

בחינה בתוכנה 1 סיון טולדו, מיכל עוזרי-פלאטו, אורנית דרוו שני ביולי 2006

משך הבחינה שלוש שעות.

יש לענות על כל השאלות. בשאלות שבהן יש צורך לנמק, תשובה ללא נימוק לא תזכה באף נקודה.

יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תפסל. תשובות במחברת הבחינה לא תיבדקנה.

יש למלא מספר סידורי (מספר מחברת) ומספר ת"ז על כל דף של טופס הבחינה.

אסור השימוש בחומר עזר כלשהו, כולל מחשבוניו או כל מכשיר אחר פרט לעט.

שאלה 1, 36 נקודות

המחלקות והמנשקים הבאים מתוכננים לשליטה במכונות. כל המחלקות והמנשקים שייכים לחבילה אחת. התוכנית HyperbaricChamber מיועדת לשמור תא לחץ ברמת לחץ פחות או יותר קבועה.

מנשקים

- `Sensor<T>` מייצג חיישן שמודד גודל כלשהו בעולם הפיזי (לחץ, טמפרטורה, כיוון, מיקום, וכדומה). פרמטר הטיפוס `T` הוא הטיפוס של מדידה בודדת.
- `Actuator<T>` מייצג רכיב פיזי שמשנה את מצב המכונה הנשלטת, רכיב כגון מתג, מנוע, וכדומה.

מחלקות

- `PressureSensor` מייצג חיישן לחץ.
- `OnOffActuator` מייצג מתג `on-off`, שבתוכנית הזו מפעיל מדחס.
- `Policy` מייצגת מדיניות: איך להגיב למצבים שונים בעולם הפיזי. למחלקה הזו שירות `testAndRespond` יחיד, שצריך להפעיל באופן מחזורי. כאשר השירות הזה מופעל, הוא יכול לקרוא את החיישנים ולהגיב על ידי הפעלת רכיבי `actuator` אם צריך. רק חלק מהקוד של המחלקה הזו נתון במבחן.
- `Dispatcher` מפעיל באופן מחזורי את השירות `testAndRespond` של כל העצמים מטיפוס `Policy`. במחלקה הזו יש שירות מחלקה (static method) יחיד בשם `dispatch`, שהמימוש שלו לא נתון במבחן.
- `HysteresisController` שולט במתג `on-off` באופן הבא. הוא קורא את המדידות של חיישן. אם המדידה נמוכה מערך סף נתון (`threshold`), הוא מדליק את המתג (מעביר אותו ל-`on`). אם המדידה גבוהה מערך סף אחר, שצריך להיות גבוה מערך הסף הראשון, הוא מכבה את המתג. בתוכנית הזו, ההתנהגות הזו אמורה לגרום ללחץ בתא הלחץ להשאר פחות או יותר בין הספים. השימוש בשני ערכי סף שונים אמור למנוע כיבוי והדלקה תכופים מדי של המדחס. מזגנים ומכונות רבות אחרות פועלים כך.
- `HyperbaricChamber` היא המחלקה שמכילה את התוכנית הראשית (`main`). התוכנית
 1. קוראת את שני הארגומנטים של התוכנית (שתי מחרוזות) ומחלצת מהם מספרים שישמשו כערכי סף,
 2. יוצרת עצם מהמחלקה `HysteresisController`, ואז
 3. קוראת לשירות `Dispatcher.dispatch`, שלא חוזר לעולם, אבל מפעיל את כל ה-`Policy`-ים באופן מחזורי.

יש לקרוא את הקוד בעיון ולענות על השאלות שמופיעות אחריו.

```
public interface Sensor <T> {
    public T read();
}

public interface Actuator <T> {
    public void act(T command);
}

public class PressureSensor implements Sensor<Double> {
    public Double read() {
        ... // reads and returns the actual sensor data
    }
}

public class OnOffActuator implements Actuator<Boolean> {
    public void act(Boolean turn_it_on) {
        if (turn_it_on)
            ... // turn the switch on
        else
            ... // turn the switch off
    }
}

public abstract class Policy {
    ... // missing declarations and/or code
    abstract public void testAndRespond();
}

public class Dispatcher {
    public static void dispatch() {
        ... // missing implementation
    }
}

// continues on the next page
```

```

public class HysteresisController {

    private double          low_threshold;
    private double          high_threshold;
    private Sensor<Double>  sensor;
    private Actuator<Boolean> actuator;

    public HysteresisController(Sensor<Double>    s,
                                Actuator<Boolean> a,
                                double            l,
                                double            h) {

        low_threshold = l;
        high_threshold = h;
        sensor        = s;
        actuator      = a;

        Policy too_low = new Policy () {
            public void testAndRespond() {
                double x = sensor.read();
                if (x < low_threshold) actuator.act(true);
            }
        };

        Policy too_high = new Policy () {
            public void testAndRespond() {
                double x = sensor.read();
                if (x > high_threshold) actuator.act(false);
            }
        };
    }
}

public class HyperbaricChamber {
    public static void main(String[] arguments) {
        PressureSensor sensor = new PressureSensor();
        OnOffActuator  actuator = new OnOffActuator();

        double low_thresh = Double.parseDouble(arguments[0]);
        double high_thresh = Double.parseDouble(arguments[1]);

        new HysteresisController(sensor, actuator,
                                low_thresh, high_thresh);

        Dispatcher.dispatch();
    }
}

```

א. ממש את החלקים החסרים של Dispatcher ושל Policy. המימוש צריך להיות כך שהשירות testAndRespond של כל עצם מטיפוס Policy (כלומר מהמחלקה Policy או ממחלקות שמרחיבות אותה) ירוץ בערך כל 10 אלפיות שניה. המימוש יכול להשתמש בשירות המחלקה Thread.sleep(long milliseconds). אסור לשנות חלקים אחרים של הקוד הנתון!

```
abstract public void testAndRespond();
}
```

ב. התוכנית הזו (HyperbaricChamber.main) משתמשת בשני עצמי Policy. השירות testAndRespond של העצמים הללו נקרא לעיתים קרובות מאוד. האם לדעתך השימוש התכוף בשירותים הללו גורם להקצאות זיכרון רבות, הקצאות שמכריחות את ה-JVM להפעיל את אוסף הזבל (garbage collector) מדי פעם? הרצה של אוסף הזבל עשויה להשהות את התגובה לשינויי לחץ, כך שהיא לא רצויה בתוכנית כזו. הסבר/י מדוע את/ה חושב/ת שהפעלת השירותים הללו גורמת או לא גורמת להקצאות זיכרון רבות. יש להתעלם מהקצאות זיכרון ב-Dispatcher.

ג. האם הפתרון שלך לחלק א' של השאלה גורם להקצאות זיכרון רבות ותכופות, ובעקבות כך להפעלה של אוסף הזבל? (התנהגות כזו לא משפיעה על הציון לסעיף א' ואין צורך לתקן את סעיף א' כך שלא יגרום להקצאת זיכרון). הסבר/י מדוע הפתרון שלך מקצה או שאינו מקצה זיכרון באופן תכוף. אם את/ה חושב/ת שהפתרון שלך לסעיף א' אכן מקצה זיכרון לעיתים תכופות, תאר/י במילים פתרון אחר שאינו גורם להקצאות זיכרון תכופות; המטרה היא לאפשר לתוכנית לרוץ לעד גם אם אוסף הזבל לא מופעל לעולם.

ד. כתוב/כתבי תנאי קדם (precondition) מתאים לשירות HyperbaricChamber.main.

ה. הבוס שלך מורה לך לבטל את תנאי הקדם של HyperbaricChamber.main ולשנות בהתאם את המימוש של השירות. למה הבוס נותן הוראה כזו, ואיזה שינויים דרושים בקוד?

ו. מתגים, כמו רכיבים פיזיים אחרים, עלולים להתקלקל ו/או להיכשל. נניח שמחליפים את המתג ששולט על המדחס במתג "חכם" שיודע לדווח האם הוא הצליח להדליק ולכבות את המדחס. שימוש במתג כזה בתוכנית דורש שינויים במחלקה OnOffActuator שדרכה התוכנית מתממשת למתג. תאר שתי (2) דרכים לשנות את המחלקה הזו כך שתתמוך במתג החכם, והסבר את היתרונות והחסרונות של כל אחת שיטה בהשוואה לשיטה השניה. אפשר להציע שיטות שדורשות שינויים במחלקות/מנשקים נוספים, אבל אפשר להתעלם משינויים שאולי דרושים בלקוחות של OnOffActuator, כלומר ב-HysteresisController, וכדומה. HyperbaricChamber

ז. המחלקה HysteresisController מיועדת לשלוט על מתג שגורם למדידות של החיישן לעלות כאשר המתג מופעל ולרדת כאשר המתג כבוי. לפעמים, ההתנהגות של המכונה הפוכה. למשל, במזגן הטמפרטורות שמודד חיישן טמפרטורה יורדות כאשר המנוע מופעל, ועולות כאשר המנוע כבוי. ממש/י מחלקה HysteresisDownController ששולטת במכשיר שבו הפעלה של המתג גורמת למדידות של החיישן לרדת. אם יש צורך במחלקות נוספות, ממשו אותן כמחלקות פנימיות. יש להשתמש בקוד קיים הרבה ככל שניתן. בפרט, אל תציע/י מימוש שמשכפל למעשה את HysteresisController הקיים, פרט להחלפה של actuator.act(true) ב- actuator.act(false) ולהיפך. זה דורש שכפול של קוד, ואנו מעוניינים להמנע משכפול כזה.

שאלה 2, 24 נקודות

בכל חלק של השאלה הזו, יש להסביר האם הקוד מתקמפל (compiles) או שהקומפיילר מדווח על שגיאה. אם לדעתך הקוד לא מתקמפל, יש להסביר מדוע. אם לדעתך הוא כן מתקמפל, תאר/י מה קורה כאשר הקוד רץ (מה הפלט, או מה החריג שנזרק, וכדומה).

```
1. double d = 1.0 / 0.0;
   if (d > 1.0)
       System.out.println("X");
   else
       System.out.println("Y");
```

```
2. int i;
   System.out.println(i);
```

```
3. int d=1;
   switch(d) {
       case 0: System.out.print("A");
              break;
       case 1: System.out.print("B");
       case 2: System.out.print("C");
       default: System.out.print("XXX");
   }
```

```
4. public class A {  
    public static void main(String[] arr) {  
        System.out.println(arr[2]);  
    }  
}
```

We run this program with the arguments "hello world".

```
5. public class A {  
    private int i;  
    public void copy(final A a, int i) { a.i = i; }  
    public static void main(String[] arg) {  
        A a = new A();  
        a.copy(a,4);  
        System.out.println(a.i);  
    }  
}
```

```
6. public class Outer {
    private int i;

    private class Inner {
        private int j = i++;
        public int g() { return j; }
    }

    public Object f(){
        Inner inner = new Inner();
        return inner;
    }
}

public class A{
    public static void main(String[] args){
        Object o = new Outer().f();
        Outer.Inner inner = (Outer.Inner) o;
        System.out.println(inner.g());
    }
}
```

שאלה 3, 20 נקודות

נתונות שתי המחלקות הבאות:

```
public class A {
    public void f(A a) {
        System.out.println("in A");
    }
}

public class B extends A {
    public static void main(String[] args) {
        B b = new B();
        A a = new A();
        b.f(b);
        b.f(a);
    }
}
```

לגבי כל אחד מהשירותים הבאים: האם השירות יכול להיות מוגדר כשירות של המחלקה B בקוד למעלה? אם לא (כלומר אם לדעתך הקוד לא יתקמפל), יש להסביר מדוע. אם כן, יש להסביר מה קורה כאשר מריצים את B.main.

1.

```
protected void f(A a) {  
    System.out.println("in B");  
}
```

2.

```
public void f(A a) {  
    System.out.println("in B");  
    throw new java.io.IOException();  
}
```

3.

```
public void f(B b) {  
    System.out.println("in B");  
}
```

4.

```
public void f(Object o) {  
    System.out.println("in B");  
}
```

שאלה 4, 20 נקודות

א. מה עושה הקוד הבא ולאיזה צורך?

```
double [][] L = new double [10][];  
double [][] U = new double [10][];  
for (int i = 0; i < 10; i++) {  
    L[i] = new double[i+1];  
    U[i] = new double[10-i];  
}
```

ב. מה מייצגים הצמתים (vertices) של עץ מיני-מקס (mini-max tree או game-search tree)?

ג. הגדרי את המונח big endian.

ד. תארי סיבה אחת שגורמת ל-servlet בג'אווה לא לפעול או לא לפעול באופן תקין, כלומר שלא על פי ההתנהגות שמוגדרת בקבצי ה-java. וה-class של ה-servlet. התשובה חייבת להיות ספיציפית ל-servlets; תשובה שתקיפה לקוד ג'אווה אחר לא תתקבל.
