פתרון לשאלון 97105 - מבני נתונים ותכנות מונחה עצמים JAVA - בחינה לדוגמה

שאלה מס' 1

```
public class Card
{
     private int number;
     private String color;
     private int shape;
     public Card (int number, int shape){
          this.number = number;
          this.shape =shape;
          if(shape==1 || shape ==2)
               this.color = "Red";
          else
               this.color = "Black";
     }
     public Card (String color){
          this.number = 15;
          this.shape = 5;
          this.color = color;
     }

     public boolean isSame (Card other){

          if(this.color.equals(other.color)&&
(this.number==other.number))
              return true;
          if(this.number == 15 || other.number == 15)
                if(this.color.equals(other.color))
                 return true;

          return false;

     }
}


import java.util.Random;
public class Pack
{  public static Random rnd = new Random();
```

```
private Card[]cards= new Card[54];

public  Pack (){
     int p=0;
     int f;
     String st;
  int klaf = 0;
         for(int num = 2;num<=14;num ++){
           for( f = 1;f<=4;f++){
             if(f==1 || f == 2)
                   st = "Red";
             else
                   st = "Black";
         }
        cards[klaf]= new Card(num,f);
         klaf++;
     }
    }
    cards[53]= new Card("Red");
    cards[54]= new Card("Black");
}
```

שאלה 2

```
import unit4.collectionsLib.*;
public class Q2
{
      public static boolean ifFound (Stack<Integer>s, int num){
          boolean found = false;
          int x;
          int y;
          Stack<Integer>temp= new Stack<Integer>();
          while(!s.isEmpty()){
                x= s.pop();
                temp.push(x);
                if(x==num)
                     found = true;
          }
                while(!temp.isEmpty())
                     s.push(temp.pop());
          return found;
         }
    public static boolean ok(Stack<Integer>s){
      while(!s.isEmpty()){
       int x = s.pop();
       if(ifFound(s,x))
            return false;
      }
      return true;
    }
     public static void main(String[] args)
     {
       Stack<Integer> s1 = new Stack<Integer> ();
       s1.push(5);
       s1.push(6);
       s1.push(7);
       s1.push(9);
       s1.push(9);
       System.out.println(ok(s1));
     }
}
```

שאלה 3

נתונות שתי המחלקות Flower ו-Rose

```
public class Flower
{
  protected  int  height;
  public  Flower (int  val)    { height = val; }
  public  int  getHeight()     { return height; }
}
public class Rose extends Flower
 {
  private  String  color;
  public Rose(int val, String col)
    {
super(val);
        color=col;
    }
  public  boolean  validHeight()
    {
       return height > 10 && height < 30;
    }
 }
```

א.

1. לא Flower לא יורשת מ- Rose
2. כן
3. כן, התכונה מוגדרת כ- protected
4. לא

ב. לפניך קטע קוד מהתוכנית הראשית:

```
Flower  first = new Rose (20, "RED");
Flower  second = new Flower (93);
```

בעבור כל אחת מההוראות שלפניך קבע אם היא תקינה או אינה תקינה.
אם היא אינה תקינה, נמק את קביעתך וכתוב אם זו שגיאת ריצה או שגיאת הידור (קומפילציה).

1. `boolean b = first. validHeight();`

first זו הפנייה מסוג Flower, ולא לא ניתן לבצע את זימון validHeight() שהיא פעולה של המחלקה Rose

2. `boolean b = second. validHeight();`

לא, second הוא הפנייה של המחלקה Flower ועצם מסוג המחלקה Flower לכן לא ניתן את הפעולה validHeight

3. `boolean b = ((Rose)first). validHeight();`

כן, אפשרי, כי first מצביע על עצם מסוג Rose וניתר להמיר אותו כלפי "מטה"

4. `boolean b = ((Rose)second). validHeight();`

לא second מצביע על עצם מסוג Flower ולא ניתן להמירו להפנייה המצביעה על Rose

כתוב פעולה המקבלת מערך עצמים מטיפוס Object. על הפעולה להדפיס כמה עצמים הם מטיפוס **Rose**, כמה עצמים מטיפוס **Flower** ואינו מטיפוס **Rose** וכמה עצמים הם לא מטיפוס **Flower**.

```
public static void print(Object[] data){
   int r=0;
   int f = 0;
  int n =0;
   for(int p=0;p<data.length;p++){
     if(data[p] instanceof Rose)
       r++;
     if(data[p] instanceof Flower && !(data[p] instanceof Flower))
       f++;
     if(!(data[p] instanceof Flower))
     n++;
     }
 }
```

שאלה 4

א:

נתונה הפעולה paint במחלקה Square

```
public void paint(double scale, String color){..}
```

אילו מהחתימות הבאות מהוות העמסה (overloading) חוקית של הפעולה paint?

```
1. public int paint (double y, String x)
```

זו לא חוקי שתי הפעולות עם אותה חתימה, כי סוג הפרמטרים זהים ושם הפונקציה זהה ,וזו דריסה ולא העמסה

```
2. public void paint (int x)
```

זו העמסה חוקית , ערך ההחזרה זהה והפרמטרים שונים

```
3. public double paint (double a, char b)
```

זו העמסה חוקית, כי סוג הפרמטרים שונים

```
private void paint(double x, String y)
```

לא חוקי שתי פעולות עם חתימה זה באותו של המחלקה

```
4. private double paint (String y, double x)
```

זו העמסה חוקית שם הפונקציה זהה הפרמטרים סדר מיקומם שונה

ב:

נתונה הפעולה perimeter במחלקה Square

```
public int perimeter(double scale){ … }
```

נתונה המחלקה היורשת Rectangle

אלו מהפעולות הבאות הן פעולות דורסות:

```
1. private int perimeter (double x)
2. public int perimeter (double y)
3. public void perimeter ()
4. public int perimeter (int y)
5. public void perimeter (double x)
```

תשובות:

1 – פעולה דורסת, ואסור להחמיר בהרשאת גישה. (כלומר בדריסה, בפעולה הדורסת במחלקה היורשת, לא ניתן לצמצם את אפשרות הגישה בהשוואה להרשאה במחלקה המורישה

2- פעולה דורסת, שם ופרמטרים זה

3- הפעולה לא דורסת, הפרמטרים שונים

4- הפעולה לא דורסת הפרמטרים שונים

5-לא חוקי, שמות המשתנים שונים אבל הטיפס double שונה, הפעולות באותה מחלקה

סעיף ב'

.1

אין במחלקה Square פעולת equals ולכן מופעלת הפעולה equals של המחלקה Object שממנה יורשת Square(כמו כל מחלקות של JAVA) ובפעולה הזו יש השוואה של הפניות, ואכן ההפניות לא שוות

זימון פעולת equals עם הפניה של עצם מסוג Square ל- Square אין פעולת equals ולכן מזומנת הפעולה equals של המחלקה Obejct, ופעולה זו משווה בין שתי הפניות, והן לא שוות ולכן מודפס false

```
System.out.println(sqr1.equals(sqr2)); false  -
```

.2

sqrRec זו הפנייה של מסוג Square , אין ל- Square פונקצית equals לכן מזומנת הפעולה של המחלקה Object והשוואה בין שתי ההפניות תוצאתה false

```
System.out.println(sqrRec.equals(rec));false
```

.3

sqrRec זו הפנייה של מסוג Square , אין Square פונקצית equals, על ההפניה rec מופעלת המרה כלפי מעלה להפנייה מסוג Squar לכן מזומנת הפעולה של המחלקה Object והשוואה בין שתי ההפניות תוצאתה false

```
System.out.println(sqrRec.equals((Square)rec));false
```

.4

כאן משווים בין שני מלבנים, אבל חשוב לשים לב שנדרשנו לעשות המרה של הפנייה sqrRec להפנייה של מלבן

```
System.out.println(((Rectangle)sqrRec).equals(rec));true
```

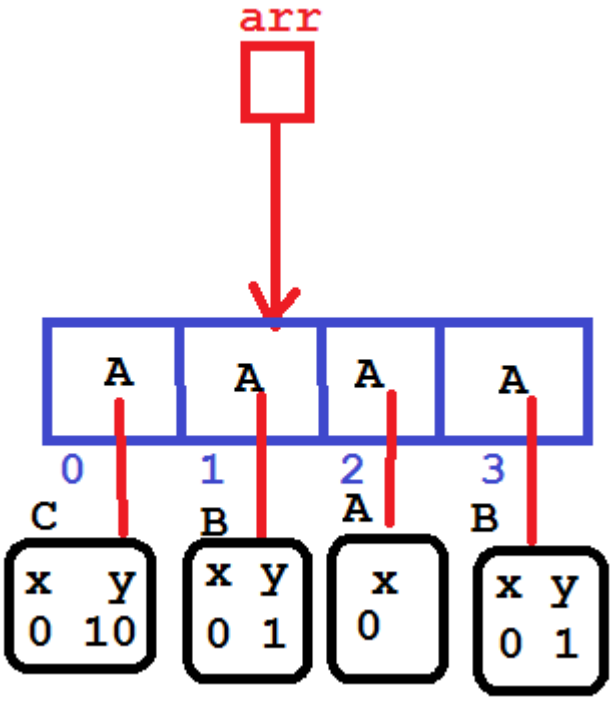
.5 `System.out.println(rec.equals(sqrRec)); false`

שאלה 5

יצירת מערך ועצמים

```
A [] arr = new A[4];
arr [0] = new C();
arr [1] = new B();
arr [2] = new A();
arr [3] = new B();
```

תוצאת יצירת העצמים:



לולאת הסריקה של המערך והפעלת פעולות:

```
for (int k = 0; k < arr.length; k ++)
{
  System.out.println("arr[" + k + "] = " + arr [k]);
  arr [k]. method (k+2);
  System.out.println("arr[" + k + "] = " + arr [k]);
}
```

ההדפסה הראשונה מדפיסה שם המערך והמיקום arr[k], תוצאת פעולת toString של כל עצם לפי המחלקה של העצם כפי שיצרנו, והפעלת method עם הערך של k+2.

לכן הוראת ההדפסה תפיק את הפלט עבור עצם של המחלקה C

arr[0] = 0/10

מופעלת ההוראה: arr[0].method(k+2) ===> כלומר זימון arr[0].method(2)

מתבצע זימון של הפעולה method של המחלקה C

ולכן ערכי x,y מקבלים הוספה של 2, ועתה ערכם בתכונות העצם במקום -0 הוא:

x = 2, y = 12

הוראת הפלט השנייה תדפיס:

arr[0] = 2/12

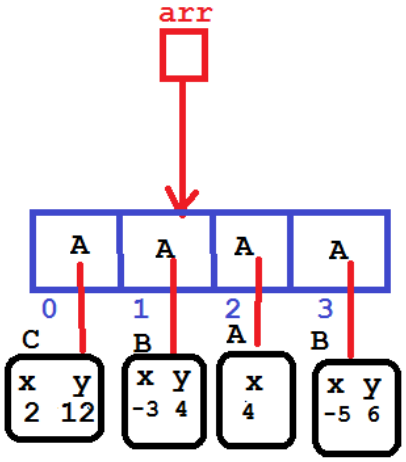
לכל אחת מהמחלקות יש פעולת toString ולכל אחת מהמחלקות יש פעולת method, ולכן הזימון יהיה בכל מחזור בלולאה לפי המחלקה ממנה נוצר העצם, והפלט הסופי:

arr[0] = 0/10
arr[0] = 2/12

arr[1] = 0.1
arr[1] = -3.4

arr[2] = 0
arr[2] = 4
arr[3] = 0.1
arr[3] = -5.6

ערכי העצמים גם משתנים כתוצאה מהפעולה method ונקבל את המערך הבא:



שאלה 6

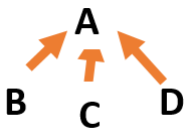
חלק א

```
public static void main(String[] args)
  {
   1. C c = new A();
   2. B b1 = (B)(new A());
   3. B b2 = new D();
   4. A a = new D();
}
```

מספרתי את ההוראות ונבדוק את משמעותם:

סעיף 1: ציור עץ ההורשה

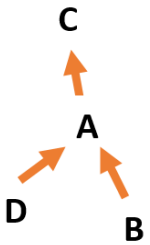
הוראה 1– שגיאה תחבירית C יורשת מ- A



סעיף 2 : ציור עץ ההורשה

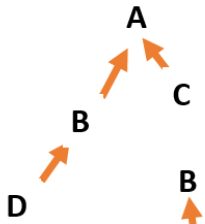
הוראה 1 תקינה

הוראה 2 שגיאת ריצה



סעיף 3 : ציור עץ ההורשה

הוראה 1 - שגיאה תחבירית A לא יורש מ- C



סעיף 4: : ציור עץ ההורשה

הוראה 1 : תקינה

הוראה 2 : תקינה

הוראה 3 : תקינה

הוראה 4 : תקינה

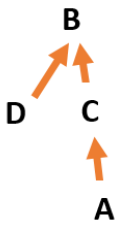
סעיף 5: ציור עץ ההורשה

הוראה 1 : תקינה

הוראה 2 : תקינה

הוראה 3 : תקינה

הוראה 4 : שגיאה תחבירית



חלק ב

כל המחלקות:

```
public class Point
 {
     private int x;
     public Point (int x)
   {
          this.x = x;
      }
     public String toString()
   {
          return " x= " + this.x;
      }
  }
public class Circle extends Point
 {
    private int radius;
    public Circle( int x, int radius)
    {
     super(x);
     this.radius= radius;
    }
    public String toString()
    {
     return super.toString()+ ", radius = "+ radius;
    }
}
public class Cylinder extends Circle
 {
   private int height;
    public Cylinder(int x,int radius,int height){
     super( x, radius);
     this.height = height;
    }
    public String toString(){
      return super.toString() + "height= "+height;
    }
 }
```

```
public class TestGalil{
       public static void main(String[] args)
       {
        Circle b = new Circle (1, 2);
        System.out.println(b);
        Cylinder c = new Cylinder (10, 20, 30);
        System.out.println(c);
         }
      }
```

שאלה 7

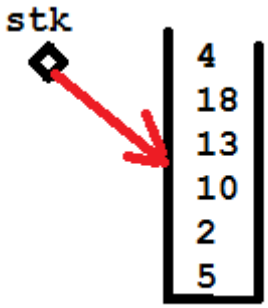
סעיף א.

```
public static Queue<Integer> doIt(Node<Node<Integer>> lst){
Queue<Integer>q = new Queue<Integer>() ;
 while(lst!=null){
   Node<Integer> p = lst.getValue();
    while(p.getNext()!=null){
      p= p.getNext();
   }
  q.insert(p.getValue());
 lst = lst.getNext();
}
return q;
}
```

סעיף ב.

```
public static int doIt(Node<Node<Integer>> lst){
int max=0 ;
while(lst!=null){
    Node<Integer> p = lst.getValue();
   int power = 1;
   double s = 0;
  while(p!=null){
       s=s+p.getValue() *Math.pow(10, power);
      p=p.getNext();
      power = power ++;
  } // end loop p!=null
  if( s> max)
     max = s;
  lst = lst.getNext();
 } // end loop lst!=null
return  max;
 } //end method
```

שאלה 8



לולאה ראשונה:

הרשימה שהמצביע שלה עובר ל- first היא:

list1: 4,18,2,7,3,1,2

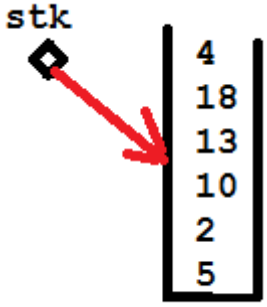
הרשימה שהמצביע שלה עובד ל- second היא:

list2: 5,2,10,13,18,4

לולאה ראשונה מבצעת העברת רשימה second למחסנית לכן נקבל את המחסנית stk

second: : 5,2,10,13,18,4

עוברת למחסנית stk



הלולאה השנייה סורקת את רשימה שהמצביע first מצביע עליה, וסורקת את המחסנית ,

כל עוד מתקיים התנאי שלא הגענו לסוף הרשימה וגם לא למחסנית ריקה הלולאה מתבצעת, ועם תנאי להדפסה

אם האיבר בראש המחסנית = לאיבר שהמצביע first מצביע עליו

list1: 4,18,2,7,3,1,2

הלולאה:

4=4 לכם 4 יודפס

18 = 18 לכן 18 יודפס

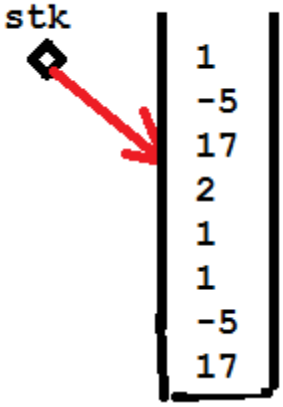
אין איברים שווים באותו הסדר, הלולאה תסתיים כי המחסנית תהיה ריקה (לפני סיום המעבר על הרשימה)

כדי לקבל את הפלט 1,-5,17

אנחנו צרכים שתי רשימות בהם יש את שלושת המספרים הללו בסדר הפוך,

chain1 תיהיה הפרמטר השני(second) , כי יש לה משני הצדדים את הערכים, ולכן במעבר למחסנית , שלושת הערכים הללו יהיו בראש הרשימה

chain1: 17,-5,1,1,2,17,-5,1



במעבר למחסנית נקבל:

ולכן הפרמטר הראשון (first) יהיה

chain3: 1,-5,17,0,2,13,6

שכוללת בתחילת הרשימה רק שלושה ערכים ששווים לפי הסדר ל-3 הערכים הראשונים במחסנית.

שאלה 9

```
public class Classroom
{
   private String buildingName;
   public int roomNum;
   private int seats;
   private int[]group ;
   public Classroom(String buildingName,int roomNum, int seats,
int[]g ){
        this.buildingName=buildingName;
        this.roomNum = roomNum;
        this.seats = seats;
        group = g;
   }
   protected boolean isFree(int day){
        return group[day]==0;
   }
}
public class Lab extends Classroom{
 private String [] tools;
 public Lab(String buildingName,int roomNum, int seats, int[]g,
String[] eq){
      super( buildingName, roomNum,  seats,g);
      tools = eq;
 }
 public boolean isEquipmentExist(String e){
      boolean found = false;
      for(int i=0;i<tools.length;i++)
           if(tools[i].equals(e))
                found = true;
      return found;
 }

}
public class Classrooms
{
   private Classroom[]rooms;
   public void printFreeClassrooms (int day){
        for(int i=0; i<rooms.length;i++ )
             if(!(rooms[i] instanceof Lab))
                 if(rooms[i].isFree(day))
                      System.out.println(rooms[i].getRoomNum());
   }
```

```
   public  Queue<Lab>collectEquipmentExist(String eq){
        Queue<Lab>q= new Queue<Lab>();
        for(int i=0; i<rooms.length;i++ )
             if(rooms[i] instanceof Lab)
     if(((Lab)(rooms[i]).isEquipmentExist(eq))
        q.insert((Lab)rooms[i]);
       )
        return q;
   }
}
```

שאלה 10

א. חלק שמגדיר את המחסנית הכפולה, רק החלק של הכותרת והתכונות נדרש

```
import unit4.collectionsLib.*;
import java.util.Random;
public class DoubleStack
{
 private Stack<Integer>sLeft = new Stack<Integer>();
 private Stack<Integer>sRight = new Stack<Integer>();

 public DoubleStack(int n,int x,int y){
      Random rnd = new Random();
      for(int i=1;i<=n;i++){
           int s = rnd.nextInt(2);
           int num = rnd.nextInt(x)+(y-x);
           if(s==0)
                sLeft.push(num);
           else
                sRight.push(num);
      }
 }
}
```

ב. מיון המחסניות תוך שימוש בממשק הנתון

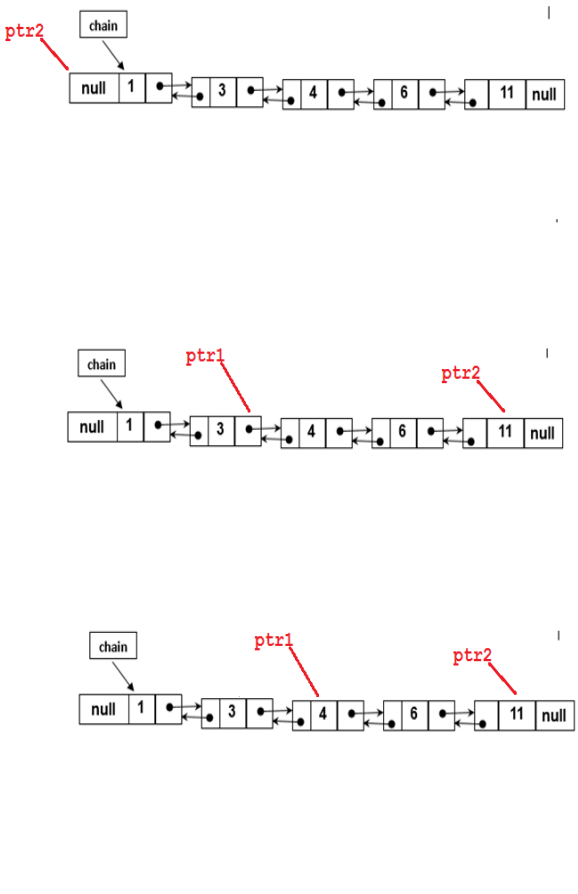
```
  public class TestD
  {
     public static void sort (DoubleStack du){
          int nLeft = numElements(1);
          int nRight = numElements(2);
          for(int i=1;i<=nLeft;i++)
               du.moveMin(2);
          for(int i=1;i<=nLeft;i++){
               du.move(1);
          for(int i=1;i<=nRight;i++)
               du.moveMin(1);
          for(int i=1;i<=nRight;i++){
               du.move(2);
          }
     }
     public static void main(String[] args)
     {
          DoubleStack du= new DoubleStack(20,1,10);
          sort(du);
     }
```

שאלה 11

תיאור לפי המעקב מה מתבצע עבור הזימון what(chain,7)

הרשימה ממויינת בסדר עולה, שני המצביעים מצביעים על קצוות הרשימה. מחושב ממוצע הרשימה, ומתבצעת השוואה בין הממוצע שחושב לערך num שהתקבל, אם יש שיוון, מודפסים

ערכי שתי החוליות בצד שמאל, בצד ימין שסכום ממוצע הערכים בין שני מצביעים אלו .
ההשוואה בין num ל-x
אם יש שויון מוחזר ערך אמת,



temp	c	ptr1.getValue()	ptr2.getValue()	x	num	הוראה	
0	0				7	temp=0, c=0	
						ptr2 = chain	
						while(ptr2.get...)	לולאה חוזרת עד לחוליה האחרונה, חישוב סכום הרשימה ומספר איברים
						temp=ptr2.getValue()	
						ptr2= ptr2.getRight()	
14	4					c++	
25	5					temp+=ptr2.getValue()	חיבור הערך האחרון , עדכון מספר האיברים, חישוב ממוצע הרשימה
						c++	
				5		x= temp/c	
		1	5			ptr1 = chain	
						while(ptr1.getLeft()!ptr2)	לולאה עד לקיום התנא
						if(x == num)	האם הממוצע = לערך שהתקבל 5
						if(x<num)	התנאי מתקיים 5<7
24						temp-=ptr1.getValue()	
						ptr1=ptr1.getRight()	
		3	11			c--;	
				6		x= temp/c	
						while(ptr1.getLeft()!ptr2)	
						if(x == num)	
						if(x<num)	
21						temp-=ptr1.getValue()	
	3	4	11			ptr1=ptr1.getRight()	
						c--	
				7		x=temp/3	
						while(ptr1.getLeft()!ptr2)	
						if(x=num)	תנאי אמת 7 =7
						s.o.p(...)	מודפסים הערכים של שתי חוליות הקצוות
						return true	

אם הערך של הממוצע של תת הרשימה בין שני המצביעים < מ-num , המצביע בצד שמאל ptr1 "ינוע" ימינה,

אם הערך של הממוצע של תת הרשימה בין שני המצביעים >מ-num , המצביע בצד ימין ptr2 ינוע "ימינה "

אם שני המצביעים
כאשר ptr2 – ptr1.getLeft(), המשמעות שאין תת רשימה שהממוצע שלה = num ומחוזר ערך false
לכן
עבור הזימון what(chain,4.5) יוחזר ערך false
עבור הזימון what(chain,4) יוחזר ערך false