פתרון מוצע לבחינת מה"ט במבני נתונים ב ***JAVA***

מועד א' תשפ"ב, מרץ 2202
מחבר: ד"ר אודי לביא, מכללת אורט רחובות

חלק א'

שאלה מס. 1

מחלקה ראשונה – מחלקת Node:

```
public class Node<T>
{
  private T value;
  private Node<T> next;

  public Node (T value)
  {
    this.value=value;
    this.next=null;
  }
  public Node(T value, Node<T> next)
  {
    this.value=value;
    this.next=next;
  }
  public Node<T> getNext()
  {
    return this.next;
  }
  public void setNext(Node<T> next)
  {
    this.next=next;
  }
  public T getValue()
  {
    return this.value;
  }
  public void setValue(T value)
  {
    this.value=value;
  }
  public boolean hasNext()
  {
    return this.next!=null;
  }
```

```
}
```

מחלקה שניה (ובה המטודות כולל הMain):

```
public class Rashi {

    public static void first(Node<Integer> chain){
        if(chain == null)
            return;
        Node<Integer> copy = new Node<>(chain.getValue());
        copy.setNext(chain.getNext());
        chain.setNext(copy);
        first(copy.getNext());
    }

    public static void second(Node<Integer> chain){
        if(chain == null)
            return;
        Node<Integer> copy = new Node<>(chain.getValue());
        Node<Integer> index = copy;
        chain = chain.getNext();
        while(chain.hasNext()){
            index.setNext(new Node<>(chain.getValue()));
            index = index.getNext();
            chain = chain.getNext();
        }
        index.setNext(new Node<>(chain.getValue()));
        chain.setNext(copy);
    }

    public static void print(Node<Integer> chain){
        System.out.println("chain: ");
        while (chain != null){
            System.out.print(chain.getValue() + " ");
            chain = chain.getNext();
        }
    }
```

```
        public static void main(String[] args) {

                Node<Integer> n1 = new Node<>(1);
                  Node<Integer> n2 = new Node<>(2);
                  Node<Integer> n3 = new Node<>(3);
                  Node<Integer> n4 = new Node<>(4);

                  n1.setNext(n2);
                  n2.setNext(n3);
                  n3.setNext(n4);

                  System.out.println("Before:");
                  print(n1);
                  System.out.println();
                  System.out.println("After:");
//                first(n1);
                  second(n1);
                  print(n1);
        }

}
```

פלט עבור סעיף א' - (הפונקציה first עובדת):

```
Before:
chain:
1 2 3 4
After:
chain:
1 1 2 2 3 3 4 4
```

פלט עבור סעיף ב' - (הפונקציה second עובדת):

```
Before:
chain:
1 2 3 4
After:
chain:
1 2 3 4 1 2 3 4
```

שאלה מס. 2

2.א.

מחלקת Node (קיימת).

מחלקת Stack:

```
public class Stack<T>
{
  private Node <T>first;
  public Stack()
  {
    this.first= null;
  }
  public boolean isEmpty()
  {
    return this.first==null;
  }
  public void push(T x)
  {
    this.first=new Node <T>(x, this.first);
  }
  public T pop()
  {
    T x = this.first.getValue();
    this.first=this.first.getNext();
    return x;
  }
  public T top()
  {
    return this.first.getValue();
  }
  public String toString()
  {
    if (this.isEmpty())
    {
      return("\t[] (Stack is Empty!)");
    }
    String str="[";
    Node <T> pos =this.first;
    while (pos.hasNext()) {
      str=str+pos.getValue()+",";
      pos=pos.getNext();
    }
    str = str +pos.getValue()+"]";
    return str;
  }
```

```
  public int size()
  {
    int counter = 0;
    Node<T> temp = this.first;
    while (temp != null)
    {
      counter++;
      temp = temp.getNext();
    }
    System.out.println("num of elenemts in stack :");
    return counter;
  }
}
```

המחלקה הראשית:

```
public class Ex2 {

    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        stack.push(6);
        stack.push(12);
        stack.push(2);
        stack.push(10);
        stack.push(7);
        stack.push(5);

        sort(stack);
        System.out.println(stack);
    }

    public static void sort(Stack<Integer> stack){
        if(stack == null || stack.isEmpty())
            return;
        double sum = 0;
        int counter = 0;

        Stack<Integer> temp = new Stack<>();
        while (!stack.isEmpty()){
            counter++;
            sum += stack.top();
            temp.push(stack.pop());
        }
```

```
        double avg = sum / counter;
        Stack<Integer> biggerThanAvg = new Stack<>();
        Stack<Integer> smallerOrEqualToAvg = new Stack<>();

        while (!temp.isEmpty()){
            if(temp.top() > avg)
                biggerThanAvg.push(temp.pop());
            else
                smallerOrEqualToAvg.push(temp.pop());
        }

        while (!smallerOrEqualToAvg.isEmpty()){
            stack.push(smallerOrEqualToAvg.pop());
        }

        while (!biggerThanAvg.isEmpty()){
            stack.push(biggerThanAvg.pop());
        }
    }
}
```

2.ב. – סיבוכיות O(n) . כל איבר מוצא מהמחסנית הראשית למחסנית זמנית O(n) , מהמחסנית המשנית האיברים מחולקים לפי גודלם ביחס לממוצע O(n) , האיברים מוחזרים בסדר הרצוי למחסנית הראשית O(n), סך הכל 3 * O(n) = O(n)

<u>שאלה מס. 3</u>

3.1.1 נכון, אחרת לא ניתן לקרוא לה ע"י h.
3.1.2 לא נכון, לבנאי ניתן להעביר גם אובייקטים כארגומנטים.
3.1.3 נכון, רק אובייקטים יכולים להפעיל על עצמם מתודות.
3.1.4 לא נכון, לא ניתן להעביר פונקציות כפרמטר.
3.1.5 נכון, ניתן לייצר אובייקט מטיפוס מסוים ולהתייחס אליו כאילו הוא אב קדמון של אותו הטיפוס.
3.1.6 לא נכון, הורשה לא יכולה להיות מעגלית.

3.2 תשובה מספר 2.

<u>שאלה מס. 4</u>

4.1.1 לא נכון, יש את הבנאי הדיפולטיבי הריק שנוצר באופן אוטומטי.
4.1.2 לא נכון, המחלקה A לא יורשת כלל מ B.
4.1.3 נכון, B הוא בעצם A עם תוספות.
4.1.4 לא נכון, המחלקה B לא יכולה לגשת לתכונה myVal כיוון שהשדה מוגדר כ private
4.1.5 לא נכון, מחלקה לא יכולה לגשת לתכונות של היורשים שלה.

4.2 public B(int myVal, double x) {super(myVal); this.x = x;}

4.3.1 לא תקין, שגיאת קומפילציה, one הוא אובייקט מטיפוס A ולו אין פונקציה בשם goodCode(),
העובדה שמאחורי הקלעים בעצם קיים אובייקט מטיפוס B לא ידועה לקומפיילר ולכן תהיה שגיאה.
4.3.2 לא תקין, שגיאת קומפילציה, two הוא אובייקט מטיפוס A ולו אין פונקציה בשם goodCode()
4.3.3 תקין.
4.3.4 לא תקין, שגיאת זמן ריצה, two הוא מטיפוס A, בקוד אנחנו אומרים לקומפיילר שיסמוך עלינו
ובעצם זה אובייקט מטיפוס B, לכן הקומפיילר לא ייתן שגיאה ובזמן ריצה תקרה השגיאה.
4.4 ניתן לקבל מידע כך:
במחלקה B יש להוסיף מונה בדומה לזה שיש במחלקה A:

```
public static int countB = 0;
```

במחלקה A להוסיף לפונקציה func() את שורת הקוד:

```
countA++;
```

במחלקה B יש לכתוב פונקציה שתדרוס את func() ובה יה הקוד זהה לזה שכתוב ב A פרט לשורה שמעלה
את המונה של A (מה שהוספנו הרגע לA), ובמקומה נכתוב:

```
countB++;
```

השדות הללו סטטיים וציבוריים ולכן ניתן לקרוא להם מהתוכנית הראשית ולראות אתא ערכם, בנוסף, כל
מופע של האובייקט יעלה את המונה הרלוונטי.
במקרה של תכנות מקבילי, יש להוסיף את המילה synchronize בחתימה של הפונקציה func כדי שהמונים
יתעדכנו בצורה תקינה במקרה של כמה תהליכים שמנסים להעלות את אותו המונה.

<u>שאלה מס. 5</u>

5.1

```
public abstract class Animal
public class Mammal extends Animal
public class Bat extends Mammal implements ICanFly
public class Fish extends Animal implements IHasEggs
public class Bird extends Animal implements IHasEggs, ICanFly
```

5.2.1 חוקי.
5.2.2 לא חוקי, IHasEggs הוא ממשק ולכן לא ניתן לייצר אובייקט כזה.
5.2.3 תקין.
5.2.4 תקין.
5.2.5 תקין.
5.2.6 לא תקין, שגיאת קומפילציה היות ול IHasEggs אין פונקציה בשם fly

5.3

```
public int countCanFly(Animal[] animals){
  int counter = 0;
  for(int i = 0; i < animals.length; i++) {
    if(animals[i] instanceof ICanFly)
      counter++;
  return counter;
}
```

שאלה מס. 6

פתרון בקוד:

```
public class Ex6 {

    public static void main(String[] args) {
        Node<Integer> n1 = new Node<>(80);
        Node<Integer> n2 = new Node<>(40);
        Node<Integer> n3 = new Node<>(90);
        Node<Integer> n4 = new Node<>(-1);
        Node<Integer> n5 = new Node<>(95);
        Node<Integer> n6 = new Node<>(-1);
        Node<Integer> n7 = new Node<>(85);
        Node<Integer> n8 = new Node<>(80);
        Node<Integer> n9 = new Node<>(-1);

        n1.setNext(n2);
        n2.setNext(n3);
        n3.setNext(n4);
        n4.setNext(n5);
        n5.setNext(n6);
        n6.setNext(n7);
        n7.setNext(n8);
        n8.setNext(n9);

        System.out.println("A");
        Node<Double> doubleNode = averageList(n1);

        while (doubleNode != null){
            System.out.println(doubleNode.getValue());
            doubleNode = doubleNode.getNext();
        }

        System.out.println("B");
        print(n1);
    }
```

```
    public static Node<Double> averageList(Node<Integer> lst){
        int counter = 0;
        double sum = 0;
        Node<Double> newLst = new Node<>(0.0);
        Node<Double> index = newLst;
        while (lst != null){
            if(lst.getValue() == -1){
                index.setNext(new Node<>(sum / counter));
                index = index.getNext();
                counter = 0;
                sum = 0;
            }
            else{
                counter++;
                sum += lst.getValue();
            }
            lst = lst.getNext();
        }
        return newLst.getNext();
    }

    public static void print(Node<Integer> lst){
        int counter = 0;
        double sum = 0;
        int studentId = 1;
        int minGrade = 101;
        while (lst != null){
            if(lst.getValue() == -1){
                if(counter > 1){
                    System.out.println(studentId + " " + sum / counter + " " + (sum
- minGrade) / (counter - 1));
                }
                else{
                    System.out.println(studentId + " " + sum / counter + " " + sum /
counter);
                }
                counter = 0;
                sum = 0;
                minGrade = 101;
                studentId++;
            }
            else{
                if(lst.getValue() < minGrade)
```

```
                minGrade = lst.getValue();
            counter++;
            sum += lst.getValue();
        }
        lst = lst.getNext();
    }
  }
}
```

שאלה מס. 7

א+ב':

```
public class Ex7 {

   public static void main(String[] args) {
      Stack<Integer> s0 = new Stack<>();
      s0.push(2);
      s0.push(5);
      s0.push(-1);
      s0.push(8);

      Stack<Integer> s1 = new Stack<>();
      s1.push(10);
      s1.push(-38);
      s1.push(7);

      Stack<Integer> s2 = new Stack<>();
      s2.push(-900);

      TStack tStack = new TStack(s0, s1, s2);

      System.out.println("Before Sorting:");
      System.out.println(tStack);

      System.out.println("After Sorting:");
      tStack.sort();

      System.out.println(tStack);

   }

   public static class TStack {

      private Stack<Integer> s0;
      private Stack<Integer> s1;
      private Stack<Integer> s2;
```

```
    public TStack() {
        s0 = new Stack<>();
        s1 = new Stack<>();
        s2 = new Stack<>();
    }

    public TStack(Stack<Integer> s0, Stack<Integer> s1, Stack<Integer> s2) {
        this.s0 = s0;
        this.s1 = s1;
        this.s2 = s2;
    }

    public void move(int from, int to){
        if((from == to) || (from == 0 && to == 2) || (from == 1 && to == 0) || (from == 2
&& to == 1)){
            // illegal move
            return;
        }
        if(getStackById(from).isEmpty())
            return;
        getStackById(to).push(getStackById(from).pop());
    }

    private Stack<Integer> getStackById(int id){
        return switch (id){
            case 0 -> s0;
            case 1 -> s1;
            case 2 -> s2;
            default -> null;
        };
    }

    public boolean bigOrEqual(int from, int toCompare){
        if(getStackById(from).isEmpty())
            return false;
        if(getStackById(toCompare).isEmpty())
            return true;
        return getStackById(from).top() >= getStackById(toCompare).top();
    }

    public boolean isEmpty(int stackId){
        return getStackById(stackId).isEmpty();
    }

    public void maximum(){
        while (!isEmpty(0)){
            if(bigOrEqual(0, 1)){
                //move to s1
                move(0, 1);
            }
```

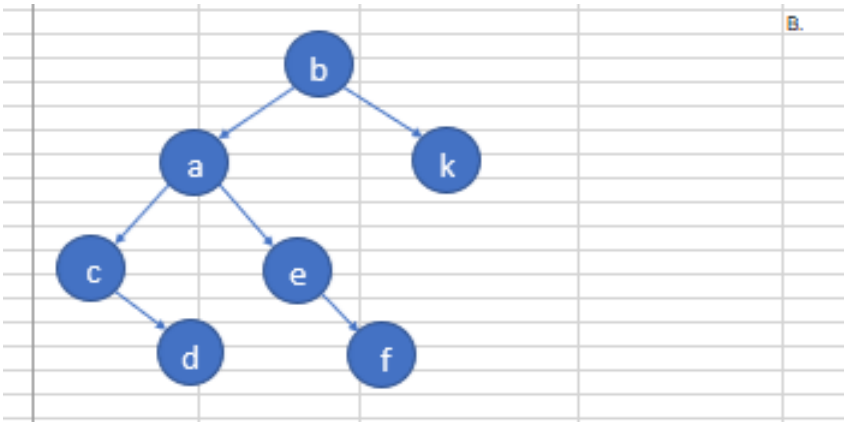
```
            else{
                //move to s2
                move(0, 1);
                move(1, 2);
            }
        }
    }

    public void sort(){
        //move all to s0
        while (!isEmpty(1)){
            move(1, 2);
        }
        while (!isEmpty(2)){
            move(2, 0);
        }


        while (!isEmpty(0)){
            //put the biggest from s0 on top of s1
            maximum();

            //move all the small items back to s0 until we see the last biggest
            while (!s2.isEmpty() && !bigOrEqual(2, 1)){
                move(2, 0);
            }

            //move the current biggest to s2
            move(1, 2);

            //move all the items from s1 to s0
            while (!isEmpty(1)){
                move(1, 2);
                move(2, 0);
            }
        }
    }

    @Override
    public String toString() {
        return "TStack{" +
                "s0=" + s0 +
                ", s1=" + s1 +
                ", s2=" + s2 +
                '}';
    }
  }
}
```

7.3 הסיבוכיות היא O(n*n) היות ועבור כל איבר במחסנית שברגע נתון הוא המקסימלי (אחרי שבודדנו את המקסימלי לפניו) יש לעשות n השוואות עם שאר איברי המחסנית.
במחסנית יש n איברים, כפול n השוואות והגענו ל n*n.

שאלה מס. ***8***

```
public static void mystery(BinNode<Character> t)
        0 if (t != null){
                1 System.out.print (t.getValue() + " ");
                2 Mystery (t.getLeft());
                3 System.out.print (t. getValue ()+ " ");
                4 Mystery (t. getRight());
                5 System.out.print (t.getValue()+ " ");
        }
}
```

			abbdddbacefffeecca		A:
0	TRUE	b	null	d	
1	TRUE	b	null	d	b
2	TRUE	b	null	d	
0	FALSE	null			
3	TRUE	b	null	d	b
4	TRUE	b	null	d	
0	TRUE	d	null	null	
1	TRUE	d	null	null	d
2	TRUE	d	null	null	
0	FALSE	null			
3	TRUE	d	null	null	d
4	TRUE	d	null	null	
0	FALSE	null			
5	TRUE	d	null	null	d
5	TRUE	b	null	d	b
3	TRUE	a	b	c	a
4	TRUE	a	b	c	
0	TRUE	c	e	null	
1	TRUE	c	e	null	c
2	TRUE	c	e	null	
0	TRUE	e	f	null	
1	TRUE	e	f	null	e
2	TRUE	e	f	null	
0	TRUE	f	null	null	
1	TRUE	f	null	null	f
2	TRUE	f	null	null	
0	FALSE	null			
3	TRUE	f	null	null	f
4	TRUE	f	null	null	
0	FALSE	null			
5	TRUE	f	null	null	f
3	TRUE	e	f	null	e
4	TRUE	e	f	null	
0	FALSE	null			
5	TRUE	e	f	null	e
3	TRUE	c	e	null	c
4	TRUE	c	e	null	
0	FALSE	null			
5	TRUE	c	e	null	c
5	TRUE	a	b	c	a



שאלה מס. ***9***
רצ"ב קוד:

```
import java.util.Date;

public class Ex9 {

    public class Store{

        //list of buys
        private Node<Buy> firstBuyNode;
        //number of buys in the list
        private int numberOfBuys;

        public Store(Node<Buy> firstBuyNode, int numberOfBuys) {
            this.firstBuyNode = firstBuyNode;
            this.numberOfBuys = numberOfBuys;
        }

        public void print(String creditNum){
            Node<Buy> temp = firstBuyNode;
            while (temp != null){
                Buy buy = temp.getValue();
                Node<Payment> paymentNode = buy.getFirstPaymentNode();
                while (paymentNode != null){
                    Payment payment = paymentNode.getValue();
                    if(payment instanceof Credit &&
((Credit)payment).getCardId().equals(creditNum)){
                        System.out.println(buy);
                        break;
                    }
```

```
                paymentNode = paymentNode.getNext();
            }
            temp = temp.getNext();
        }
    }

    public int cashPayments(){
        int counter = 0;
        Node<Buy> temp = firstBuyNode;
        while (temp != null){
            Buy buy = temp.getValue();
            if(buy.getNumberOfPaymentMethods() == 1 &&
buy.getFirstPaymentNode().getValue() instanceof Cash){
                counter++;
            }
            temp = temp.getNext();
        }

        return counter;
    }
  }

  public class Buy{
    private Date date;
    private double price;
    private int numberOfPaymentMethods;
    private Node<Payment> firstPaymentNode;

    public Buy(Date date, double price, int numberOfPaymentMethods,
Node<Payment> firstPaymentNode) {
        this.date = date;
        this.price = price;
        this.numberOfPaymentMethods = numberOfPaymentMethods;
        this.firstPaymentNode = firstPaymentNode;
    }

    public Node<Payment> getFirstPaymentNode() {
        return firstPaymentNode;
    }

    public int getNumberOfPaymentMethods() {
        return numberOfPaymentMethods;
    }
```

```
    public boolean check() {
        double sum = 0;
        Node<Payment> temp = firstPaymentNode;

        while (temp != null){
            sum += temp.getValue().getPrice();
            temp = temp.getNext();
        }
        return sum == price;
    }
}

public abstract class Payment{
    private double price;

    public Payment(double price) {
        this.price = price;
    }

    public double getPrice() {
        return price;      }    }

public class Cash extends Payment{

    public Cash(double price) {
        super(price);
    }
}

public class Credit extends Payment{

    private String cardId;
    private Date endDate;
    private Date payDate;

    public Credit(double price, String cardId, Date endDate, Date
payDate) {
        super(price);
        this.cardId = cardId;
        this.endDate = endDate;
        this.payDate = payDate;
    }
```

```
        public String getCardId() {
            return cardId;
        }
    }

    public class Cheque extends Payment{

        private String chequeId;
        private String bankName;
        private Date chequeDate;

        public Cheque(double price, String chequeId, String bankName, Date
chequeDate) {
            super(price);
            this.chequeId = chequeId;
            this.bankName = bankName;
            this.chequeDate = chequeDate;
        }
    }
}
```

שאלה מס. **10**

רצ"ב קוד:

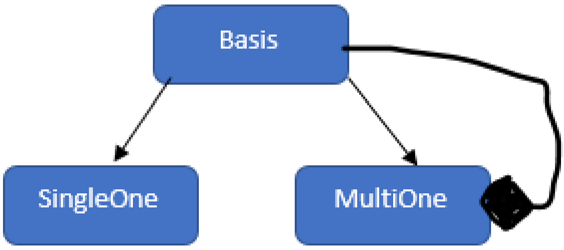
```
public class Ex10 {

    public class Advert {

        private int length;
        private String product;
        private String company;
        private double price;

        public int getLength() {
            return length;
        }

        public double getPrice() {
            return price;
        }
    }
```

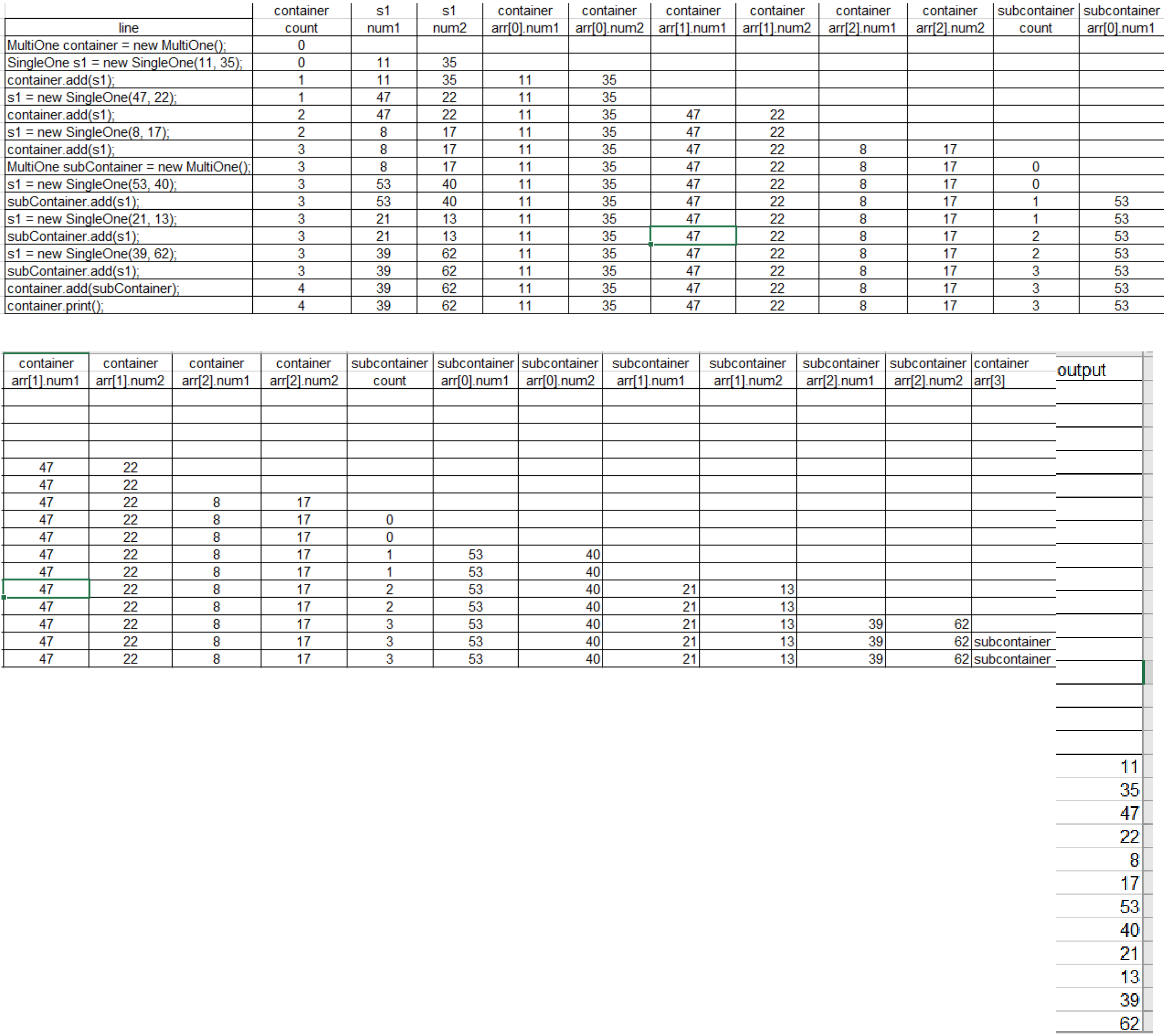
```
public class AdvertHour {

    // hour id (0 - 23)
    private int hour;
    // list of adverts
    private Node<Advert> firstAdvertNode;
    //number of adverts
    private int numberOfAdverts;
    //time left
    private int timeLeft;


    public AdvertHour(int hour) {
        this.hour = hour;
        firstAdvertNode = null;
        numberOfAdverts = 0;
        timeLeft = 15 * 60;
    }

    public int freeTime(){
        return timeLeft;
    }

    public boolean isPossible(Advert adv){
        return numberOfAdverts < 15 && adv.getLength() < timeLeft;
    }

    public void addAdvert(Advert adv){
        if(isPossible(adv)){
            if(firstAdvertNode == null){
                firstAdvertNode = new Node<>(adv);
                return;
            }
            Node<Advert> temp = firstAdvertNode;
            while (temp.hasNext()){
                temp = temp.getNext();
            }
            temp.setNext(new Node<>(adv));
            numberOfAdverts++;
            timeLeft -= adv.getLength();
        }
    }
```

```
        // this function will help us with benefitDay we will write in
ManageDay class
        public double getBenefit(){
            double sum = 0;
            Node<Advert> temp = firstAdvertNode;
            while (temp != null){
                sum += temp.getValue().price;
                temp = temp.getNext();
            }
            return sum;
        }
    }
public class ManageDay{

        // hold 24 adverts hours (0 - 23)
        private AdvertHour[] advertHours;

        // list of adverts to add
        private Node<Advert> advertsList;

        public ManageDay(Node<Advert> advertsList) {
            advertHours = new AdvertHour[24];
            this.advertsList = advertsList;
        }

        public double benefitDay(){
            double sum = 0;
            for (int i = 0; i < 24; i++) {
                sum += advertHours[i].getBenefit();
            }
            return sum;
        }
    }
}
```

שאלה מס. 11

11.א.





line	container count	s1 num1	s1 num2	container arr[0].num1	container arr[0].num2	container arr[1].num1	container arr[1].num2	container arr[2].num1	container arr[2].num2	subcontainer count	subcontainer arr[0].num1
MultiOne container = new MultiOne();	0										
SingleOne s1 = new SingleOne(11, 35);	0	11	35								
container.add(s1);	1	11	35	11	35						
s1 = new SingleOne(47, 22);	1	47	22	11	35						
container.add(s1);	2	47	22	11	35	47	22				
s1 = new SingleOne(8, 17);	2	8	17	11	35	47	22				
container.add(s1);	3	8	17	11	35	47	22	8	17		
MultiOne subContainer = new MultiOne();	3	8	17	11	35	47	22	8	17	0	
s1 = new SingleOne(53, 40);	3	53	40	11	35	47	22	8	17	0	
subContainer.add(s1);	3	53	40	11	35	47	22	8	17	1	53
s1 = new SingleOne(21, 13);	3	21	13	11	35	47	22	8	17	1	53
subContainer.add(s1);	3	21	13	11	35	47	22	8	17	2	53
s1 = new SingleOne(39, 62);	3	39	62	11	35	47	22	8	17	2	53
subContainer.add(s1);	3	39	62	11	35	47	22	8	17	3	53
container.add(subContainer);	4	39	62	11	35	47	22	8	17	3	53
container.print();	4	39	62	11	35	47	22	8	17	3	53

container arr[1].num1	container arr[1].num2	container arr[2].num1	container arr[2].num2	subcontainer count	subcontainer arr[0].num1	subcontainer arr[0].num2	subcontainer arr[1].num1	subcontainer arr[1].num2	subcontainer arr[2].num1	subcontainer arr[2].num2	container arr[3]
47	22										
47	22										
47	22	8	17								
47	22	8	17	0							
47	22	8	17	0							
47	22	8	17	1	53	40					
47	22	8	17	1	53	40					
47	22	8	17	2	53	40	21	13			
47	22	8	17	2	53	40	21	13			
47	22	8	17	3	53	40	21	13	39	62	
47	22	8	17	3	53	40	21	13	39	62	subcontainer
47	22	8	17	3	53	40	21	13	39	62	subcontainer

output
11
35
47
22
8
17
53
40
21
13
39
62