

פתרון בחינה בתוכנה 1

ד"ר ליאור וולף

ליאור שפירא, אוהד ברזילי

מועד א' סמסטר ב' תשס"ח, 3 באוגוסט 2008

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 מחשבוניס או כל מכשיר אחר פרט לעט.

בהצלחה!

לשימוש הבודקים

31				7 ₄	8 ₃	8 ₂	8 ₁	1
30						15 _ב	5x3 _א	2
21					7 _{III}	7 _{II}	7 _I	3
9							3x3 _א	4
9							3x3 _א	5
100	סה"כ							

שאלה אחת

בשאלה זו נעבוד עם זרמים (Streams). זרם הוא רצף אינסופי של נתונים מטיפוס מסוים. אנו נממש זרם בעזרת אצנים (Iterators) בצורה הבאה:

```
public abstract class Stream<T> implements Iterator<T> {

    public boolean hasNext() {
        return true;
    }

    public void remove() {
        throw new RuntimeException("Unsupported Operation");
    }

}
```

1. השלימו את הקוד הבא כך שיודפס למסך הרצף האינסופי:

1, 1, 1, 1, 1, 1, 1,...

```
Stream<Integer> ones = new Stream<Integer>() {
    public Integer next() {
        return 1;
    }
};

while(true) {
    System.out.print(ones.next() + ", ");
}
```

בסעיפים הבאים נעבוד עם סוכמים (accumulators) ומסננים (filters) הפועלים על זרמים אחרים. נגדיר לשתי משפחות הטיפוסים האלה מחלקת בסיס אבסטרקטית משותפת כך:

```
public abstract class CompositeStream<T> extends Stream<T> {

    protected Stream<T> underlyingStream;

    public CompositeStream(Stream<T> underlyingStream) {
        this.underlyingStream = underlyingStream;
    }

}
```

2. סוכם (AccumulateStream) הוא זרם הפועל על זרם נתון ומייצר עבורו זרם חדש המייצג את "סכמו" של הזרם המקורי. ניתן להגדיר את משמעות פעולת הסכום ע"י מחלקת עזר המממשת את הממשק `Accumulator<T>`:

```
public interface Accumulator<T> {
    T accumulate(T t1, T t2);
}
```

בדוגמא שלפניכם אנו מגדירים את זרם המספרים הטבעיים:

1, 2, 3, 4, 5,...

ע"י הגדרת הסוכם `naturals` הפועל על הזרם `ones` מהסעיף הקודם:

```
Accumulator<Integer> sum = new Accumulator<Integer>(){
    public Integer accumulate(Integer t1, Integer t2) {
        return t1+t2;
    }
};

AccumulateStream<Integer> naturals =
    new AccumulateStream<Integer>(ones, sum);
```

הגדירו את המחלקה `AccumulateStream`:

```
public class AccumulateStream<T>
```

```
    extends CompositeStream<T>
```

```
    protected Accumulator<T> accumulator;
    protected T sum = null;

    public AccumulateStream(Stream<T> underlyingStream,
        Accumulator<T> accumulator) {
        super(underlyingStream);
        this.accumulator = accumulator;
    }

    public T next() {
        if (sum == null)
            sum = underlyingStream.next();
        else
            sum = accumulator.accumulate(sum, underlyingStream.next());

        return sum;
    }
}
```

```
}
```

שגיאות נפוצות וניקודן:

- ירשה שגויה (שהובילה לשכפול קוד): -2
- חסר טיפול מיוחד באיבר ראשון: -3
- חסר בנאי או בנאי לא תקין: -2
- אי מימוש `next`: -5

3. מסנן (FilterStream) היא מחלקה מופשטת המייצגת זרם הפועל על זרם נתון ומייצר עבורו זרם חדש המייצג תת קבוצה של הזרם המקורי. בדוגמא שלפניכם אנו מיצרים את זרם המספרים הטבעיים הזוגיים, evens, ע"י הגדרת מסנן העובד על הזרם naturals שהוגדר בסעיף הקודם:

```
Stream<Integer> evens = new FilterStream<Integer>(naturals){
    @Override
    public boolean accept(Integer element) {
        return element % 2 == 0;
    }
};
```

הגדירו את המחלקה FilterStream:

```
public abstract class FilterStream<T> extends CompositeStream<T> {
    public abstract boolean accept(T element);
    public FilterStream(Stream<T> underlyingStream) {
        super(underlyingStream);
    }
    public T next() {
        T temp;
        do {
            temp = underlyingStream.next();
        } while (!accept(temp));
        return temp;
    }
}
```

שגיאות נפוצות וניקודן:

- ירושה שגויה (שהובילה לשכפול קוד): -2
- חסרה לולאה או קריאה רקורסיבית: -3
- חסר בנאי או בנאי לא תקין: -2

4. השלימו את הקוד הבא כך ש `primes` יצביע לזרם המספרים הראשוניים הטבעיים:

2, 3, 5, 7, 11, 13, 17, 19,...

```
FilterStream<Integer> primes =
    new FilterStream<Integer>(naturals) {
        {
            underlyingStream.next(); // anonymous constructor
        }

        public boolean accept(Integer element) {
            for (int i = 2; i < element; i++) {
                if (element%i == 0)
                    return false;
            }
            return true;
        }
    };
```

שגיאות נפוצות וניקודן:

- אי מימוש אלגוריתם prime : -5
- 1 אינו ראשוני : -1
- 2 הינו ראשוני : -1

שאלה שתיים

נתון הקוד הבא:

```
import java.io.*;

abstract class PurchasePower {

    protected final double base = 500;
    protected PurchasePower successor;

    public void setSuccessor(PurchasePower successor){
        this.successor = successor;
    }

    abstract public void processRequest(PurchaseRequest request);
}

class Manager extends PurchasePower {
    private final double ALLOWABLE = 10 * base;

    public void processRequest(PurchaseRequest request ) {
        if( request.getAmount() < ALLOWABLE )
            System.out.println("Manager will approve $" + request.getAmount());
        else
            if( successor != null)
                successor.processRequest(request);
    }
}

class Director extends PurchasePower {
    private final double ALLOWABLE = 20 * base;

    public void processRequest(PurchaseRequest request ) {
        if( request.getAmount() < ALLOWABLE )
            System.out.println("Director will approve $" + request.getAmount());
        else
            if( successor != null)
                successor.processRequest(request);
    }
}

class VicePresident extends PurchasePower {
    private final double ALLOWABLE = 40 * base;

    public void processRequest(PurchaseRequest request) {
        if( request.getAmount() < ALLOWABLE )
            System.out.println("Vice President will approve $" + request.getAmount());
        else
            if( successor != null )
                successor.processRequest(request);
    }
}

class President extends PurchasePower {
    private final double ALLOWABLE = 60 * base;

    public void processRequest(PurchaseRequest request){
        if( request.getAmount() < ALLOWABLE )
            System.out.println("President will approve $" + request.getAmount());
        else
            System.out.println( "Your request for $" + request.getAmount() +
                " needs a board meeting!");
    }
}
```

```
class PurchaseRequest {

    private int number;
    private double amount;
    private String purpose;

    public PurchaseRequest(int number, double amount, String purpose){
        this.number = number;
        this.amount = amount;
        this.purpose = purpose;
    }

    public double getAmount() {
        return amount;
    }
    public void setAmount(double amt){
        amount = amt;
    }

    public String getPurpose() {
        return purpose;
    }
    public void setPurpose(String reason) {
        purpose = reason;
    }

    public int getNumber(){
        return number;
    }
    public void setNumber(int num) {
        number = num;
    }
}

class CheckAuthority {

    public static void main(String[] args) throws Exception{

        Manager manager = new Manager();
        Director director = new Director();
        VicePresident vp = new VicePresident();
        President president = new President();

        manager.setSuccessor(director);
        director.setSuccessor(vp);
        vp.setSuccessor(president);

        //enter ctrl+c to kill.
        while (true) {
            System.out.println("Enter amount.");
            System.out.print(">");
            double d = Double.parseDouble(new BufferedReader(
                new InputStreamReader(System.in)).readLine());
            manager.processRequest(new PurchaseRequest(0, d, "General"));
        }
    }
}
```

א. מה יודפס אם תורץ המתודה main והמשתמש יכניס את הערכים :

500
5000
11000
30000
20000

(כלומר יכניס 500 ילחץ Enter יכניס 5000 וכולי...)

```
>Enter amount.  
>500  
Manager will approve $500.0  
Enter amount.  
>5000  
Director will approve $5000.0  
Enter amount.  
>11000  
Vice President will approve $11000.0  
Enter amount.  
>30000  
Your request for $30000.0 needs a board meeting!  
Enter amount.  
>20000  
President will approve $20000.
```


ב. כתבו את הקוד של המחלקה PurchasePower ושל המחלקות היורשות כך ששכפול הקוד ימוזער. אם יש צורך בשינוי CheckAuthority ציינו גם אותו:

```

abstract class PurchasePower {

    protected final static double base = 500;
    protected final double ALLOWABLE;
    protected String positionName;
    protected PurchasePower successor;

    public PurchasePower(final double multiplier, String positionName) {
        ALLOWABLE = base * multiplier;
        this.positionName = positionName;
    }

    public void setSuccessor(PurchasePower successor) {
        this.successor = successor;
    }

    public void processRequest(PurchaseRequest request) {
        if (request.getAmount() < ALLOWABLE)
            System.out.println(positionName + " will approve $"
                + request.getAmount());
        else
            handleReject(request);
    }

    protected void handleReject(PurchaseRequest request) {
        if (successor != null)
            successor.processRequest(request);
    }
}

class Manager extends PurchasePower {

    public Manager() {
        super(10, "Manager");
    }
}

class Director extends PurchasePower {
    public Director() {
        super(20, "Director");
    }
}

class VicePresident extends PurchasePower {
    public VicePresident() {
        super(40, "VicePresident");
    }
}

class President extends PurchasePower {
    public President() {
        super(60, "President");
    }

    protected void handleReject(PurchaseRequest request) {
        System.out.println("Your request for $" + request.getAmount()
            + " needs a board meeting!");
    }
}

```

שגיאות נפוצות וניקודן:

- יצירת מופע ממחלקה מופשטת: -5
- העברת לוגיקה של בעלי התפקידים ל main (שמות, מספרים)
- שינוי התנהגות של הקוד המקורי: manager מכנס ישיבת הנהלה, president מעביר את הטיפול הלאה -5
- בדיקה במחלקת הבסיס האם היא president: -3 (הקוד במחלקת בסיס אמור להיות כללי!)

שאלה שלוש

בסעיף זה אתם מתבקשים לממש 3 גרסאות לפונקצית שירות אשר מוסיפה נתונים חדשים בהתחלת קובץ (בלי למחוק את שאר הנתונים).

הגרסאות נבדלות זו מזו בתנאי הקצה אשר מנוסחים בתחביר design by contract. שימו לב: עליכם להקפיד על כללי תיכון לפי חוזים. למשל, לא לבצע בדיקות מיותרות.

יש למזער את שכפול הקוד בין הגרסאות . ניתן להוסיף לשם כך שרותי עזר או להשתמש בקריאות הדדיות בין הגרסאות השונות.

```
public class Prepend {

    /**
     * @param filename The name of the file that data should be prepended to
     * @param data The data to be prepended
     *
     * @pre new File(filename).exists()
     * @pre new File(filename).canRead()
     * @pre new File(filename).canWrite()
     *
     * @post filename starts with the string data and then its original content
     *
     * @throws IOException
     */
    public static void prepend1(String filename, String data) throws IOException {

        File file = new File(filename);
        Scanner s = new Scanner(file);
        StringBuffer content = new StringBuffer(data);

        while (s.hasNextLine()) {
            content.append(s.nextLine() + '\n');
        }

        writeToExistingFile(file, content.toString());
    }
}
```

```
public static void writeToExistingFile(File f, String data) throws IOException {
    FileWriter fw = new FileWriter(f);
    fw.write(data);
    fw.close();
}
```

שגיאות נפוצות וניקודן:

- אי שימוש ב streams כלל: -7

```

/**
 * @param filename The name of the file that data should be prepended to
 * @param data The data to be prepended
 *
 * @post $prev( ! (new File(filename).exists()))
 *          $implies new File(filename).exists()
 *
 * @throws IOException
 */
public static void prepend2(String filename, String data) throws IOException {
    File file = new File(filename);
    if (!file.exists()){
        file.createNewFile();
        writeToExistingFile(file, data);
        return;
    }

    if (file.canRead() && file.canWrite())
        prepend1(filename, data);
    else
        throw new IOException("Read/Write permission problem");
}

```

שגיאות נפוצות וניקודן:

- שכפול קוד: -3
- אי יצירת קובץ אם לא קיים -2
- קריאה ל- prepend1 ללא בדיקת החוזה שלה: -2

```

/**
 * @param filename The name of the file that data should be prepended to
 * @param data The data to be prepended
 *
 * @throws FileNotFoundException
 */
public static void prepend3(String filename, String data)
    throws FileNotFoundException {
    try{
        prepend2(filename, data);
    }
    catch (IOException e) {
        if (e instanceof FileNotFoundException)
            throw (FileNotFoundException)e;
        else
            System.err.println("Unexpected IO Error occured");
    }
}

```

שגיאות נפוצות וניקודן:

- שכפול קוד: -3
- זריקת חריג אסור: -3

שאלה ארבע

נתונים הקבצים הבאים:

```
1. package org;
2. public class Robot { }
```

```
1. package org.ex;
2. public class Pet { }
```

```
1. package org.ex.why;
2. public class Dog { int foo = 5; }
```

ונתון הקוד הלא שלם הבא:

// הכנס קוד כאן

```
public class MyClass {

    Robot r;
    Pet p;
    Dog d;

    void go() {
        int x = d.foo;
    }

}
```

איזה מההצהרות הבאות יש להכניס ל MyClass במקום ההערה כדי שהמחלקה תתקמפל? הקיפו בעיגול תשובה אחת או יותר לפי הצורך (כל ההצהרות שבחרתם מוכנסות למקום המסומן בזמנית בזו אחר זו)

A) package org;

B) import org.*;

C) package org.*;

D) package org.ex;

E) import org.ex.*;

F) import org.ex.why;

G) package org.ex.why;

H) package org.ex.why.Dog;

<p>תשובה נכונה שסומנה: +3 תשובה שגויה שסומנה: -3 לא ניתן היה לקבל ציון מתחת לאפס בשאלה</p>

שאלה חמש

נתון הקוד הבא:

```
1. import java.util.*;
2. public class Fruits {
3.     public static void main(String [] args) {
4.         Set<Citrus> c1 = new TreeSet<Citrus>();
5.         Set<Orange> o1 = new TreeSet<Orange>();
6.         bite(c1);
7.         bite(o1);
8.     }
9.     // insert code here
10. }
11. class Citrus { }
12. class Orange extends Citrus { }
```

אילו מההצהרות הבאות תגרום לתוכנית להתקמפל בהצלחה אם תוכנס בשורה 9?
הקיפו בעיגול את כל התשובות הנכונות (כל תשובה נכונה תוכנס בנפרד בתוכנית משלה)

- A) `public static void bite(Set<?> s) { }`
- B) `public static void bite(Set<Object> s) { }`
- C) `public static void bite(Set<Citrus> s) { }`
- D) `public static void bite(Set<? super Citrus> s) { }`
- E) `public static void bite(Set<? super Orange> s) { }`
- F) `public static void bite(Set<? extends Citrus> s) { }`
- G) עקב שגיאות אחרות בקוד התוכנית לא תתקמפל כלל

תשובה נכונה שסומנה: +3
תשובה שגויה שסומנה: -3
לא ניתן היה לקבל ציון מתחת לאפס
בשאלה

בהצלחה!!