

# המסלול האקדמי המכללה למינהל

## ביה"ס למדעי המחשב



ת.ז הסטודנט: \_\_\_\_\_

מספר חדר: \_\_\_\_\_

מספר נבחן: \_\_\_\_\_

מספר אסמכתא: \_\_\_\_\_

ברקוד נבחן

### מבחן בקורס: תכנות מונחה עצמים

תאריך הבחינה: 03.08.17

שנת הלימודים: תשע"ז, סמסטר: ב', מועד: א'

משך הבחינה: 3 שעות

שם המתרגל/ים:

רועי יהושוע

שם המרצה/ים:

ד"ר אליהו חלסצ'י

מבנה הבחינה: הבחינה מורכבת מחלק אחד.

מספר השאלות הכולל בבחינה: 4.

משקל כל שאלה: בצמוד לכל שאלה

הוראות לנבחן:

- אסור השימוש בכל חומר עזר
- יש לענות במחשב.
- לא נדרש להחזיר את השאלון.
- לא מצורף נספח לבחינה
- מחברת טיוטה : לא
- מחברת נפרדת לכל שאלה : לא

**בהצלחה!!**

## הקדמה

במבחן זה עליכם לענות על 4 \ 4 שאלות תכנותיות ב JAVA. משך המבחן 3 שעות. חומר פתוח. עליכם להקפיד היטב על ההוראות, ובפרט על הוראות ההגשה, שכן הבדיקה הינה אוטומטית.

אתם מקבלים:

- את קובצי המקור אותם עליכם להשלים
  - MainAPI לבדיקה לוקאלית של ה API
  - MainTrain של מוד האימון במערכת ההגשה
- עליכם להגיש
- את קובצי המקור מושלמים. לא ב zip או דומיו, אלא את קובצי המקור עצמם.

קוד שלא מתקמפל או שיש לו שגיאות בזמן ריצה יקבל הפחתה אוטומטית של עד 20 נקודות באופן יחסי לסעיף הבדיקה שנכשלה (וייבדק ידנית)

עליכם להקפיד על ה API הנדרש, הבדיקות של ה Main הלוקאלי, וכמובן על הוראות התרגיל, שכן ה Main הלוקאלי ו MainTrain בפירוש לא בודקים את כל המקרים שכן נבדקים במערכת האוטומטית של המבחן.

במהלך המבחן תוכלו להגיש את הקוד כמה פעמים שתמצאו גם במוד אימון וגם במוד הגשה. המבחן ייבדק רק ע"פ הקוד שהוגש למוד הגשה לפני שתם זמן המבחן. אל תטעו בהגשה או במודים. כל פרטי ההגשה נמצאים בסוף מסמך זה.

**אזהרה:** כבר נתפסו מעתיקים בעבר. נא להימנע מדבר שעלול להרוס לכם את התואר.

**בהצלחה!**

אלי.

## שאלה 1 (30 נק'): Architectural Patterns – pipes & filters, concurrency

בקובץ Q1.java מופיעה המחלקה Stream ומתודת mainAPI הבודקת את ה API והמטרות שלה:

```
BlockingQueue<Point> result;
// define the stream
Stream<Point> s=new Stream<>();
result = s.filter(p->p.x>=0).filter(p->p.y<=0).getBuffer();
// the stream is still empty.

// printing thread
final boolean[] stop={false};
new Thread()->{
    try {
        while(!stop[0])
            while(!result.isEmpty())
                System.out.println(result.take());
    } catch (InterruptedException e) {}
}).start();

// a demo of a slow stream-generation
Random r=new Random();
for(int i=0;i<500;i++){
    s.push(new Point(-100+r.nextInt(201),-100+r.nextInt(201)));
    Thread.sleep(50);
}
// stopping the stream(s)
s.endOfStream();

// stopping the printing thread
stop[0]=true;

// result: as the new points are generated,
//          only points with x>=0 & y<=0 are printed
```

ב main לעיל אנו מייצרים מופע של Stream<Point> המאפשר ארכיטקטורת pipes and filters ו fluent programming. באמצעות ביטוי למדה המתודה filter מאפשרת להעביר הלאה את כל הנקודות עם x לא שלילי, ומאלה להשאיר רק את הנקודות עם y לא חיובי. התוצאה תישמר ב result. אולם, בינתיים לכאורה לא קורה דבר, שכן ה stream ריק ממידע.

כעת אנו מגדירים ת'רד אנונימי שפשוט מדפיס את התוכן של result, ככל שיתקבלו לתוכו אובייקטים. הוא חי ברקע.

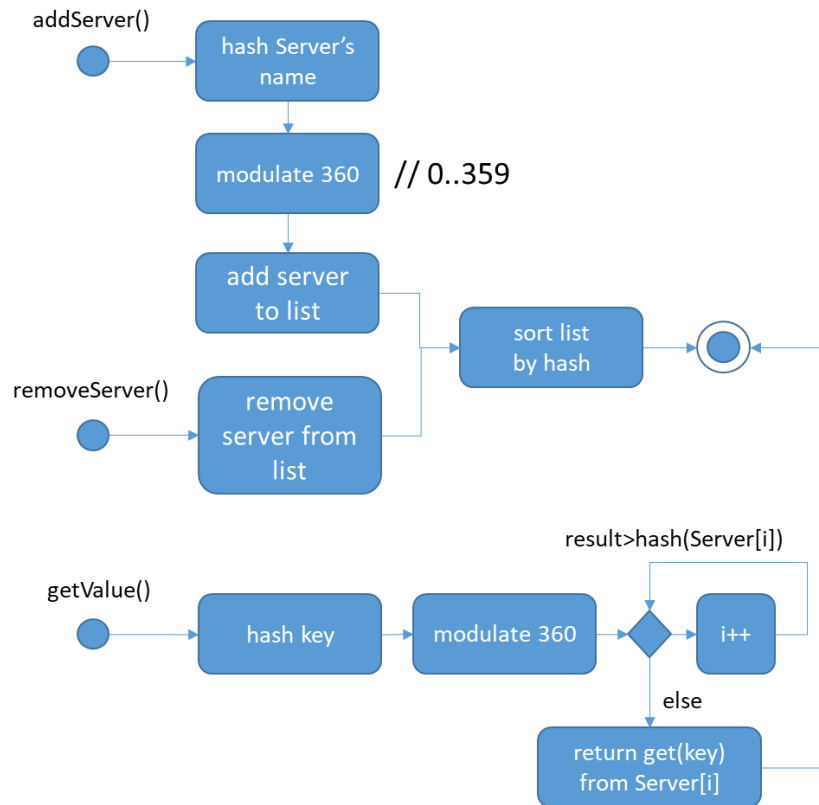
לאחר מכן אנו מייצרים 500 נקודות אקראיות עם ערכי x, y בין -100 ל 100, ומכנסים אותן ל stream. תור כדי הכנסת (ולא רק לאחר שמסתיים הקלט) הן יעברו סינון בהתאם להגדרות לעיל; "השורדים" יכנסו ל result, ויודפסו ע"י הת'רד שהגדרנו.

הפקודה endOfStream מורה על סיום הקלט הנכנס ל stream וכל משאב שצרכנו ישוחרר.

עליכם להשלים את הקוד של המחלקה Stream<T> כך שנוכל להפעיל את המתודות filter ו endOfStream בהצלחה ולקבל את התוצאה הרצויה להפעלה דומה לזו שב main לעיל.

## שאלה 2 (30 נק'): UML

בקובץ Q2.java מופיעה המחלקה ConsistentHasher שמטרתה לבצע consistent hashing לשרתים. עליכם להשלים את המתודות החסרות במחלקה זו (בלבד) ע"פ דיאגרמת ה UML הבאה. לאחר המחלקה תוכלו למצוא מחלקת שרת לבדיקה ומתודת mainAPI לבדיקת ה API ואף מעט מהלוגיקה. שימו לב שאין לשנות את מחלקת השרת לבדיקה. הפלטים הצפויים מופיעים בהערות לקוד.



## שאלה 3 (20 נק'): תבניות עיצוב

בקובץ Q3.java תמצאו את המחלקה `RoundList<T>` המייצגת רשימה מקושרת מעגלית. המחלקה

- מממשת את `Iterable` ולכן עליה להחזיר `Iterator`.
  - עושה שימוש במחלקת העזר `Item`
  - מכילה את `head` המהווה את גבול הגזרה של הרשימה ואינו מכיל `data`.
- א. השלימו את המתודה `push` כך שהאיבר החדש יידחף מימין ל `head`.
- ב. השלימו את מחלקת העזר `RoundListIterator` כך שתהווה `iterator` ל `RoundList`. הערות:
- כשנוצר `iterator` חדש הוא יצביע על האיבר הראשון עם `data`. ניתן להניח שייצרו אותו לאחר שכבר הוכנסו איברים.
  - קיים `next` אם לא הגענו כבר לגבול הגזרה.
  - כזכור, `next` מחזיר את האיבר עליו אנו מצביעים כעת, ולאחר מכן מקדם את ה `iterator`.
  - המתודה `nextR` נועדה לשימוש מעגלי, ולכן תפעל באופן דומה ל `next` אך עליה לדלג על `head`.

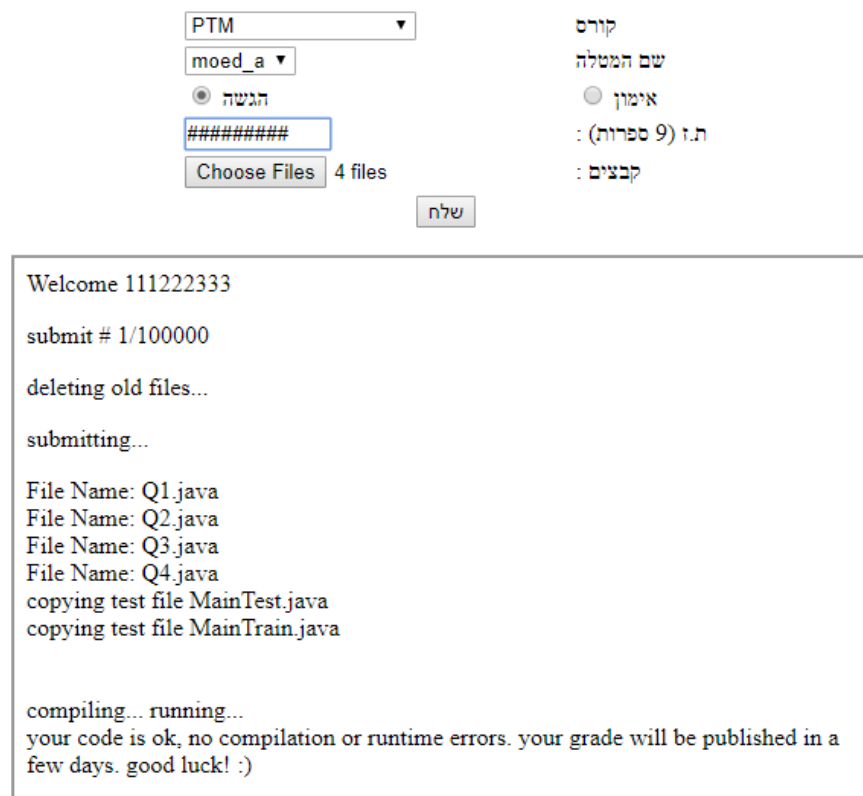
בסוף הקובץ תמצאו מתודת `mainAPI` לבדיקת ה API והמטרות. הפלטים הצפויים מופיעים בהערות לקוד.

#### שאלה 4 (20 נק'): שאלות כלליות על החומר.

נכון \ לא נכון. החזירו true או false בהתאמה במתודות a עד d של המחלקה Q4 הנמצאת בקובץ Q4.java בהתאם לסעיפים הבאים.

- ישנם שלבים של Sokoban **שלא ניתן** לפתור אותם באמצעות אלג' Strips המקורי.
- כדי להפריד עיצובית בין אלגוריתם לבעיה שאותה הוא פותר יש להשתמש ב Strategy Pattern.
- ב Java אנו יכולים לממש ספריית "naked objects" בעצמנו מכיוון שהיא מאפשרת reflection.
- ב MVVM ה M מחזיק VM, ה VM מחזיק V.

בהצלחה!



הגשה:

- <http://ck.cs.colman.ac.il>
  - כל המחלקות צריכות להיות ב package בשם test
  - עליכם להגיש למערכת ההגשה, תחת PTM ו moed\_a, את הקבצים Q1.java, Q2.java, Q3.java, Q4.java בלבד.
  - כל הגשה צריכה לכלול את כל הקבצים האלה.
  - תבדילו בין מוד אימון למוד הגשה. רק מוד הגשה נחשב כהגשת המבחן.
  - המערכת לא תקבל ZIP או דומיו, או כל קובץ שאינו קוד מקור.
  - אנא אל תעשו טעויות מביכות שיעלו לכם במבחן. ההוראות ברורות ופשוטות.
- בנוסף הגישו את קוד המקור לשרת הגיבוי. כאן ניתן להשתמש ב ZIP. אך אנא וודאו שהוא מכיל את קוד המקור ושהוא לא יוצא לכם פגום. תבדקו את הקובץ לפני שליחה.

<http://db.cs.colman.ac.il/test/>