וואלון שוב

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

תכנות מתקדם 2: 211-עס – מועד א' ונשע' ו

זמן המבחן: **שעתיים**, יש לענות על 6 מתוך 6 שאלות, **בגוף השאלון בלבד**. חומר **סגור**.

בקיאות

שאלה1: (16 נק')

נתונה לנו מערכת לניהול אותות מצוקה (signals) עבור קבוצת רובוטים. במחלקה SignalManager קיימת המתודה ()sendAck שפועלת באופן הבא: כל עוד אות המצוקה נקלט - המשך לשלוח Ack לרובוט שבמצוקה.

אליס העלתה שקיים חשש ש ()sendAck תהיה מתודה חוסמת במקרה וה Ack לא נקלט אצל הרובוט שבמצוקה - הרי הוא ימשיך לשלוח את אות המצוקה כל עוד הוא לא קיבל Ack...

בוב טוען שלכן המחלקה SignalManager צריכה להיות Active Object. שכן, אם הגיעו מספר אותות מצוקה במקביל ועבור כל אחד הפעלנו את ()sendAck, אז היות ו ()sendAck הופכת להיות אסינכרונית אז לא תהיה בעיה לשלוח לשאר הרובוטים Ack, גם כאשר אחד מהם לא מצליח לקבל את האות.

אליס טוענת שבוב טועה, והמחלקה SignalManager צריכה לעשות שימוש ב Thread Pool.

נמקו מדוע האחד צודק והשני טועה.

(8 נק') מי צודק? אליס, מדוע? מרגיון שבשימוש ב - ThreadPool אנחנו ~~[illegible]~~ (בעל thread לכל רובוט שיצא אל מצוקה. וכך, גם אם ~~[illegible]~~ המתודה sendAck() תהיה חוסמת (כי הרובוט לא מקבל את ה Ack) - הדבר לא יפריע לשאר ה threads להמשיך כרגיל מול רובוטים אחרים. כלומר - מה רובוטים ~~[illegible]~~ קבלו הרי חוסמת (בהנחה-פה יש בעיה של הרבה שינים לתשאל קהים לו ציוני)

(8 נק') מי טועה? בוב, מדוע? כי ב Active Object יש רק thread אחד של ברקע (ומופעל בקונסטרקטור). ובכל פעם שיקראו למתודה sendAck() הבקשה תיכנס ל- BlockingQueue ורק אחרי שמישהו אחר מסיים ~~[illegible]~~ ה thread עדיין לשניה. כלומר, אותם המתודה תהפוך לדבר אסינכרוני אל ~~[illegible]~~ בגלל שרובוט אחד מגיע - הבאו מחכים שיגיע, כלומר, לא שיגיע דבר ונשארנו למצב שאז sendAck תסוף - לא נוכל לטפל בשאר הרובוטים שמחכים כי המשימה לא נגמרה ~~[illegible]~~ ונוכל לעבור לבאו בשאר.

שאלה 2: (12 נק')

הקיפו בעיגול את התשובות הנכונות:

(א) static design עלול במקרים מסוימים להוביל לקוד כפול
ב. נעדיף קוד שמתאפיין ב high coupling וב low cohesion
ג. ב REST נעשה שימוש ב SOAP אך לא ב WSDL
(ד) Memcahed היא טכנולוגיה המאפשרת linear scalability.

מיומנות עיצוב קוד (Design) וכתיבת קוד

שאלה 3: (20 נק') נתון לנו הממשק SignalReader המגדיר את המתודה read() שמחזירה double. בנוסף, נתונות לנו המחלקות RFSignalReader, HFSignalReader, UHFSignalReader שמימשו את הממשק, כל אחת עבור סוג אחר של אות.

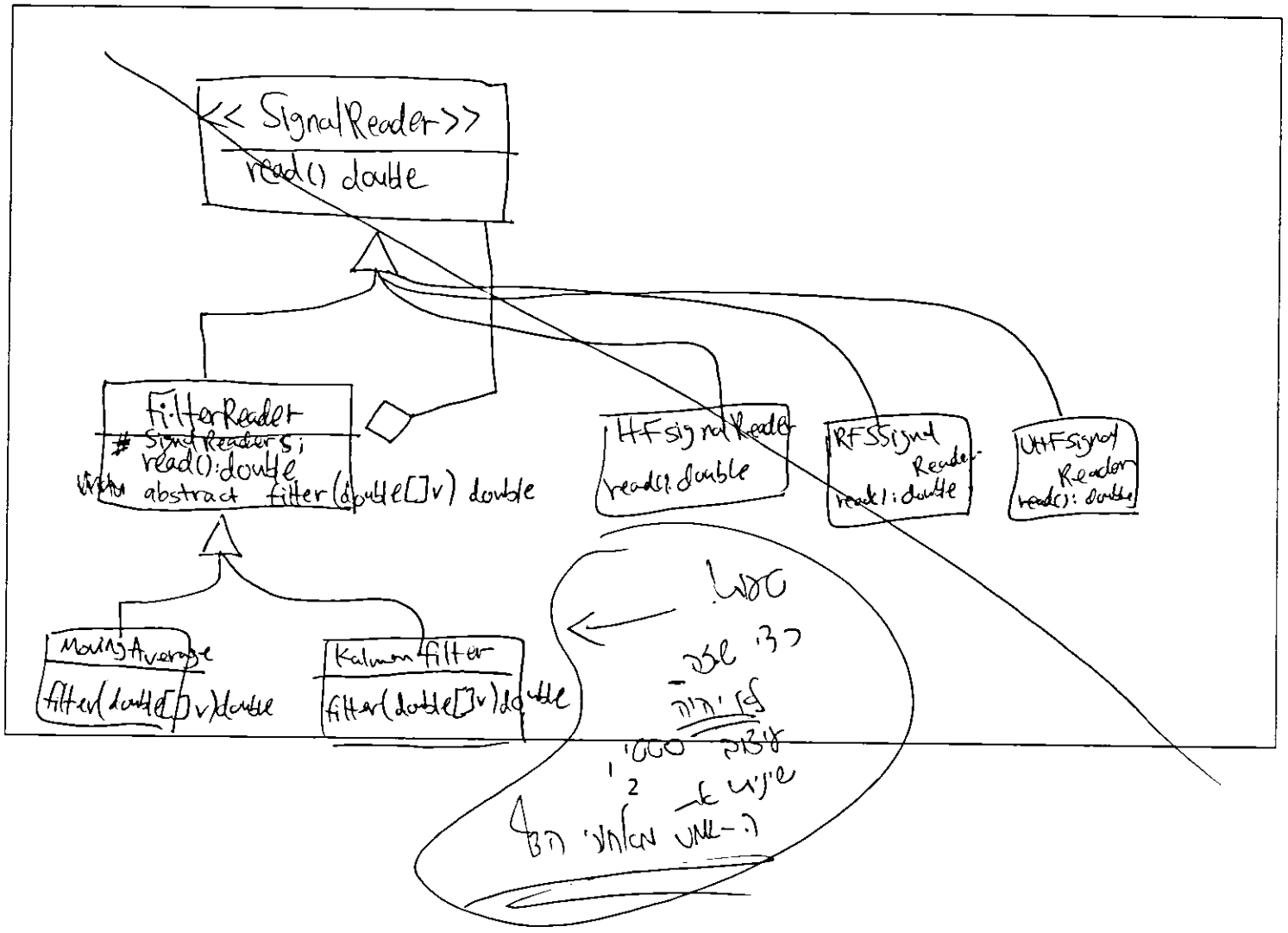
אולם, האותות המוחזרים ע"י מחלקות אלו הינם "רועשים". כלומר, בכל דגימה (הפעלה של read) הערך שמוחזר לנו נע מסביב לערך האמתי. לדוג' אם הערך האמתי הוא 88 אז חוזרים לנו ערכים מסביב ל 88 כמו 88.2, 88.3, 87.9 וכו'.

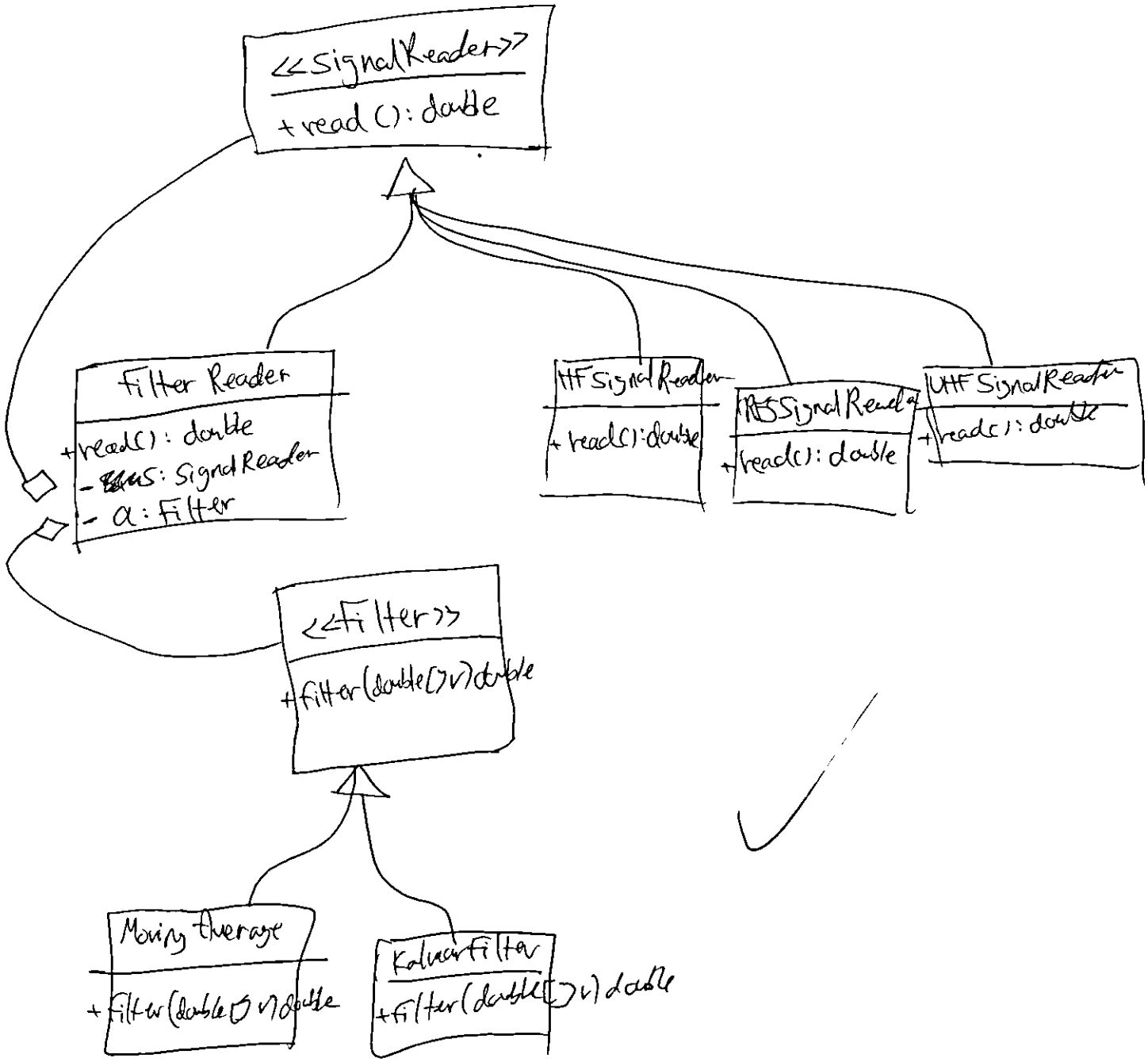
רעשים אלו צריך לסנן ולהחזיר את הערך המשוער כאמתי (לדוג' 88). המחלקות MovingAverage ו KalmanFilter מימשו כל אחת אלגוריתם שונה לסינון רעשים במתודה double filter(double[] v).

עליכם **לשרטט** תרשים מחלקות (class diagram) **ב UML**, המציג עיצוב שמאפשר לסנן את הרעשים המוחזרים ע"י RFSignalReader ודומיה. בפרט:

- המחלקה FilterReader תהיה אחראית לסינון רעשים
- הקליינט צריך להכיר רק את SignalReader (4 נק')
  - כלומר אין זה משנה לו מהו הסוג הספציפי של SignalReader, ועדיין נרצה לאפשר לו לקרוא אותות מסוננים.
- נרצה יכולת לסנן כל סוג של אות (4 נק')
- ובאמצעות כל אלגוריתם סינון (6 נק')
- נרצה שהעיצוב ישמור על עיקרון ה open / close (6 נק')

תשובה: (על השרטוט להכיל את כל המחלקות הרלוונטיות + הערות ע"פ הצורך)




<<SignalReader>>
+read():double
FilterReader
+read():double
-s: SignalReader
-a: Filter
HFSignalReader
+read():double
RFSignalReader
+read():double
UHFSignalReader
+read():double
<<Filter>>
+filter(double[]v)double
MovingAverage
+filter(double[]v)double
KalmanFilter
+filter(double[]v)double

שאלה 4: ממשו בקוד (באיזו שפה מונחית עצמים שתרצו) את המחלקה FilterReader ע"פ העיצוב שלכם משאלה 3, כך שתפעל באופן הבא: בהינתן מקור הקלט הרועש s, ובהינתן אלגוריתם הסינון a, עבור כל בקשת read אחת מ FilterReader נבצע 100 בקשות read מ s, נזין אותן ל a, ונחזיר את הערך האמתי המשוער. (16 נק')

תשובה:

```
public class FilterReader implements SignalReader{

    public FilterReader(SignalReader s){
        this.S = S;
    }

    Private SignalReader S;
    public SignalReader GetS(){
        return this.S;
    }

    public double read(){
        double [] c = new double [100];
        for (int i=0; i<100; i++){
            c[i] = this.S.read();
        }

        return filter(c);
    }

}
```

כך שומרים ולא לשלם.

לא הבנתי

too רק מ-S.

הצמד ל-a ומחזיר ערך משוער.

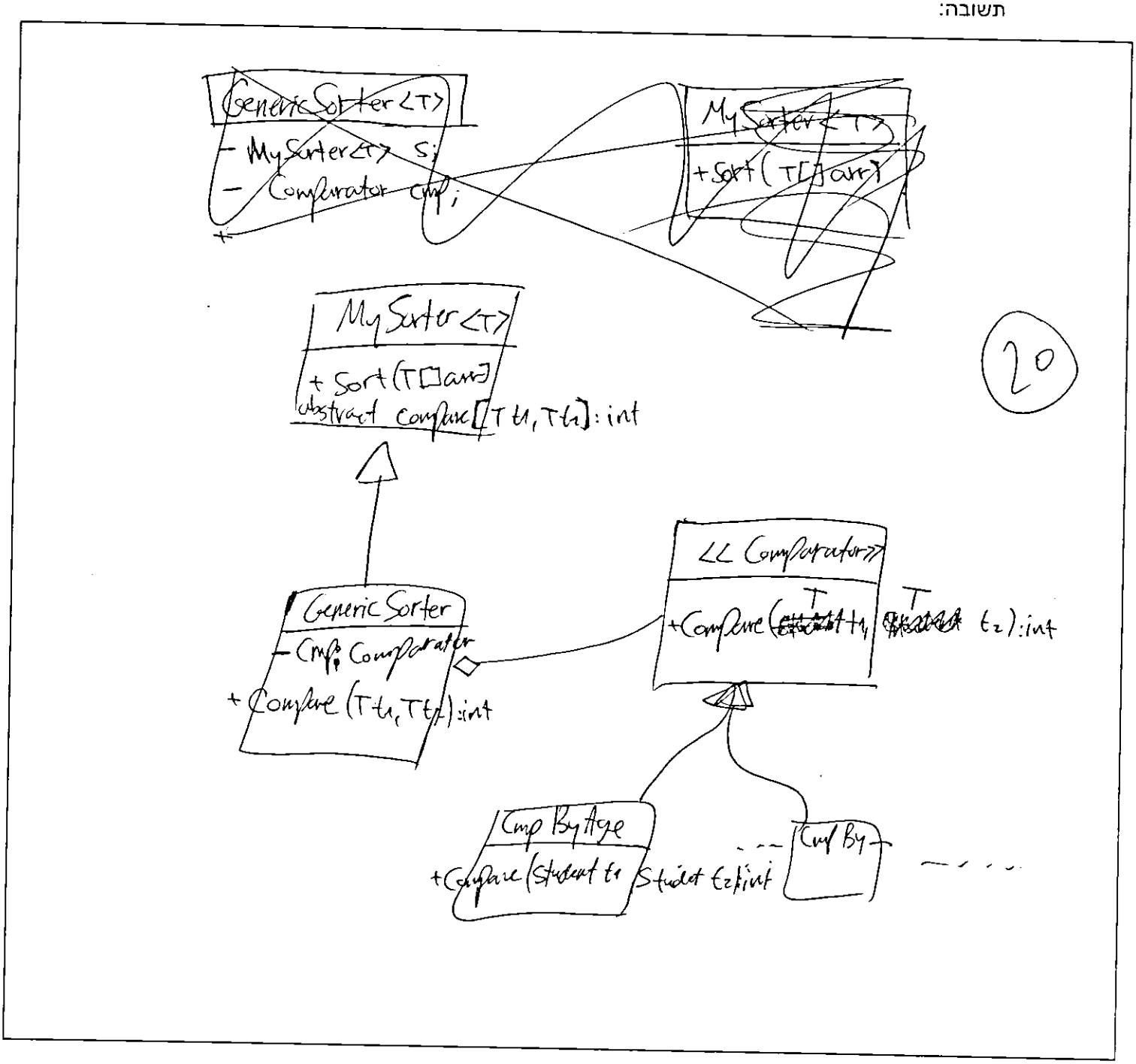
```
public class ~~SignalReader~~ FilterReader implenets SignalReader{

    private SignalReader s;
    private filter a;

    public FilterReader (SignalReader s, filter a){
        this.s=s;
        this.a=a;
    }

    public double read(){
        double [] c = new double[100];
        for (int i=0; i<100; i++){
            c[i] = this.s.read();
        }
        // מימוש
        // הפילטר פועל
        // על רשימה
        // 100 מדגמים S-S.

        // מעביר ל-a
        // ומחזיר לו
        // משהו
        return this.a.filter(c);
    }
}
```

שאלה 5: (20 נקודות)

נתונה המחלקה MySorter<T> (כקוד סגור שלא ניתן לשינוי) המאפשרת מיון של אובייקטים מסוג פרמטרי T במתודה sort(T[] arr). כדי לקבוע את הקריטריון ההשוואה בין כל שני T-ים המתודה sort מפעילה (בעת צורך) את המתודה האבסטרקטית abstract int compare(T t1, T t2). לצערנו זהו static design.

עליכם **לשרטט** תרשים מחלקות (class diagram) **ב UML**, שעושה שימוש חכם ב MySorter<T> (5 נק') אך מציג dynamic design עבור בעיית המיון (15 נק').

תשובה:



שאלה 6: ממשו בקוד (באיזו שפה מונחית עצמים שתרצו) את העיצוב של שאלה 5 והדגימו במתודת main כיצד נמיין מערך של אובייקטים מסוג Student ע"פ הגילאים שלהם (ניתן להניח כי קיימת מתודת getAge במחלקה Student). (16 נק')

16

בהצלחה!

תשובה 6:

```
template class <T>
public interface Comparator<T> {

    int Compare(T t1, T t2);
}

template class <T>
public class GenericSorter<T> extends MySorter <T>{

  Privat Comparator<T> cmp;

CTOR //  public GenericSorter <T>(Comparator <T> cmp){
            this.cmp = cmp;
        }

מימוש האינטרפייס //  public int Compare(T t1, T t2){
            return this.cmp.compare(t1,t2);
        }
}

public class CmpByAge implements Comparator <Student>{
CTOR  // public CmpByAge (){ }
```

המשך בעמ' הבא

המשך המחלקה CompByAge:

```
public int Compare(Student t1, Student t2){
    return t1.getAge() >= t2.getAge();
}
}
```

מימוש הקומפרטור.

אחרי ה main:

```
public Static Void main (){
    ....
    ....
    ~~GenericSorter<Student> new CompByAge()~~
```

מהיום: S - מערך של Student

שכבר יש בו template

לא בטוח ?

```
GenericSorter<Student> gs =
        new GenericSorter<Student>(new CompByAge());
// gs.Sort(s);
```

מעולה לאחרי הלא-בר מיין MySorter

```
}
public class Student {
    Private int age;
    Pubic int getAge(){
        return this.age;
    }
}
```

לפי הנתונים בטוח לא נמצא בכל מקום ב Student...