מדעי המחשב

פתרון בחינת הבגרות

פרק ראשון - (יסודות)

שאלה 1 **Java**

```
int num = input.nextInt();

int sum = 0;
for (int i = 0 ; i < arr.length ; i++)
{
    if (arr[i] < num)
        sum = sum + arr[i];
}
System.out.println("sum = " + sum);
```

C#

```
int num = int.Parse(Console.ReadLine());

int sum = 0;
for (int i = 0; i < arr.Length; i++)
{
    if (arr[i] < num)
        sum = sum + arr[i];
}
Console.WriteLine("sum = " + sum);
```

שאלה 2

Java

```
public class A
{
    //--- תכונות המחלקה .i ---
    private int n1, n2;

    //--- בנאים .ii  ---
    public A (int n1, int n2)
    public A () {}

    //--- get / set פעולות .iii ---
    public int getN1()  {   return this.n1; }
    public int getN2()  {   return this.n2; }
    public void setN1 (int n1) { this.n1 = n1; }
    public void setN2 (int n2) { this.n2 = n2; }

    //--- הוספת מספר לשתי התכונות .iv ---
    public void add(int n)
    {
        this.n1 += n;
        this.n2 += n;
    }

    //--- פעולה המחזירה מחרוזת המציגה את העצם כתרגיל חשבוני .v ---
    public String toString()
    {
        return this.n1 + " + " + this.n2 + " = " + (this.n1 + this.n2);
    }
}
```

ב.

a		a1		פלט
A		A		4 + 5 = 9 4 + 5 = 9 8 + 9 = 17
n1	n2	n1	n2	
? 4	? 5	4 8	~~5~~ 9	

שאלה 2 **C#:**

```
class A
{
    //--- תכונות המחלקה .i ---
    private int n1, n2;

    //--- בנאים .ii  ---
    public A(int n1, int n2)
    {
        this.n1 = n1;
        this.n2 = n2;
    }
    public A() { }

    //--- Get / Set פעולות .iii ---
    public int GetN1() { return this.n1; }
    public int GetN2() { return this.n2; }
    public void SetN1(int n1) { this.n1 = n1; }
    public void SetN2(int n2) { this.n2 = n2; }

    //--- הוספת מספר לשתי התכונות .iv ---
    public void Add(int n)
    {
        this.n1 += n;
        this.n2 += n;
    }

    //--- פעולה המחזירה מחרוזת המציגה את העצם כתרגיל חשבוני .v ---
    public override String ToString()
    {
        return this.n1 + " + " + this.n2 + " = " + (this.n1 + this.n2);
    }
}
```

ב.

a		a1		פלט
A		A		4 + 5 = 9 4 + 5 = 9 8 + 9 = 17
n1	n2	n1	n2	
? 4	? 5	~~4~~ 8	~~5~~ 9	

שאלה 3 **: Java**

```
public class Button
{
    private int num;          //  מספר סידורי
    private int size;         //  גודל הכפתור
    private String color;     //  צבע הכפתור

    //--- א. בונים ---
    public Button(int num)
    {
        this.num = num;
        this.size = 5;
        this.color = "black";
    }

    public Button(int num, int size, String color)

    //---  ב. פעולה המגדילה את גודל הכפתור ---
    public void addToSize (int x){  this.size += x; }

    //--- ג. פעולה המחזירה אמת אם גודל הכפתור הנוכחי ---
    //---        זהה לגודל הכפתור האחר, ושקר אחרת     ---
    public boolean isSameSize (Button other)
    {
        return this.size == other.size;
    }
}
```

לא נדרש

ד.

a1

Button	
num	1
size	~~12~~ 14
color	red

a2

Button	
num	2
size	12
color	green

a3

Button	
num	3
size	12
color	blue

a4

Button	
num	4
size	~~5~~ 14
color	black

תנאי	ערכים	תוצאה	פלט
a1.isSameSize(a3)	$14 \neq 12$	false	
a4.isSameSize(a1)	14 = 14	true	$$$
a2.isSameSize(a3)	12 = 12	true	###

שאלה 3 **C#:**

```
class Button
{
    private int num;        //  מספר סידורי
    private int size;       //  גודל הכפתור
    private String color;   //  צבע הכפתור

    //--- א. בנאים ---
    public Button(int num)
    {
        this.num = num;
        this.size = 5;
        this.color = "black";
    }

    public Button(int num, int size, String color)...

    //---  ב. פעולה המגדילה את גודל הכפתור ---
    public void AddToSize(int x) { this.size += x; }

    //--- ג. פעולה המחזירה אמת אם גודל הכפתור הנוכחי ---
    //---        זהה לגודל הכפתור האחר, ושקר אחרת    ---
    public bool IsSameSize(Button other)
    {
        return this.size == other.size;
    }
}
```

לא נדרש

ד.

a1

Button	
num	1
size	~~12~~ 14
color	red

a2

Button	
num	2
size	12
color	green

a3

Button	
num	3
size	12
color	blue

a4

Button	
num	4
size	~~5~~ 14
color	black

תנאי	ערכים	תוצאה	פלט
a1.IsSameSize(a3)	$14 \neq 12$	false	
a4.IsSameSize(a1)	14 = 14	true	$$$
a2.IsSameSize(a3)	12 = 12	true	###

פרק שני - (מבני נתונים)

שאלה 4 : **Java**

פעולות עזר:

```
//--- פעולה המחזירה את מספר האיברים ברשימה ---
public static int size (Node <Integer> lst)
{
    int count = 0;
    while (lst != null)
    {
        count ++;
        lst = lst.getNext();
    }
    return count;
}

//---                 פעולה המקבלת רשימה ומספר n ---
//--- ומחזירה הפנייה לאיבר שנמצא במקום n ברשימה ---
//---       null אם לא קיים איבר במקום זה יוחזר ---
public static Node <Integer> place (Node<Integer> lst, int n)
{
    while (lst != null  && n > 0)
    {
        n -- ;
        lst = lst.getNext();
    }
    return lst;
}
```

```
//--- פעולה המחזירה "אמת" אם הרשימה היא "רשימה משולשת" ו- "שקר" אחרת ---
//---          רשימה משולשת היא רשימה לא ריקה שמספר איבריה מתחלק ב- 3 ---
//---                והאיברים בכל שלוש ברשימה זהים ומופיעים באותו סדר ---

//--- פתרון באמצעות 3 הפניות, אחת על כל שלוש ברשימה ---
public static boolean isTriple (Node <Integer> L)
{
    int listSize = size(L);
    int n = listSize / 3;

    //--- אם הרשימה אינה עומדת בקריטריונים ---
    if (listSize == 0 || listSize % 3 != 0)
        return false;

    Node <Integer> pos1 = L;                //    הפנייה לתחילת הרשימה
    Node <Integer> pos2 = place (L, n);     //  הפנייה לתחילת השליש השני
    Node <Integer> pos3 = place (L, 2*n);   // הפנייה לתחילת השליש השלישי

    while (pos3 != null)
    {
        if(pos1.getValue()!= pos2.getValue() || pos1.getValue()!= pos3.getValue())
            return false;
        pos1 = pos1.getNext();
        pos2 = pos2.getNext();
        pos3 = pos3.getNext();
    }
    return true;
}
```

```
//--- פעולה המחזירה "אמת" אם הרשימה היא "רשימה משולשת" ו- "שקר" אחרת ---
//---          רשימה משולשת היא רשימה לא ריקה שמספר איבריה מתחלק ב- 3 ---
//---                והאיברים בכל שלוש ברשימה זהים ומופיעים באותו סדר ---

//--- פתרון באמצעות שתי הפניות, האחת לתחילת הרשימה ---
//---                    והשנייה לתחילת השליש השני ---
public static boolean isTriple (Node <Integer> L)
{
    int listSize = size(L);
    int n = listSize / 3;

    //--- אם הרשימה אינה עומדת בקריטריונים ---
    if (listSize == 0 || listSize % 3 != 0)
        return false;

    Node <Integer> pos1 = L;                // הפנייה לתחילת הרשימה
    Node <Integer> pos2 = place (L, n);     //  הפנייה לתחילת השליש השני

    while (pos2 != null)
    {
        if(pos1.getValue()!= pos2.getValue())
            return false;
        pos1 = pos1.getNext();
        pos2 = pos2.getNext();
    }
    return true;
}
```

שאלה 4 **C#:**

פעולות עזר:

```
//--- פעולה המחזירה את מספר האיברים ברשימה ---
public static int size (Node <Integer> lst)
{
    int count = 0;
    while (lst != null)
    {
        count ++;
        lst = lst.getNext();
    }
    return count;
}

//---                    פעולה המקסלת רשימה ומספר n ---
//---  ומחזירה הפניה לחוליה שנצאת במקום ה-n ברשימה ---
//---          null אם לא קיים איבר במקום זה, יוחזר ---
public static Node <Integer> place (Node<Integer> lst, int n)
{
    while (lst != null  && n > 0)
    {
        n -- ;
        lst = lst.getNext();
    }
    return lst;
}
```

```
//--- פעולה המחזירה "אמת" אם הרשימה היא "רשימה משולשת" ו- "שקר" אחרת ---
//---         רשימה משולשת היא רשימה לא ריקה שמספר איבריה מתחלק ב- 3 ---
//---               והאיברים בכל שליש ברשימה זהים ומופיעים באותו סדר ---

//--- פתרון באמצעות 3 הפניות, אחת על כל שליש ברשימה ---
public static bool IsTriple(Node<int> L)
{
    int listSize = Size(L);
    int n = listSize / 3;

    //--- אם הרשימה אינה עומדת בקריטריונים ---
    if (listSize == 0 || listSize % 3 != 0)
        return false;

    Node<int> pos1 = L;                 // הפנייה לתחילת הרשימה
    Node<int> pos2 = Place(L, n);       //  הפנייה לתחילת השליש השני
    Node<int> pos3 = Place(L, 2*n);     // הפנייה לתחילת השליש השלישי

    while (pos2 != null)
    {
        if (pos1.GetValue() != pos2.GetValue() || pos1.GetValue() != pos3.GetValue())
            return false;
        pos1 = pos1.GetNext();
        pos2 = pos2.GetNext();
        pos3 = pos3.GetNext();
    }
    return true;
}

//--- פעולה המחזירה "אמת" אם הרשימה היא "רשימה משולשת" ו- "שקר" אחרת ---
//---         רשימה משולשת היא רשימה לא ריקה שמספר איבריה מתחלק ב- 3 ---
//---               והאיברים בכל שליש ברשימה זהים ומופיעים באותו סדר ---

//--- פתרון באמצעות שתי הפניות, האחת לתחילת הרשימה ---
//---                   והשנייה לתחילת השליש השני ---
public static bool IsTriple(Node<int> L)
{
    int listSize = Size(L);
    int n = listSize / 3;

    //--- אם הרשימה אינה עומדת בקריטריונים ---
    if (listSize == 0 || listSize % 3 != 0)
        return false;

    Node<int> pos1 = L;                 // הפנייה לתחילת הרשימה
    Node<int> pos2 = Place(L, n);       //  הפנייה לתחילת השליש השני

    while (pos2 != null)
    {
        if (pos1.GetValue() != pos2.GetValue())
            return false;
        pos1 = pos1.GetNext();
        pos2 = pos2.GetNext();
    }
    return true;
}
```

a.length = 5
a.Length = 5

	0	1	2	3	4
a	2	4	7	12	18

שאלה 5

א. sod (a,11)

k	i	i<4	j	j<5	a[i]	a[j]	a[i]+a[j] == k	ערך מוחזר
11	0	T	1	T	2	4	F	
			2	T		7	F	
			3	T		12	F	
			4	T		18	F	
			5	F				
	1	T	2	T	4	7	T	אמת

ב. sod (a,10)

k	i	i<4	j	j<5	a[i]	a[j]	a[i]+a[j] == k	ערך מוחזר
10	0	T	1	T	2	4	F	
			2	T		7	F	
			3	T		12	F	
			4	T		18	F	
			5	F				
	1	T	2	T	4	7	F	
			3	T		12	F	
			4	T		18	F	
			5	F				
	2	T	3	T	7	12	F	
			4	T		18	F	
			5	F				
	3	T	4	T	12	18	F	
			5	F				
	4	F						שקר

ג. הפעולה מחזירה "אמת" אם קיימים שני איברים במערך שסכומם k, ו- "שקר" אחרת.

ד. סיבוכיות הפעולה sod היא $O(n^2)$.

בהנחה שיש במערך n איברים:
הלולאה החיצונית רצה על כל המערך - סה"כ n צעדים
ובתוכה לולאה פנימית הרצה בכל פעם על איבר אחד פחות:

$$(n-1)+(n-2)+(n-3)+...+2+1 \Rightarrow S_n = \frac{(1+n-1)(n-1)}{2} \Rightarrow \frac{n^2-n}{2} \Rightarrow O(n^2)$$

ה. what (a, k)

a.length = 5
a.Length = 5

a

0	1	2	3	4
2	4	7	12	18

k	left	right	left<right	I a[left]	II a[right]	I+II == k	I+II < k	ערך מוחזר
11	0	4	T	2	18	F	F	
		3	T		12	F	F	
		2	T		7	F	T	
	1		T	4		T		אמת

ו. סיבוכיות הפעולה what היא O(n).

הפעולה עוברת לכל היותר מעבר אחד על כל נתוני המערך, עד שמוצאת או לא מוצאת שני איברים שסכומם k.

ז. סיבוכיות הפעולה sod - ריבועית $O(n^2)$
סיבוכיות הפעולה what - ליניארית O(n)
ולכן, what יעילה יותר.

ח. (1) sod תשיג את מטרתה, כי היא בודקת בכל פעם איבר אחר מול כולם.
(2) what לא תשיג את המטרתה, כי היא אינה בודקת את כל האפשרויות.

למשל: עבור המערך הבא תחזיר sod "אמת", ואילו what תחזיר "שקר".

a

0	1	2	3	4
18	2	4	7	12

שאלה 6

Java :

א - ג. כותרות הפעולות משולב עם סעיפים ב' (מימוש פעולה בונה) ו- ג' (מימוש הפעולה sum)

```
//--- מחלקה המייצגת את 5 ערימות הקלפים ---
public class Deck
{
    private Stack<Card> [] heaps;    //  ערימות הקלפים
    private int sum;                 //  סכום הקלפים שבערימה החמישית (ערימה 0)

    //---    ב. פעולה בונה ---
    public Deck()
    {
        this.heaps = new Stack[5];

        for (int i = 0 ; i < heaps.length ; i++)
            this.heaps[i] = new Stack<Card>();

        this.sum = 0;
    }

    //--- פעולה המקבלת קלף ומכניסה אותו לראש הערימה הנכונה ---
    //--- ערימה נכונה היא ערימה 1 - 4, לפי צורת הקלף הנתון ---
    public void insert (Card c)

    public static Random rnd = new Random();

    //--- פעולה המגרילה מספר ערימה, ומחזירה 'אמת' אם הצליחה להעביר ---
    //---     קלף מראש הערימה שהוגרלה לערימה החמישית, ו-'שקר' אחרת ---
    public boolean move ()

    //--- ג. פעולה המחזירה את סכום הקלפים שבערימה החמישית  ---
    public int sum() {  return this.sum; }
```

ד.

```
//--- פעולה המחזירה 'אמת' אם המשחק הסתיים בניצחון ---
//---   על פי הכללים שמוגדרים בשאלה, ו-'שקר' אחרת ---
public static boolean game (Card [] card)
{

    Deck deck = new Deck ();

    //--- שלב ראשון של המשחק - מילוי ערימות הקלפים ---
    for (int i = 0 ; i < card.length ; i++)
        deck.insert(card[i]);

    //--- שלב שני של המשחק - העברת הקלפים לערימה החמישית ---
    boolean gameOver = false;
    while (! gameOver)
    {
        if (! deck.move())
            gameOver = ! deck.move();
    }

    //--- האם יש ניצחון ? ---
    if (deck.sum() % 100 == 0)
        return true;
    return false;
}
```

דרך כתיבה נוספת לניהול השלב השני של המשחק:

```
boolean gameOver = false;
while (! gameOver)
{
      if (! deck.move())
            gameOver = true;

}
```

שאלה 6

C# :

א - ג. כותרות הפעולות משולב עם סעיפים ב' (מימוש פעולה בונה) ו- ג' (מימוש הפעולה sum)

```
class Deck
{
    private Stack<Card>[] heaps;     //  ערימות הקלפים
    private int sum;                 //  סכום הקלפים שבערימה החמישית (ערימה 0)

    //---    ב. פעולה בונה ---
    public Deck()
    {
        this.heaps = new Stack<Card>[5];

        for (int i = 0; i < heaps.Length; i++)
            this.heaps[i] = new Stack<Card>();

        this.sum = 0;
    }

    //--- פעולה המקבלת קלף ומכניסה אותו לראש הערימה הנכונה ---
    //--- ערימה נכונה היא ערימה 1 - 4, לפי צורת הקלף הנתון ---
    public void Insert(Card c)...

    public static Random rnd = new Random();

    //--- פעולה המגרילה מספר ערימה, ומחזירה 'אמת' אם הצליחה להעביר ---
    //---     קלף מראש הערימה שהוגרלה לערימה החמישית, ו-'שקר' אחרת ---
    public bool Move()...

    //--- ג. פעולה המחזירה את סכום הקלפים שבערימה החמישית  ---
    public int Sum() { return this.sum; }
```

ד.

```
//--- פעולה המחזירה 'אמת' אם המשחק הסתיים בניצחון ---
//---   על פי הכללים שמוגדרים בשאלה, ו-'שקר' אחרת ---
public static bool Game(Card[] card)
{

    Deck deck = new Deck();

    //--- שלב ראשון של המשחק - מילוי ערימות הקלפים ---
    for (int i = 0; i < card.Length; i++)
        deck.Insert(card[i]);

    Console.WriteLine("\n--- the game ---");
    Console.WriteLine(deck);

    //--- שלב שני של המשחק - העברת הקלפים לערימה החמישית ---
    bool gameOver = false;
    while (!gameOver)
        gameOver = !deck.Move();

    //--- האם יש ניצחון ? ---
    if (deck.Sum() % 100 == 0)
        return true;
    return false;
}
```

דרך כתיבה נוספת לניהול השלב השני של המשחק:

```
bool gameOver = false;
while (! gameOver)
{
      if (! deck.Move())
            gameOver = true;

}
```

פרק שלישי - מערכות מחשב ואסמבלר
הפתרון לפרק זה נכתב ע"י: **רונית מרציאנו**

שאלה 7

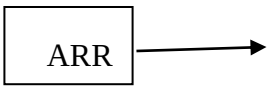
א. טבלת מעקב אחר ביצוע קטע התוכנית

AX		BX		ZF	SF	CF	
AH	AL	BH	BL				
C8H	3BH						MOV AX , C83BH
		A8H	9CH		+	+	MOV BX , A89CH
90H	76H						SHL AX , 1
	77H						OR AL , 33H
			63H				NOT BL
38H	**DAH**	**A8H**	**63H**			+	ADD AX , BX

ב. נתונה ההגדרה הבאה: ARR DB 5 DUP(?)

עקוב בעזרת טבלת מעקב אחר ביצוע של כל אחד מהקטעים וקבע אם הוא מבצע את הנדרש או לא.

i. **לא מבצע** את הנדרש, מאפס רק 4 תאים ראשונים



?	00	00	00	00
04H	03H	02H	01H	00H

```
        MOV SI,0
        MOV CX,4
A1:     MOV ARR[SI],0
        INC SI
        LOOP A1
```

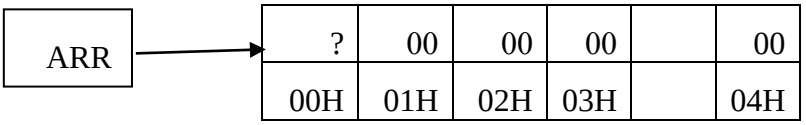
CX		SI
00H	04H	0000H
		0001H
		0002H
		0003H
		0004H

ii. **מבצע** את הנדרש,

```
        MOV   CX,5
        LEA   BX,ARR
        MOV   AL,0
A1:     MOV   [BX],AL
        INC    BX
        LOOP  A1
```

CX		BX	AX		
			AH	AL	
00H	05H	0000H		00	
00H	04H	0001H			
00H	03H	0002H			
00H	02H	0003H			
00H	01H	0004H			
00H	00H	0005H			

iii **לא מבצע** את הנדרש, מאפס רק 4 תאים אחרונים, לא מאפס את התא הראשון




```
        MOV  BX,5
        DEC  BX
A1:     MOV   ARR[BX],0
        DEC   BX
        JNZ   A1
```

BX	
00H	05H
00H	04H
00H	03H
00H	02H
00H	01H
00H	00H

iv מבצע את הנדרש,

```
        MOV DI,0
A1:     MOV ARR[DI],0
        INC DI
        CMP DI,5
        JC  A1
```

DI
0000H
0001H
0002H
0003H
0004H
0005H

שאלה 8

יש לכתוב קטע תוכנית באסמבלי , שיציב באוגר BL את מספר הפעמים שהרצף 1011 מופיע במספר הבינארי שבאוגר AX.

```
        MOV     CX,12 ; LOOP
        MOV     BL,0 ; COUNTER 1011
AGAIN:  MOV     DX,AX
        AND     DX,000BH
        CMP     DX,000BH
        JNZ     CONT
        INC     BL
CONT:   SHR     AX,1
        LOOP    AGAIN
```

פרק שלישי - מבוא לחקר ביצועים

שאלה 9

שאלה 10

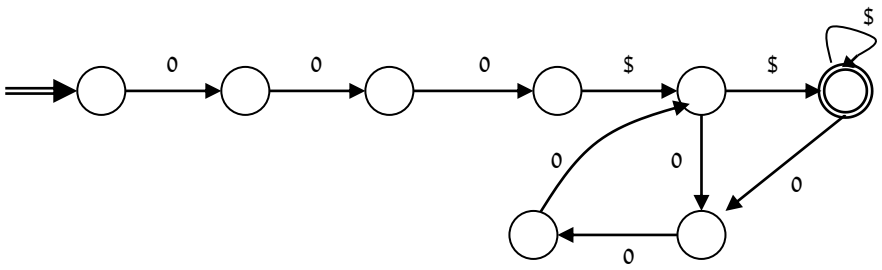
פרק שלישי - מודלים חישוביים

הפתרון לפרק זה נכתב ע"י **רחל לודמר**.

שאלה 11

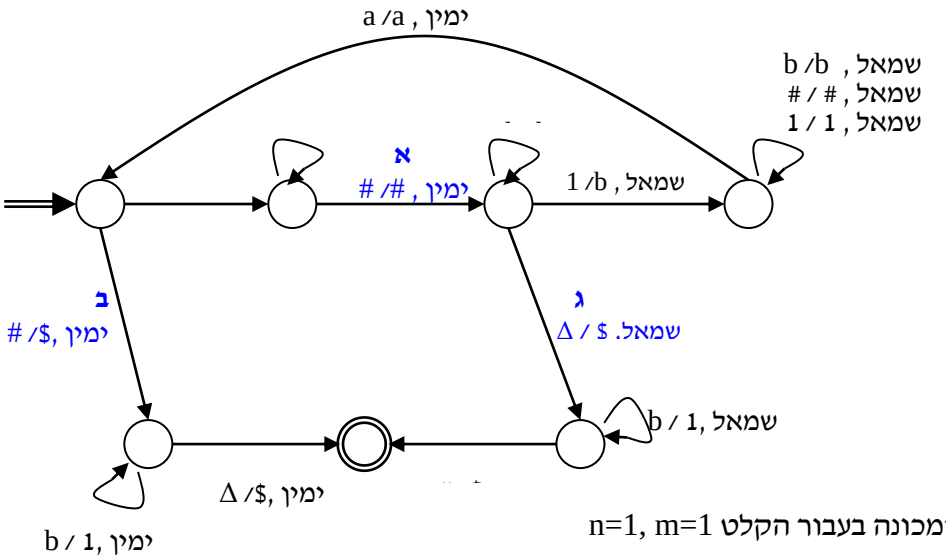
א. המילה הקצרה ביותר היא $$000.

ב. האוטומט שמקבל את השפה L:



שאלה 12

א. השלמת מכונת טיורינג עבור הפונקציה $f(m,n)=\min(m,n)$ כאשר $m,n>0$, כתובים בשפה האונרית וסימן # מפריד בניהם. תוצאת המכונה תיכתב בין שני $.



ב. תהליך החישוב של המכונה בעבור הקלט n=1, m=1

⊢	1	#	1	Δ	

q_0

⊢	a	#	1	Δ	

q_1

⊢	a	#	1	Δ	

q_2

⊢	a	#	b	Δ	

q_3

⊢	a	#	b	Δ	

q_3

⊢	a	#	b	Δ	

q_0

⊢	a	$	b	Δ	

q_4

⊢	a	$	1	Δ	Δ

q_4

⊢	a	$	1	$	Δ

q_5

פרק שלישי - תכנות מונחה עצמים Java

תרגיל 13

א.

```
public class Program
{
    public static void main(String[] args)
    {
        TransportationCompany company1 = new TransportationCompany();

        Vehicle boat = new Boat ("sea ", 50);
        company1.addVehicle (boat);

        company1.addVehicle (new Train (150, 6));

        company1.display();

    }
}
```

אפשר כך (הוספת סירה)

ואפשר גם כך (הוספת רכבת)

ב. נוסיף בכל מחלקה פעולה toString() מתאימה:

במחלקה Vehicle:

```
public String toString()
{
      return this.type + "\t" + this.way +
            "\t max speed : " + this.maxSpeed;
}
```

במחלקה Train:

```
public String toString()
{
       return "Train: \t\t" + super.toString() +
              "\t num of carriages: " + this.numOfCarriages;

}
```

במחלקה Airplane:

```
public String toString()
{
      return "Airplane: \t" + super.toString() +
            "\t max height: " + this.maxHeight;

}
```

במחלקה Boat: (למעשה אפשר לוותר על הפעולה במחלקה זו כי אין לה תכונות נוספות מעבר לאלו שב- Vehicle. הפעולה הוספה למחלקה כדי שגם כלי השיט יציג את שם המחלקה שלו)

```
public String toString()
{
      return "Boat: \t\t" + super.toString();
}
```

ג. הוספת קרונות רכבת הינה באחריותה של הרכבת.
הוספת קרונות לכל הרכבות שבצי כלי הרכב של החברה הינו באחריותה של החברה. רק במחלקה זו יש גישה ישירה למערך כלי הרכב,
ולכן נוסיף את הפעולה במחלקה TransportationCompany:

```
//--- הוספת n שבחברה הרכבות לכל קרונות ---
public void addCarriages (int n)
{
	for (int i = 0 ; i < this.counter ; i++)
	{
		if (this.vehicles[i] instanceof Train)
			((Train)this.vehicles[i]).incNumOfCariagges(n);
	}
}
```

תרגיל 14

```
//---  א ---
public boolean isLike(Object obj)
{
    if (obj instanceof AA  &&  ((AA)obj).st.equals(this.st))
        return true;
    return false;
}

//---  ב ---
public boolean isLike(Object obj)
{
    if (obj instanceof BB  &&  ((BB)obj).num == this.num)
        return true;
    return false;
}
```

```
//-------- ג ---------
AA a = new AA("excellent");
BB b = new BB();

a = b;
if (a.isLike(b))
    System.out.println(a);


/*      טקע התכנית תקין.  פלט
 *  st = excellent  num = 1
 */
```

```
//-------- ד -----
AA aa = new AA("excellent");
BB bb = new BB(2, "excellent");

// bb = aa;
if (bb.isLike(aa))
    System.out.println(bb);

/*
 *                  קטע התכנית שגוי
 *      (מחלקה העל שלו) AA  אינו יכול להפנות לעצם מסוג (מסוג תת מחלקה) bb
 */
```

```
//----- ה -----
//---- פעולה המשרשרת את המחרוזת ---
public static String longString (Object [] arr)
{
    String str = "";
    for (int i = 0 ; i < arr.length ; i++)
    {
        if (arr[i] != null)
        {
            if (arr[i] instanceof BB)
            {
                int num = ((BB)arr[i]).getNum();
                String st = ((BB)arr[i]).getSt();
                for (int j = 0 ; j < num ; j++)
                    str += st + " ";
            }
            else
                if (arr[i] instanceof AA)
                    str += ((AA)arr[i]).getSt();
        }
    }
    return str;
}
```

הבדיקה לא חובה. אם ערך התא null הפעולה instanceof תחזיר שקר

פרק שלישי - תכנות מונחה עצמים C#

שאלה 15

שאלה 16

נכתב ע"י **זיוה קונצמן**

א. במחלקה AA:

```
public virtual bool IsLike(Object obj)
{
        return obj  is AA   &&  ((AA)obj).GetSt().Equals(this.GetSt());
}
```

ב. במחלקה BB:

```
public override bool IsLike(object obj)
{
        return obj  is BB   &&   ((BB)obj).GetNum() == this.GetNum();
}
```

ג. קטע התוכנית נכון. מתבצעת המרה אוטומטית כלפי מעלה של משתנה מטיפוס הבן להפניה מטיפוס האב. הבן הוא גם טיפוס האב לכן אין בעיה.

הפלט יהיה:

```
st = excellent   num = 1
```

ד. קטע התוכנית שגוי. לא יכול להמיר כלפי מטה הפניה מסוג האב להיות הפניה מסוג הבן, הוא לא נוצר כ-BB אלא כ- AA, ואינו יכול לעבור המרה למשהו שהוא לא. השגיאה היא שגיאת קומפילציה.

ה.

```
public static string LongString(Object[] a)
{
        string st = "";
        for (int i = 0; i < st.Length; i++)
        {
                if  (a[i] is AA   &&   ! (a[i] is BB))
                        st += ((AA)a[i]).GetSt();
                else
                        if (a[i] is BB)
                        {
                                for (int j = 1 ;  j < ((BB)a[i]).GetNum() ;  j++)
                                        st += ((BB)a[i]).GetSt();
                        }
        }
        return st;
}
```