המחלקה למדעי המחשב, אוניברסיטת בן גוריון
מבני נתונים, סמסטר אביב 2018
עבודת בית מספר 2

נושאים: מערכים, רשימות

<u>תאריך הגשה:</u> 19.04.2018 23:59

<u>הוראות חשובות לפני שמתחילים</u>:

1. את העבודה יש להגיש בזוגות.
2. לתרגיל מצורפים קבצי Java, אחד מהם מכיל את פונקציית ה- main אותה אנו נריץ במהלך הבדיקה. עליכם להוסיף קוד שלכם בתוך הקבצים שסופקו. מותר להוסיף מחלקות ושדות נוספים.
3. את העבודה יש להגיש דרך ה- Submission system בקובץ ZIP יחיד ששמו הוא ת״ז של המגישים מופרדים במקף תחתי, לדוגמה 111111111_22222222.zip.

   תוכן קובץ ה-ZIP הינו כל הקבצים של העבודה (קבצי java בלבד) יחד עם קובץ PDF המכיל פירוט <u>מוקלד</u> של מבני הנתונים והצדקה לזמני הריצה הנדרשים. על קובץ הפירוט לא לעלות על שני עמודים (Calibri 11 עם גבולות סטנדרטים).
4. חובה להוסיף תיעוד לקוד שלכם.
5. שאלות הקשורות לעבודה ניתן לשאול בפורום או בשעות הקבלה של האחראים על העבודה.
6. קראו היטב את כל העבודה לפני שאתם מתחילים!
7. מותר להניח שהקלט הוא תקין.

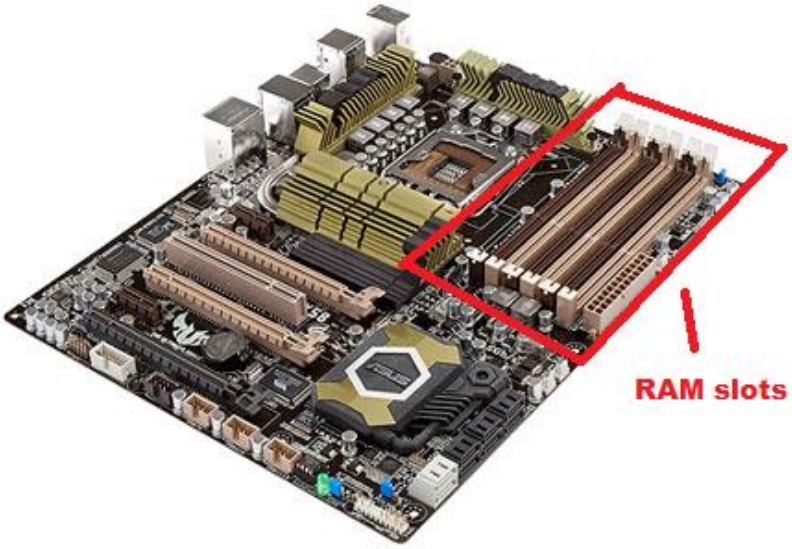
מבוא- מערכת לניהול זיכרון

זיכרון מחשב המכונה RAM (Random Access Memory) הוא איחוד של רכיבי מחשב, התקנים והתקני רישום המכילים נתונים דיגיטליים ומסייעים לחישובים בטווח זמן מסוים. שמירת נתוני מחשב היא אחת מהיכולות המרכזיות של מחשבים מודרניים המאפשרת שמירה על נתונים ויחד עם יכולת עיבוד הנתונים של המעבד היא מהווה את בסיס מודל המחשוב מאז שנות הארבעים של המאה ה-20.

Random Access memory פירושו שניתן לגשת לתאי זיכרון בסדר כלשהו.

ברמה הפשוטה ביותר ניתן לחלק את זיכרון המחשב לשני סוגים: זיכרון ראשי (main memory or primary memory) וזיכרון משני (secondary memory). הסיבה שיש שני סוגי זיכרון היא המהירות והקיבולת.

זיכרון ראשי מודרני מגיע בדרך כלל בצורת מעגל משולב על כרטיסים - המתחברים ללוח האם (ראו ציור למטה).



תכונות של זיכרון ראשי

- מחובר קרוב מאוד למעבד, לכן העברת פקודות ונתונים מהזיכרון הראשי למעבד מתבצעות בצורה מאוד מהירה
- המידע המאוחסן ניתן לעדכן בקלות ומהירות
- מאחסן תוכניות ונתונים שהמעבד כרגע משתמש בהם
- מבצע אינטראקציה עם המעבד מיליון פעמים בשנייה
- מצריך חיבור לחשמל כדי לשמור מידע

שום דבר קבוע לא נשמר בזיכרון הראשי. לפעמים הנתונים ממוקמים בזיכרון הראשי למשך מספר שניות בלבד, רק כל עוד הם נדרשים.

<u>זיכרון משני</u> הוא מקום לשמירת תוכניות ונתונים לטווח ארוך. התקני זיכרון משני נפוצים הם hard disk ודיסק אופטי.

קיבולת של הזיכרון המשני היא בפי כמה סדרי גודל יותר מקיבולת הזיכרון הראשי.
אבל, דיסק הוא מאוד איטי ביחס לזיכרון הראשי. אילו למחשב היה רק זיכרון משני, המחשבים היו איטיים כמו צב!
בלוקים גדולים של הנתונים מועתקים מהזיכרון המשני לזיכרון הראשי. הפעולה מאוד איטית, ולכן בדרך כלל מעתיקים בבת אחת כמות גדולה של נתונים ולא נתון בודד. המעבד יכול לקרוא ולכתוב במהירות את הנתונים שנמצאים בזיכרון הראשי. בתום הפעולה, הנתונים שכתבנו נרשמים בזיכרון משני.

<u>העבודה</u>

בעבודה זו אתם נדרשים לממש מערכת המדמה מערכת לניהול זיכרון וירטואלי.
במערכת כזאת ההנחה שכמות הנתונים שהתוכנית צריכה בזמן ריצתה היא גדולה מאוד, ואין אפשרות להעתיק את כולם לזיכרון הראשי. אבל, המעבד יודע להשתמש בנתונים או לבצע הוראות רק אם הם נמצאים בזיכרון הראשי.
איך מתגברים על הבעיה?
מחלקים את הנתונים ל- m בלוקים בגודל קבוע הנקראים דפים (למשל של 4K), ובכל רגע נתון דואגים שהדפים הנחוצים לביצוע ההוראה הנוכחית של התוכנית יימצאו בזיכרון הראשי.
כל הדפים נשמרים ב**זיכרון המשני**. לעומתו, ה**זיכרון הראשי**, יכול להכיל רק n דפים בכל רגע נתון.
שימו לב: $m >> n$.
כאשר התוכנית רוצה לפנות לדף מסוים, על מנת לקרוא ממנו או לכתוב אליו, הדף צריך להיות בזיכרון הראשי. לכן, לפני שניגשים לדף, בודקים אם הוא נמצא בזיכרון הראשי. אם הוא לא נמצא מביאים אותו מהזיכרון המשני.
בעבודה זו, נניח שלכל דף בזיכרון המשני יש מפתח ייחודי – אינדקס מ- 0 עד m-1, והגישה לדף מתבצעת באמצעות המפתח שלו.
מהרגע שבו דף מסוים מועלה לזיכרון הראשי, התוכן שלו משתנה אך ורק בזיכרון הראשי. רק ברגע שדף מוצא מהזיכרון הראשי נעדכן אותו גם בזיכרון המשני – דוגמה תינתן בהמשך.

- **מרכיבי המערכת:**
  - זיכרון משני

    נייצג דפים בזיכרון המשני על ידי מערך A בגודל m של מחרוזות (String). כל תא במערך מייצג דף בזיכרון, כאשר A[i] שומר את התוכן של הדף שהמפתח שלו i.

    אתם נדרשים לממש גישה לדף בזיכרון המשני בהינתן המפתח ב - **O(1)**.
  - זיכרון ראשי

נייצג את הדפים בזיכרון הראשי באמצעות תור באורך n. גישה לדפים במידה והם נמצאים בתור נדרשת למימוש ב - **O(1)**. **שימו לב שמימוש התור נתון לשיקולכם ושימוש בתור סטנדרטי ללא שינויים לא יעמוד בתנאי היעילות הנדרשים**.

- **ניהול התור ואסטרטגיות להחלפות דפים**

מכיוון שהזיכרון הראשי מוגבל בכמות הדפים, כאשר דף חדש מובא מהזיכרון המשני לראשי, אם אין מקום פנוי, צריך להחליפו בדף אחר בזיכרון הראשי. בעבודה זו נממש שתי אסטרטגיות להחלפות דפים:

  - **FIFO** (First In First Out): מחליפים את הדף הוותיק ביותר.
  - **LRU** (Least Recently Used): מחליפים את הדף שזמן הפנייה (קריאה או כתיבה) האחרון אליו הוא הרחוק ביותר מבין כל הדפים הנמצאים בזיכרון.

שתי אסטרטגיות החלפת הדפים נדרשות למימוש בסיבוכיות הזמן **O(1)**.

- **אתחול המערכת**

זיכרון משני: נאתחל את כל התאים של מערך בזיכרון המשני במחרוזות ריקות.

זיכרון ראשי: נעלה את n הדפים הראשונים מהזיכרון המשני לזיכרון הראשי לפי הסדר (A[0]…A[n-1]).

<u>שימו לב</u>: במהלך התרגיל סיבוכיות הזיכרון הנדרשת <u>עבור מבנה נתונים לניהול הזיכרון הראשי</u> הינה **O(n*page_size + m)**, כאשר page_size הוא גודל הדף (4K בדוגמה שלמעלה).

- **תמיכה בפעולות**

אתם נדרשים לתמוך בפעולות הבאות.

1. read(key) – מחזירה את התוכן של הדף שהאינדקס שלו (המפתח) הוא key.
2. write(key,char) – מוסיפים לתוכן של הדף שהאינדקס שלו key את האות char (באמצעות שרשור לסוף הדף).

כפי שהוסבר קודם, בשתי הפעולות הנ"ל המערכת מחפשת קודם האם הדף נמצא בזיכרון הראשי.
אם הדף קיים בזיכרון הראשי, היא עובדת עם הדף הנ"ל.
אחרת, המערכת מעתיקה את הדף מהזיכרון המשני לראשי (עם החלפת דפים לפי הצורך לפי האסטרטגיה שנבחרה).

<u>דוגמה לפעולת כתיבה:</u>

כתיבה לזיכרון המשני מתבצעת רק כאשר מוציאים דף שכתבו עליו מהזיכרון הראשי, לדוגמא:

לאחר ביצוע write(55, 'a'):

הדף המתאים בזיכרון הראשי יכיל את המחרוזת 'a', בעוד שבזיכרון המשני, A[55] יכיל את המחרוזת הריקה. רק כאשר דף אחר יחליף את דף זה בזיכרון הראשי, התו 'a' יתווסף לתא 55 בזיכרון המשני. כלל זה מתקיים גם אם בוצעו מספר כתיבות לדף.

בנוסף הוספנו לרשותכם פונקציה נוספת:

3. ()toString – מחזירה ייצוג של הזיכרון **המשני** כמחרוזת.

- **הרצת התוכנית**

המחלקה הראשית תקרא MemoryManagement והיא תכיל את השיטה main(String[] args). הרצת התוכנית תתבצע משורת הפקודה באופן הבא:

java MemoryManagement *useLru InputFileName OutputFileName m n*

useLru – אסטרטגיית החלפת הדפים. פרמטר זה מהווה אינדיקטור לשימוש באסטרטגיית לניהול תור. נעביר "1" (ללא גרשיים) כאשר נרצה להשתמש באסטרטגיית "LRU", וכל ערך שלם אחר יגרום לשימוש באסטרטגיית "FIFO".

InputFileName – השם של קובץ הקלט אשר מכיל את הפקודות למערכת.

OutputFileName – השם של קובץ הפלט אשר אליו נכתוב את הפלט של הפקודות הניתנות למערכת.

m – גודל הזיכרון הראשי

n – גודל הזיכרון המשני

לדוגמה:

```
java MemoryManagement 1 inputFile.txt  outputFile.txt  50 1000
```

- **קבצי קלט ופלט לדוגמה:**

מצורפים לעבודה קובץ קלט לדוגמה "input.txt", אשר מכיל את אוסף הפקודות למערכת ניהול הזיכרון שלכם. בנוסף מצורפים שני קבצי פלט, output(FIFO).txt ו- output(LRU).txt, המכילים את הפלט של המערכת בהתאם לשתי האסטרטגיות השונות בעבודה עבור m =50, n = 1000.

- **עבודה עם קבצים - קריאה וכתיבה:**

בתוך מחלקה MemoryManagement ישנם פונקציות אשר קוראות וכותבות לקבצים. מומלץ לעבור על הקוד ולהבין איך הוא עובד.

- **זמני ריצה**

זמן הריצה של כל הפעולות שנדרשת לממש הינו - O(1).

- **ערות והכוונות**

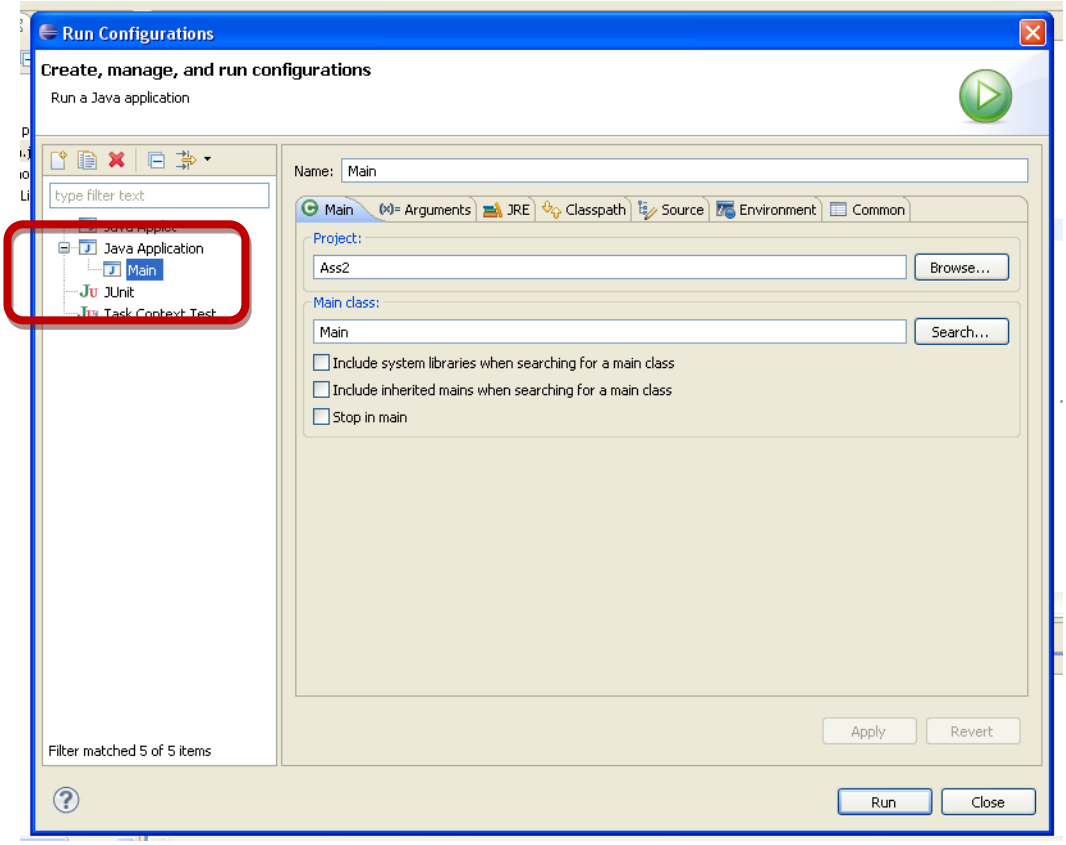
אם תרצו לדמות ב- Eclipse הרצת תוכנית משורת פקודה שמקבלת פרמטרים ב- main שלה:

1. הריצו את ה- class שמכיל את ה- main שלכם ע"י הפעלה של Run. אתם תקבלו הודעה שגיאה

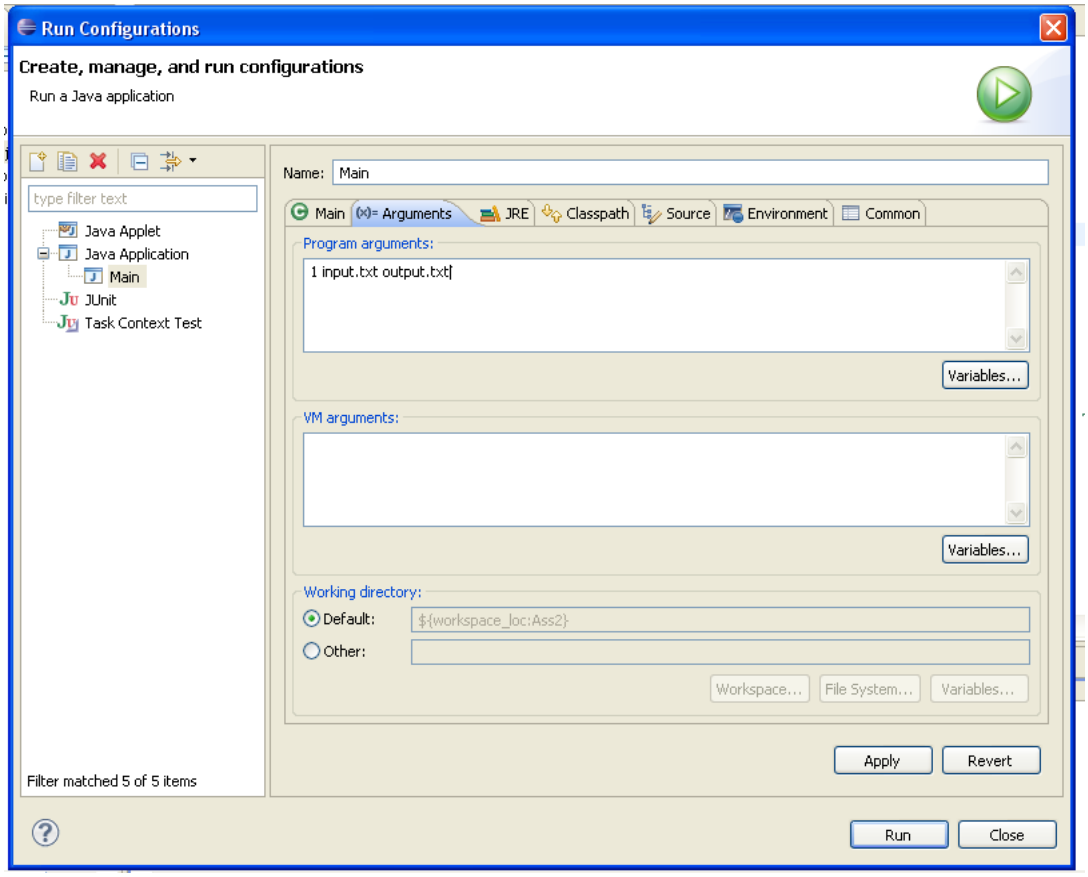
```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
    at MemoryManagement.main(Main.java:13)
```

2. עכשיו כנסו ל- Run->Run Configurations

3. בחרו בתפריט בצד שמאל Java Application → Main



4. כנסו ל- Arguments וכתבו את הארגומנטים אשר התוכנית שלכם מקבלת כלקט.



עבודה מהנה!