אינן דורשות ייבוא, פרט לפעולות המוגדרות במחלקה `BinNode` ובמחלקה `ArrayDeque`.

חלק ב – תכנות מקבילי

בחלק זה של התרגיל נדמה גישה מקבילית למסד נתונים. מסד הנתונים מכיל בתוכו נתונים, אותם ניתן לקרוא. כמו כן, ניתן להוסיף ולשנות נתונים קיימים.

תהליכונים רבים יכולים להשתמש באותו מסד נתונים, אך כאשר תהליכון אחד כותב למסד הנתונים, אף תהליכון אחר לא יכול לגשת אליו, לא לצורך קריאה ולא לצורך כתיבה, ויהיה עליו לחכות עד אשר התהליכון הכותב יסיים את משימתו.

לעומת זאת, תהליכונים שונים יכולים לקרוא נתונים ממסד הנתונים במקביל. עם זאת, על מנת שלא ליצור עומס על מסד הנתונים, ישנה הגבלה על מספר התהליכונים שיכולים לקרוא ממסד הנתונים באותו הזמן, אותו נסמן ב-k. כל עוד מסד הנתונים נמצא בשימוש על ידי פחות מ-k קוראים, קוראים נוספים יכולים לגשת אליו. במידה וישנם k תהליכונים הקוראים ממסד הנתונים, כל קורא נוסף ייאלץ לחכות עד אשר קורא אחר המשתמש במסד הנתונים יסיים את משימתו.

מימוש המשימה

לצורך מימוש המשימה, הוגדרה בעבורכם מחלקה בשם `Database` המדמה את מסד הנתונים. המחלקה מכילה תכונה מטיפוס `HashMap`, שבה נשמרים נתוני המסד. בנאי המחלקה מקבל את מספר התהליכונים המקסימלי שיכולים לקרוא ממסד הנתונים במקביל. נוסף על כך, במחלקה ממומשת פעולה בשם `put` המדמה כתיבה למסד הנתונים, וכן פעולה בשם `get` המדמה קריאה ממנו.

בנוסף, במחלקה מוגדרות מספר פעולות נוספות, אותן עליכם לממש:

- פעולה בשם `readAcquire` אשר נמצאת בשימוש לפני קריאה ממסד הנתונים. כפי שצוין, במידה ותהליכון יקרא לפעולה זו כאשר תהליכון אחר מבצע כתיבה למסד הנתונים, או ש-k תהליכונים אחרים קוראים ממנו, התהליכון יחכה עד שיוכל לקרוא.
- פעולה בוליאנית בשם `readTryAcquire` אשר בדומה ל-`readAcquire`, גם בה ניתן להשתמש לפני קריאה ממסד הנתונים. פעולה זו זהה לפעולה `readAcquire`, אך אינה חוסמת את התהליכון עד שיוכל לקרוא מן המסד, אלא מחזירה אמת אם התהליכון יכול להתחיל את הקריאה ושקר במידה והוא אינו יכול.
- פעולה בשם `readRelease` אשר נמצאת בשימוש אחרי הקריאה ממסד הנתונים. הקריאה לפעולה מסמנת שהתהליכון סיים לקרוא מן המסד. במידה ותהליכון משתמש בפעולה זו אך הוא אינו קורא כרגע ממסד הנתונים, יש לזרוק חריגה מטיפוס `IllegalMonitorStateException` עם ההודעה "Illegal read release attempt". חריגה זו הינה חריגה בלתי מסומנת המוגדרת כבר בג'אווה.
- פעולה בשם `writeAcquire` אשר נמצאת בשימוש לפני כתיבה למסד הנתונים. כפי שצוין, במידה ותהליכון יקרא לפעולה זו כאשר תהליכון אחר מבצע כתיבה למסד הנתונים, או שישנו תהליכון הקורא ממנו, התהליכון יחכה עד שיוכל לכתוב.
- פעולה בוליאנית בשם `writeTryAcquire` אשר בדומה ל-`writeAcquire`, גם בה ניתן להשתמש לפני כתיבה למסד הנתונים. פעולה זו זהה לפעולה `writeAcquire`, אך אינה חוסמת את התהליכון עד שיוכל לכתוב למסד, אלא מחזירה אמת אם התהליכון יכול להתחיל את הכתיבה ושקר במידה והוא אינו יכול.
- פעולה בשם `writeRelease` אשר נמצאת בשימוש אחרי הכתיבה למסד הנתונים. הקריאה לפעולה מסמנת שהתהליכון סיים לכתוב למסד. במידה ותהליכון משתמש בפעולה זו אך הוא אינו התהליכון שכותב כרגע למסד הנתונים, יש לזרוק חריגה מטיפוס `IllegalMonitorStateException` עם ההודעה "Illegal write release attempt".

הנחיות לשאלה

- לצורך הפתרון ניתן להיעזר **אך ורק** במחלקות ובממשקים הבאים:
  - `Thread`.
  - `Lock + ReentrantLock`.
  - `Map + HashMap`.
  - `Set + HashSet`.
  - `IllegalMonitorStateException`.
  - `Condition`.
  - `InterruptedException`.
- ניתן להיעזר במילה השמורה `synchronized`, וכן בפעולות הקשורות לסנכרון שמוגדרות במחלקה `Object` (`wait, notify, notifyAll`).
- שימו לב כי **אין בהכרח צורך** להשתמש בכל מה שמותר.
- ניתן להגדיר במחלקה `Database` פעולות עזר וכן ניתן להגדיר תכונות נוספות ולאתחל אותן בבנאי. עם זאת, **אין לשנות את כותרת הבנאי ואת הפרמטרים אותם הוא מקבל**.
- אין לשנות את פעולות ה-`put` וה-`get` המוגדרות במחלקה וכן אין לשנות את כותרות הפעולות האחרות.
- יש לממש את פעולת ה-`readAcquire` וה-`writeAcquire` באופן שבו יימנעו כמה שניתן מצבים של Busy Waiting – מצבים בהם תהליכון אינו מבצע דבר עד אשר יתקיים תנאי כלשהו ולאחר מכן התהליכון ימשיך לבצע את הקוד שנותר לו.
- כחלק מהבדיקה של מחלקת ה-`Database`, מוגדרות בקובץ Main.java מספר מחלקות (`Worker, Reader, Writer`). מחלקות אלו נוצרו אך ורק על מנת שתוכלו לבדוק את הקוד שלכם, וייתכן מאוד כי במהלך בדיקת התרגיל מחלקות אלו יוחלפו במחלקות אחרות. על כן, בקוד אותו אתם כותבים **אין להשתמש כלל** במחלקות אלו.

הכוונות לשאלה

- חישבו כיצד תוכלו לדעת כמה תהליכונים קוראים ברגע זה ממסד הנתונים (אם יש כאלו) ומי הם, והאם תהליכון כותב כרגע למסד (אם יש כזה) ומי הוא.
- חישבו כיצד תוכלו לגרום לתהליכונים להמתין כל עוד הם לא יכולים לבצע את משימתם.
- מומלץ לקרוא על המחלקה `Thread` ועל הפעולות הקיימות בה.
- המלצה: וודאו שמנגנון הסנכרון עובד לפני שתיגשו לממש את החלקים בו הקשורים לזריקת החריגות בפעולות ה-`release`.

הנחיות לפתרון

- במהלך כתיבת הקוד, יש לשמור על עקרונות תכנות נכונים, כפי שנלמד בקורס.
- בעת פתרון התרגיל ניתן ואף מומלץ להגדיר קבועים ולא להשתמש במספרי קסם.
- במהלך פתרון התרגיל, יש להקפיד על הרשאות הגישה השונות בהתאם לנלמד בקורס.
- יש להקפיד על מתן שמות משמעותיים למשתנים ולפעולות.
- יש לוודא כי כל המחלקות הן פומביות וכי כל מחלקה מופיעה בקובץ נפרד.
- יש לתעד את כל הפעולות והמחלקות אותן אתם מגדירים בעזרת שימוש ב-JavaDoc בהתאם לקובץ מוסכמות התיעוד אשר מופיע באתר הקורס. בנוסף, יש לתעד שורות קוד אשר עשויות להיות קשות להבנה. עבור פעולות שזורקות חריגה כלשהי יש לציין זאת **בתיעוד** על ידי פסקת `@throws` ביחד עם הסבר על מתי החריגה תיזרק (שימו לב כי פסקת `@throws` שונה מציון החריגה בכותרת הפעולה).

הרצת התוכנית וביצוע בדיקות

בקובץ Main.java קיימות מספר מחלקות אשר משמשות לבדיקת הקוד שכתבתם.

להזכירכם, חלק מן הבדיקה נעשה באופן אוטומטי, ולכן אין לשנות את התוכן של הקובץ, ובפרט אין לשנות את פעולות ההדפסה המתבצעות בו.

מצורף לתרגיל זה קובץ פלט בשם HW4_optional_output.txt. הפלט עבור חלקו הראשון של התרגיל הינו דטרמיניסטי, ותוכלו להשוות אותו אל מול הפלט המתקבל אצלכם. עם זאת, הפלט עבור חלקו השני של התרגיל אינו דטרמיניסטי בחלקו, והוא תלוי בסדר החלפות ההקשר המתבצעות בעת הרצת הקוד. אי לכך, הפלט שקיבלתם הינו רק **פלט אפשרי** עבור חלק זה.

בפעולה הבודקת את החלק השני של התרגיל נוצר מסד נתונים המאפשר קריאה במקביל של עד 10 תהליכונים. בנוסף, הפעולה מכילה לולאה המחולקת לשישה שלבים:

- בשלב הראשון מתבצעת בדיקה בה 11 תהליכונים מנסים לקרוא ממסד הנתונים. כאשר תהליכון מצליח לקרוא הוא מדפיס את המזהה שלו ביחד עם ההודעה "Is able to read!", ישן למשך מעט זמן, ומדפיס בשנית את המזהה שלו ביחד עם ההודעה "Done reading.". הפלט של בדיקה זו אינו דטרמיניסטי, ועליכם לוודא שהפלט שקיבלתם הוא תקין: תחילה 10 תהליכונים אמורים להצליח לקרוא, לאחר מכן לפחות אחד מהם אמור לסיים לקרוא ורק לאחר מכן תהליכון נוסף יוכל לקרוא מן המסד.
- בשלב השני מתבצעת בדיקה בה שני תהליכונים מנסים לכתוב למסד הנתונים. כאשר תהליכון מצליח לכתוב הוא מדפיס את המזהה שלו ביחד עם ההודעה "Is able to write!", ישן למשך מעט זמן, ומדפיס בשנית את המזהה שלו ביחד עם ההודעה "Done writing.". הפלט של בדיקה זו אינו דטרמיניסטי, ועליכם לוודא שהפלט שקיבלתם הוא תקין: קודם תהליכון אחד מתחיל ומסיים לכתוב, ורק לאחר מכן התהליכון השני מתחיל ומסיים.
- בשלב השלישי מתבצעת בדיקה בה תהליכון אחד מנסה לכתוב, ותהליכון אחר (שמתחיל מעט אחריו) מנסה לקרוא את הערך שהתהליכון הראשון כתב. כאשר התהליכון הכותב מצליח לכתוב הוא מדפיס את המזהה שלו ביחד עם ההודעה "Is able to write!", ישן למשך מעט זמן, ומדפיס בשנית את המזהה שלו ביחד עם ההודעה "Done writing.". כאשר התהליכון הקורא מצליח לקרוא הוא מדפיס את המזהה שלו ביחד עם ההודעה "Is able to read!", לאחר מכן מדפיס בשנית את המזהה שלו ביחד עם הערך שקרא, ישן למשך מעט זמן, ולבסוף מדפיס שוב את המזהה שלו ביחד עם ההודעה "Done reading.". הפלט של בדיקה זו הוא **כן** דטרמיניסטי.
- השלב הרביעי זהה לשלב הראשון, רק שהתהליכונים משתמשים בפעולת ה-`readTryAcquire` על מנת לקרוא ממסד הנתונים. במקרה זה, 10 תהליכונים אמורים להצליח לקרוא, ואילו אחד לא יצליח. הפלט של בדיקה זו אינו דטרמיניסטי.
- השלב החמישי זהה לשלב השני, רק שהתהליכונים משתמשים בפעולת ה-`writeTryAcquire` על מנת לכתוב למסד הנתונים. במקרה זה, תהליכון אחד אמור להצליח לכתוב, ואילו השני לא יצליח. הפלט של בדיקה זו אינו דטרמיניסטי.
- השלב השישי זהה לשלב השלישי, רק שהתהליכון הקורא משתמש בפעולת ה-`readTryAcquire` על מנת לקרוא ממסד הנתונים. במקרה זה, התהליכון הקורא לא יצליח לקרוא. הפלט של בדיקה זו הוא **כן** דטרמיניסטי.

שימו לב כי **לא ניתן להניח** שתמיד מסד הנתונים יוגבל ל-10 קוראים, שרק 11 תהליכונים ינסו לקרוא ממנו במקביל וכו'. הבדיקות המבצעות בקובץ שקיבלתם משמשות אך ורק על מנת שתוכלו לבדוק את הקוד שכתבתם על מקרים קטנים. בבדיקת התרגיל יבוצעו בדיקות של מקרים נוספים (גדולים ומורכבים יותר), ולכן מומלץ שגם אתם תבצעו בדיקות של מקרים נוספים.

בהצלחה