בס"ד

מינהל הסטודנטים | אוניברסיטת בר-אילן | מדור הבחינות

מספר הקורס: 89-211-01

תאריך בחינה: _____

הכ חברת נבדקה ביום: _____

הציון: _____

חתימת המרצה: _____

מס' סידורי _____ מתוך _____ מחברות

מס' מח': 38

שנת: תשע"ז סמסטר: 2 מועד: 2 מטלה: 1.1

קורס: 89211-02 תכנות מתקדם 2

0010002138810000015207 3

הוראות לנבחן

1. עליך להבחן בחדר בו הנך רשום.
2. הנח ליד המשגיח בבחינה את חפציך האישיים כגון: תיקים, ספרים, מחברות, מכשירים סלולריים, קלמרים וכו'.
3. אסור להחזיק בחישב יד חומר הקשור לבחינה/לקורס אלא אם הותר הדבר בכתב על ידי המרצה ורק בהתאם למותר.
4. מסור למשגיח/ה על הבחינה תעודת זהות וכרטיס נבחן חתום ותקף לסמסטר בו מתקיימת הבחינה.
5. **היציאה לשירותים במהלך הבחינה אסורה בהחלט.** נשים בהריון ונבחנים באישור מתאים רשאים לבקש מהמשגיח/ה לצאת. היציאה בליווי המשגיח/ה ובהתאם לנוהלי האוניברסיטה.
6. נבחן היוצא ללא רשות מחברתו תפסל ותועבר לועדת משמעת.
7. יש להישמע להוראות המשגיח/ת. אין לעזוב את חדר הבחינה ללא קבלת רשות. חל איסור מוחלט לפנות לנבחנים אחרים בכל עניין ודבר. בכל עניין פנה **למשגיח/ה.**
8. בתחילת הבחינה מלא את פרטיך האישיים ע"ג המחברת. תלמיד שקיבל לידו שאלון ואין ברצונו להיבחן, חייב להמתין 1/2 שעה בכיתה מתחילת הבחינה. תלמיד שעזב את האולם אחרי חלוקת השאלונים או לא מסר את מחברתו עד תום הבחינה או מסר מחברת ריקה - דינו כדין נכשל.
9. קריאת השאלון מותרת רק לאחר קבלת רשות המשגיח/ה.
10. יש לכתוב את התשובות בדיו, בכתב ברור ונקי על עמוד אחד של כל דף. אין לכתוב בשוליים. הכותב טיוטה יקדיש לה את הצד הימני של המחברת ואת ההעתקה הנקיה יכתוב בצד השמאלי. את הטיוטה יש למחוק בהעברת מו. אסור לתלוש דפים מן המחברת.
11. עבר הנבחן על תקנות הבחינות, תשלל ממנו הרשות להמשיך בבחינה, והוא יועמד לדין משמעתי.
12. משך זמן הבחינה מצויין בראש השאלון. עם הודעת המשגיח/ה כי תם הזמן, על הנבחן להפסיק את הבחינה, למסור את המחברת עם השאלון ולצאת מאולם הבחינה. מחברת שלא נמסרה בתום ההודעה לא תיבדק.
13. אחזקת מכשיר טלפון סלולרי (אפילו סגור) ברשות הנבחן, מביאו מיידית לפסילת הבחינה.

בהצלחה!

שנת"ל תשע"ז סמסטר ב' מועד ב'

מס' קורס 89-211-02

מחלקה מדעי המחשב תאריך 8.9.17

המרצה ד"ר אלעזרי אליהו

מבחן חלק (אם הבחינה בשני חלקים) _____

!עדת המשמעת מזהירה!

נבחן שיימצאו ברשותו חומרי עזר אסורים או ייתפס בהעתקה, ייענש בחומרה עד כדי הרחקתו מהאוניברסיטה.

אין לכתוב בשולי המחברת

הוראות לנבחן בנושא סריקה:

אין לכתוב במחברת בעפרון. יש לכתוב בעט בצבע כחול כהה או שחור בלבד. אין להשתמש בנוזל מחיקה (טיפקס). אין לכתוב בשוליים משני צידי הדף. מחברת בכתב מרושל משפיעה על תוצאות הסריקה.

ועדת המשמעת מזהירה!
נבחן שימצאו ברשותו חומרי עזר אסורים או יתפס בהעתקה יענש בחומרה עד כדי הרחקתו מהאוניברסיטה

89-211-01-02 **בגוף סגור**

תכנות מתקדם 2, מרצה: ד"ר אליהו חלסצ'י, מתרגל: מר רועי יהושוע

תכנות מתקדם 2: 89-211 – מועד ב' תשע"ז

זמן המבחן: **שעתיים וחצי**, יש לענות על 4 מתוך 4 שאלות, **בגוף השאלון בלבד**. חומר **סגור**.

שתי שאלות בקיאות + שתי שאלות עיצוב קוד ותכנות (java).

שאלה1 (30 נק'):

בארכיטקטורת MVC כתב אריק Controller הפועל באופן הבא. ל Controller רק שתי מתודות פומביות. האחת מאפשרת הכנסה של אובייקט מסוג ממשק Command למאגר הפקודות. השנייה מבקשת הפעלת אובייקט פקודה בהינתן מפתח מסוג String. אובייקט הפקודה נשלף ב O(1) מתוך מפה, ונכנס לתור עדיפויות. ת'רד שרץ ברקע שולף את הפקודות ומריץ אותן ע"פ העדיפות כל עוד התור אינו ריק. נתחו את הפתרון של אריק ל Controller ע"פ עקרונות SOLID (כל עקרון = 6 נק')

S: single responsibility - ה- Controller גם מכניס ומוציא לתור עדיפויות (command) ולכן יש לו אחריות יתירה ולכן האחלקה כי אחריות אינה לה.

O: כאן הכי הרבה בעיות מופיעות תמיד ויש לנו את הקונטרולר, command controller והבעיות היא איך כל זה שקשור ל- controller נמצא באחלקה. ולכן אריק מקיים עיקרון זה.

L: אסור - כל מה שאמור ל- controller קורה אולי ולכן אין בעיה אבל הפונקציה אמורה להוציא מהאחלקה כי לא תקין ה controller להכניס והוציא בקשות) וגם האחלקה מקיימת עיקרון זה.

I: interface Seprayation - הממשקים אלו לא גדולים וכל אחד יש רמוז את מה שהוא מעדיף ולכן כל משתמש הוא נפרד מהשני (וזה נכון להיות כי אי אפשר להפריד) ולכן לא מקיים את העיקרון הזה.

D: dependancy inversion - מכיוון שה- controller מניח מראש command אל בעיות תלויות ארכיטקטוניות כיון אלייקיים, כלומר אין מודולרי controller והיה מומלץ (לא אסור) אל לא יהיה מודולרי ותהיה תלויות ארכיטקטוניות כיון אלייקיים. ולכן אריק מקיים עיקרון זה.

* לפי מרצה הכרח אל בנוסף ונוכל את המשימות לא לאור ל- controller ולכן מתקיים single responsibility ל- אסור.

~~single responsibility~~

~~Liskov~~

~~I(interface)~~

interface segregation

dependancy inversion

30 40

שאלה 2 (10 נק'):

הקיפו בעיגול את התשובות הנכונות:

א. נעדיף עורך ויזואלי שמייצר לנו את הקוד במקומנו
ב. Data Binding מאחורי הקלעים ממומש ע"י ה Strategy Pattern
ג. מנגנון String Pool מאפשר השוואות של מחרוזות באורך N ב O(1).
ד. נעדיף High Cohesion ו Low Coupling

שאלה 3 (30 נק'):

ברצוננו לכתוב מנגנון ל Consistent Hashing. המפתחות יהיו מסוג String ואילו הערכים מסוג פרמטרי T. לכל שרת יש שם ייחודי. פונקציית ה hash תייצר ערך בין 0..359 (מעלות במעגל). אופן הפעולה הוא החזקה של רשימה של שרתים. השרתים יהיו ברי השוואה (Comparable). כאשר נוסף או נופל שרת נמיין את הרשימה הזו ע"פ מיקום השרתים במעגל (ה hash שלהם). בהינתן מפתח, נקצה אותו לשרת הקרוב ביותר שנמצא עם כיוון השעון על אותו המעגל. השרת יבדוק האם הערך שמור אצלו ב RAM, אם כן, יחזיר אותו, ואם לא, יאחזר אותו ממסד הנתונים וישמור ב RAM לשימוש עתידי. דוגמא להפעלה:

```
ConsistentHasher<Integer> ch=new ConsistentHasher<>();
ch.addServer(ch.new Server("AA"));
ch.addServer(ch.new Server("BB"));
ch.addServer(ch.new Server("CC"));
ch.addServer(ch.new Server("DD"));
ch.addServer(ch.new Server("EE"));
ch.debug("Eli");
```

האובייקט ConsistentHasher יודע לאחזר אובייקטים מסוג Integer. הוספנו לו חמישה שרתים בשמות שונים, והפעלנו את פונקציית ה debug המופיעה בסוף הקוד שבהמשך. הפלט שלה הוא:

```
DD,16
EE,48
AA,280
BB,312
CC,344

282 -> BB
```

ניתן לראות את המיקום במעגל שקיבל כל שרת. נשים לב ש AA קיבל 280 ו BB 312. המפתח "Eli" קבל ערך 282 ולכן הוא משויך לשרת BB. הרי כל הערכים שבין 281 ל 312 שייכים ל BB.

השלימו את הקוד הבא כדי לגרום למנגנון זה לעבוד.

```
public class ConsistentHasher<T> {

    public class Server implements Comparable<Server>{

        private HashMap<String,T> map; // 5 points
        private String name;

        public Server(String name) {
            map=new HashMap<>();
            this.name=name;
        }
```

```
T getFromDB(String key){
        /* retrieves T form data base, no need to implement*/
}

T getValue(String key) { // 5 points
  if (map.containsKey(key)) {
        return map.get(key);
  } else { T t = getFromDB(key);
        map.add(key, t);
        return t;
  }
}
public int hashCode(){ // 0..359 by servers name, 5 points
  return (name.hashCode()) % 360 ;
}
                                  אופציה
public boolean equals(Server s){return name.equals(s.name);}

@Override
public int compareTo(Server o) { // 5 points
  return this.hashCode() - o.hashCode() ;
}
}// end of inner class Server, back to ConsistentHasher ckass

List<Server> servers;
public ConsistentHasher() {servers=new LinkedList<>();}

Server debugServer;
int debugKeyIndex;

public T getValue(String key){ // 10 points
  int index=(key.hashCode()) % 360 ; // the hash of key
  debugKeyIndex=index;
  Server chosen = servers[0];
  for (int i=1; i < servers.size() ; i++) {
        if (servers[i].hashCode() - debugKeyIndex) % 360 > (servers[i].hashCode() - debugKeyIndex) % 360
              chosen = servers[i];
        }
  }
  debugServer=chosen;
  return chosen.getValue(key);
}
```

אופציה אם אין אף שרת

```
if(servers==null || servers.empty()){
    return null
}
```

Server chosen = servers[0];

for (int i=1; i < servers.size() ; i++) { לולאה

if (servers[i].hashCode() - debugKeyIndex) % 360 > (servers[i].hashCode() - debugKeyIndex) % 360

כמו (-) אבל לא

אם (-) אז

chosen = servers[i];

}

}

debugServer=chosen;

return chosen.getValue(key);

}

* אני יודע שכתבתי רק compareTo -> compareTo אבל אני לא רואה בסעיף אשמת compareTo

אבל כתבתי אותו שבלי עזרה.

```
    public void addServer(Server server){
        servers.add(server);
        Collections.sort(servers);
    }

    public void removeServer(Server server){
        servers.remove(server);
        Collections.sort(servers);
    }

    public void debug(String key){

        Iterator<Server> it=servers.iterator();
        while(it.hasNext()){
            Server s=it.next();
            System.out.println(s.name+","+s.hashCode());
        }
        System.out.println();
        getValue(key);
        System.out.println(debugKeyIndex+" -> "+debugServer.name);
    }
}
```

שאלה 4 (30 נק'):

הביטו ב main הבא, המגדיר את המטרות שעליכם להשיג.

```
BlockingQueue<Integer> result;
Stream<Point> s=new Stream<>(); // defines the stream
result = s.map(p->p.x).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();

// demo of a slow stream generation
Random r=new Random();
for(int i=0;i<500;i++){// x>=0       y<=0
    s.push(new Point(r.nextInt(101),-r.nextInt(101)));
    Thread.sleep(50);
}
// stopping the stream(s)
s.endOfStream();

// stopping the printing thread
stop[0]=true;
// result: as new points are generated, only x values are printed
```

ב main לעיל אנו מייצרים מופע של Stream<Point> המאפשר ארכיטקטורת pipes and filters ו fluent programming. באמצעות ביטוי למדה המתודה map מאפשרת למפות כל אובייקט Point לערך ה x שלו מסוג int. לו רצינו יכולנו להמשיך לשרשר עוד מיפויים. 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>=0 ו Y<=0, ומכנסים אותן ל stream. תוך כדי הכנסתן (ולא רק לאחר שמסתיים הקלט) הן יעברו מיפוי 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 map ו endOfStream בהצלחה ולקבל את התוצאה הרצויה להפעלה דומה לזו שב main לעיל.

```
public class Stream<T>{

    public interface Map<P,R> { // 5 points
        public R getVal(P p);
    }

    BlockingQueue<T> buffer;
    volatile boolean stop;
    Thread thread; // 1 points
    Stream<R> str; // 3 points

    public Stream() {
        buffer=new LinkedBlockingQueue<T>();
        stop=false;
    }

    public void push(T t){  buffer.add(t); }

    public BlockingQueue<T> getBuffer(){ return buffer;}

    public void endOfStream() { // 6 points
        stop = true;
        if (thread != null) { thread.interupt(); }
        if (str != null) { str.endOfStream(); }

    }
```

```
public <R> Stream<R> map(Map<T,R> m){  // total of 15 points
        str = new Stream<R>();
  thread = new Thread(()->{
      while (!stop){
          try {
              T t=buffer.take();
              str.push(m.getVal(t));
          } catch (InteruptedException e){}
      }
  }
  thread.start();
  return str;
}
}
```

בהצלחה!

```
try {
    T t= buffer.take
    str.push(m.getVal(t))
} catch (InteruptedException e){}
```

```
return next;
```