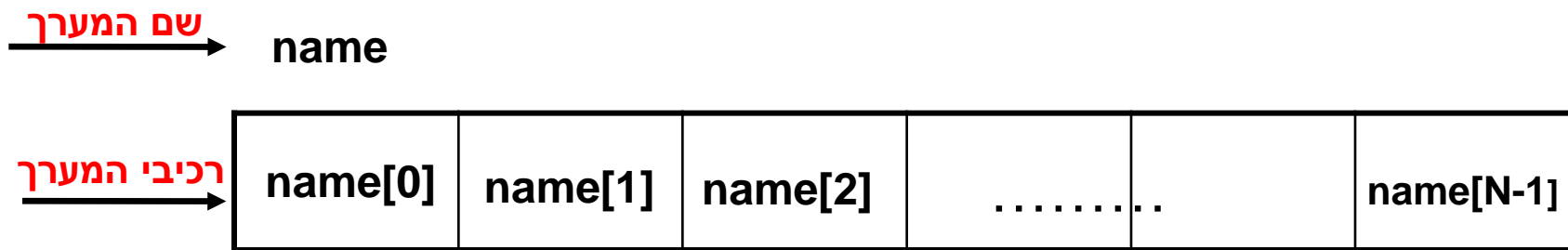


Two Dimensional Arrays

Array

תזכורת

מבנה נתונים פשוט
רצף זהה של משתנים מאותו סוג
אוסף פריטים שניתן לגשת אליהם בצורה ישירה באמצעות אינדקס
סיבוכיות גישה $O(1)$
דרישת הזיכרון היא בדיוק הזיכרון הדרוש לנתונים עצמם
חסרונות: מבנה סטטי




Array

מערך הוא אובייקט לכל דבר (יש לו ייצוג במחסנית וגם ב – heap)
כאשר אנו מגדירים מערך אנחנו מבצעים את השלבים הבאים :


Declaration
Creation
Assignment

```
Int[ ] Arr;  
Arr=new int[3];  
Arr[0]=7; Arr[1]=2; Arr[2]=12;
```



או

```
Int[ ] Arr=new int[3];  
Arr[0]=7; Arr[1]=2; Arr[2]=12;
```



או

```
Int[ ] Arr={7,2,12};
```

Declaration + Creation + Initialization

```

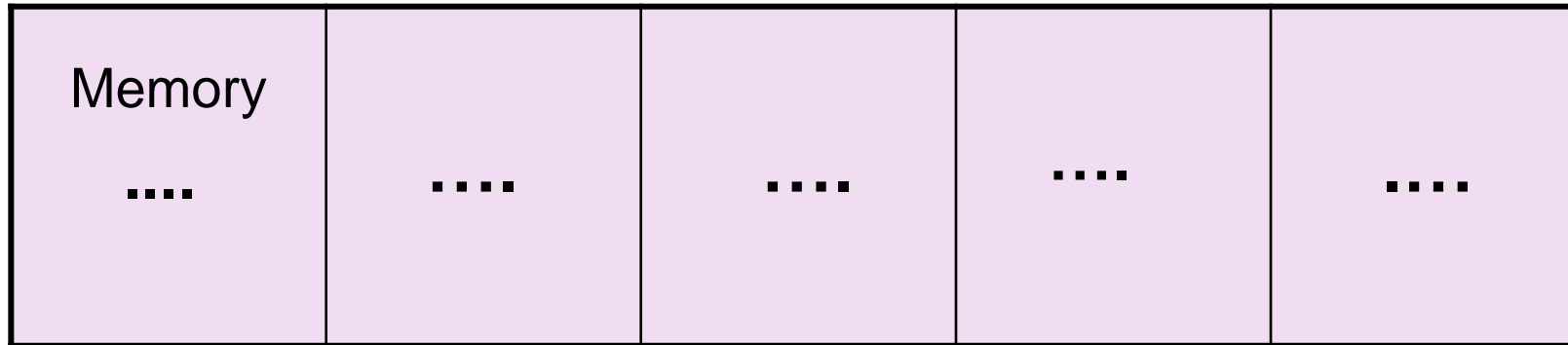
public class Tester{
    public static void main(String[] args){
        int [ ] arr;
        System.out.println (arr); //Error
        Arr = new int [3];
        System.out.println(arr); //[I@1661eeb
        System.out.println(arr[0] + " " + arr[1] + " " + " " + arr[2]); //0 0 0
        arr[0]=7;
        arr[1]=2;
        arr[2]=12;
        System.out.println(arr[0] + " " + arr[1] + " " + " " + arr[2]); //7 2 12
    }
}

```

שלבי היצירה של המערך

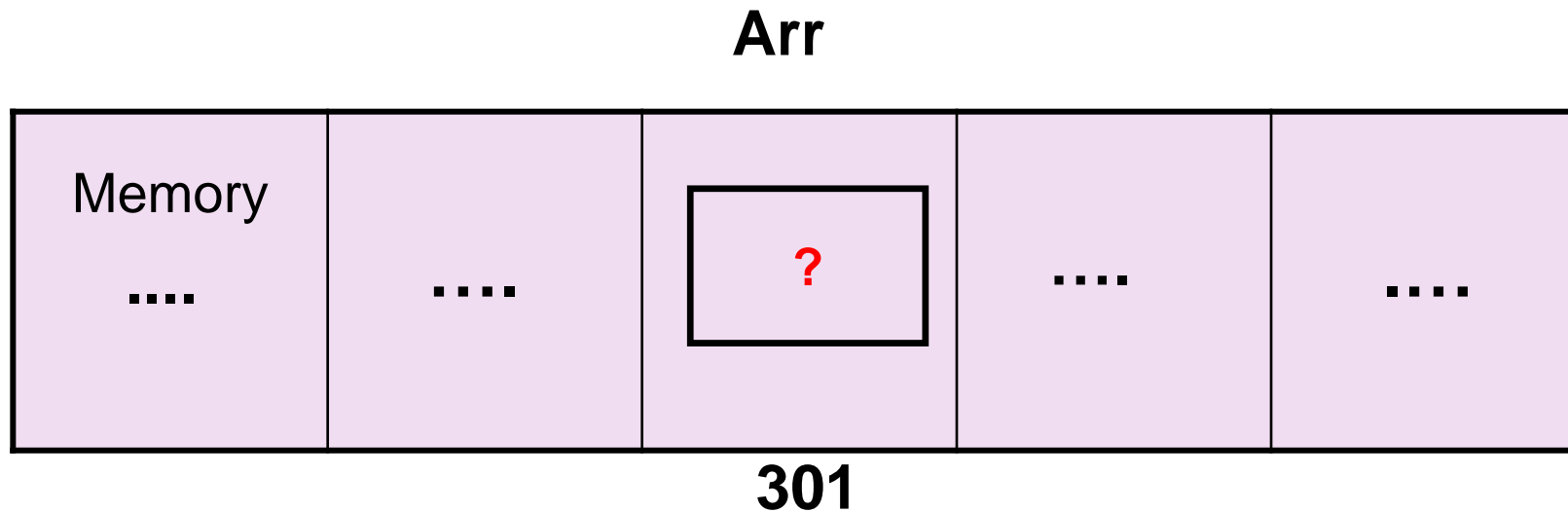


```
int[ ] Arr;  
Arr=new int[3];  
Arr[0]=7;  
Arr[1]=2;  
Arr[2]=12;
```

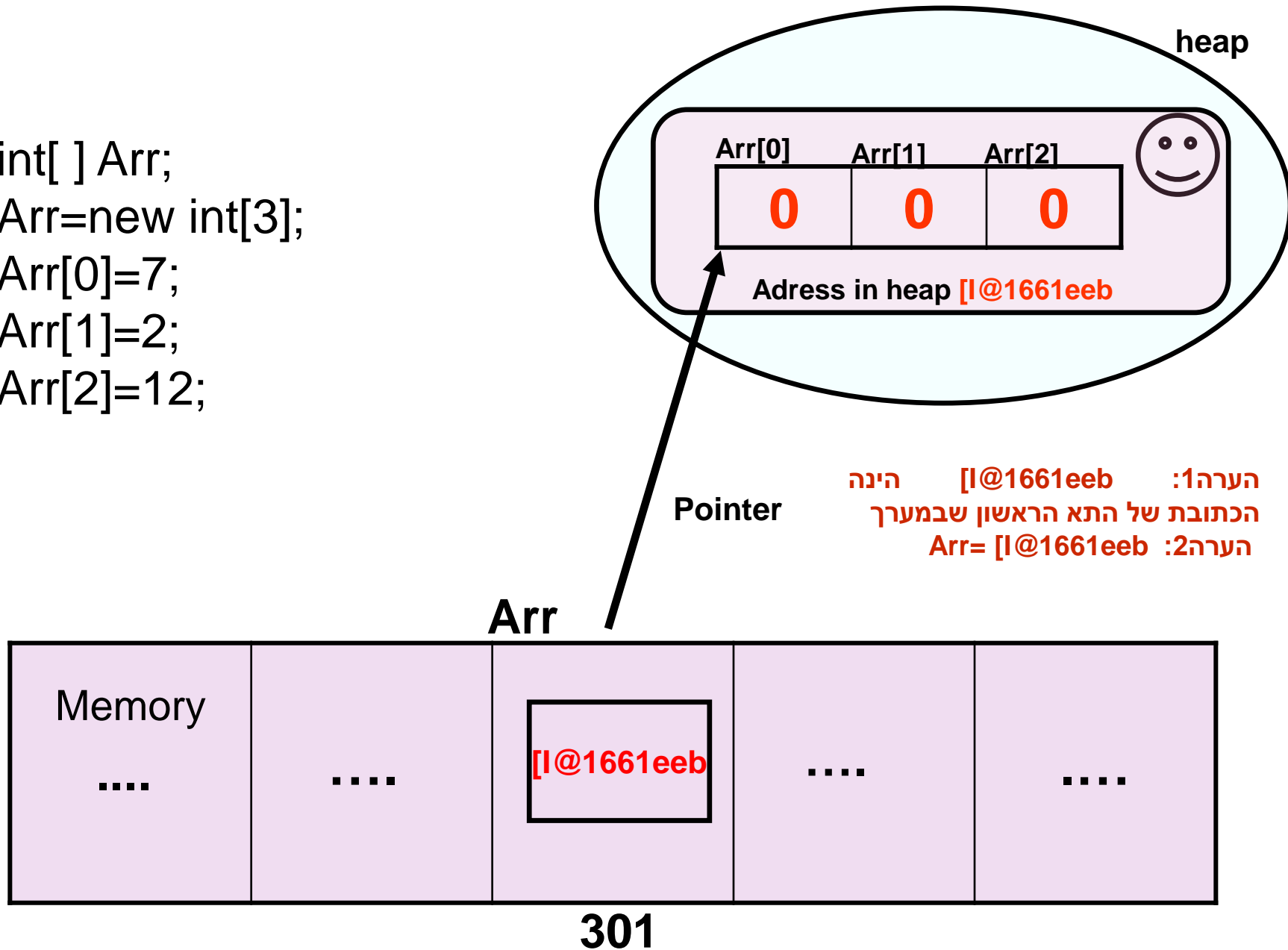


301

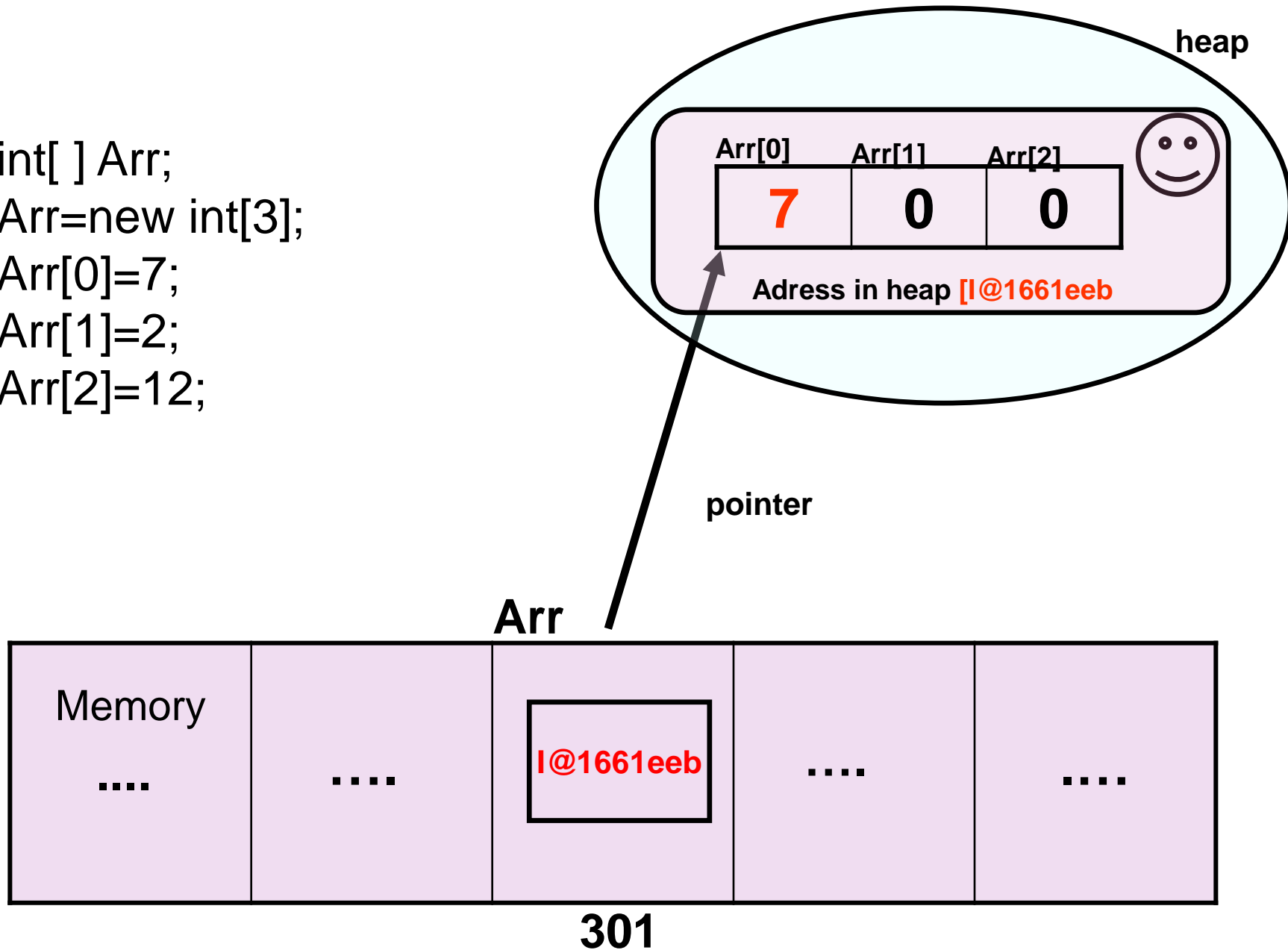
➡ `int[] Arr;`
`Arr=new int[3];`
`Arr[0]=7;`
`Arr[1]=2;`
`Arr[2]=12;`



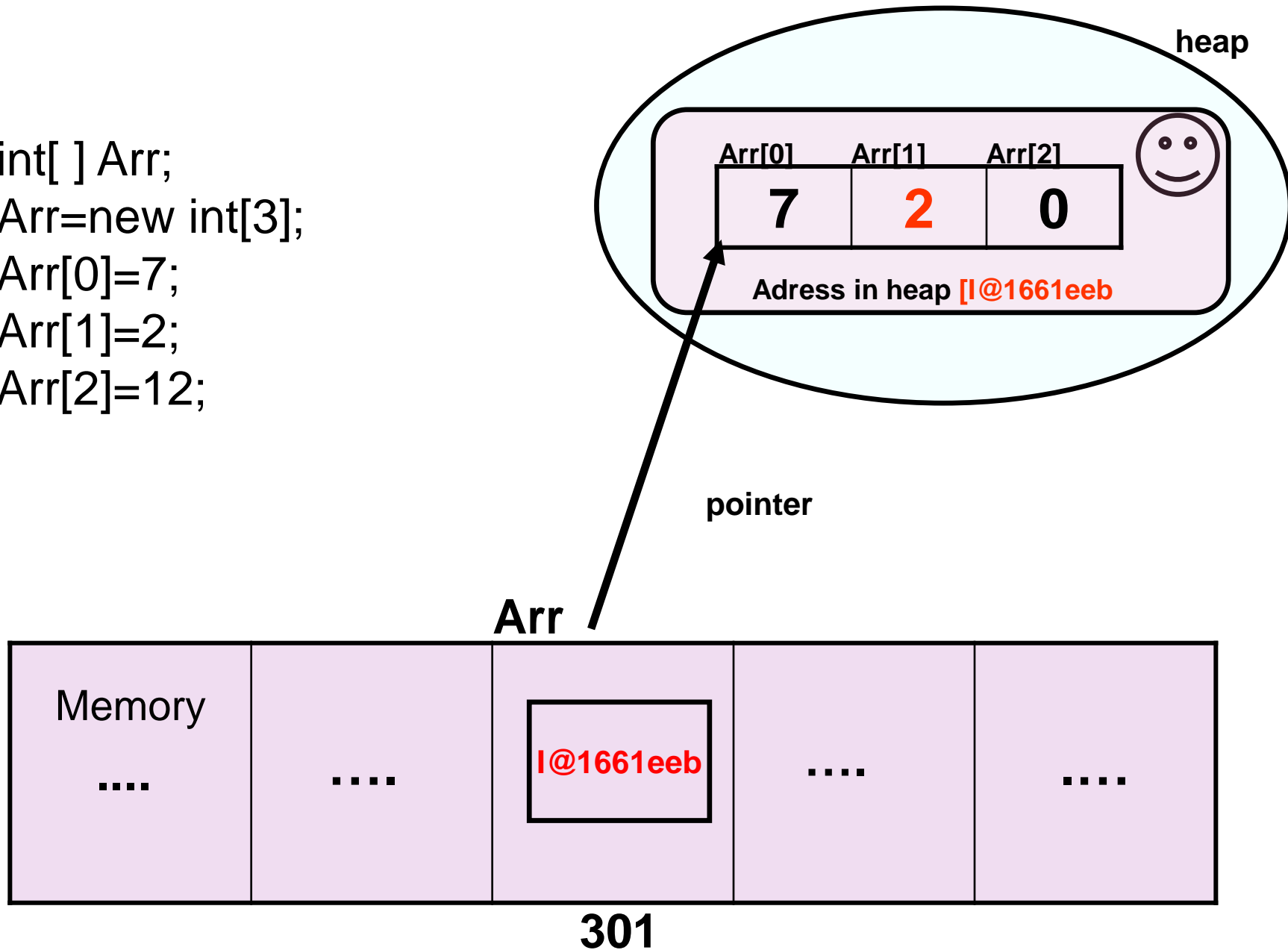
→
int[] Arr;
Arr=new int[3];
Arr[0]=7;
Arr[1]=2;
Arr[2]=12;



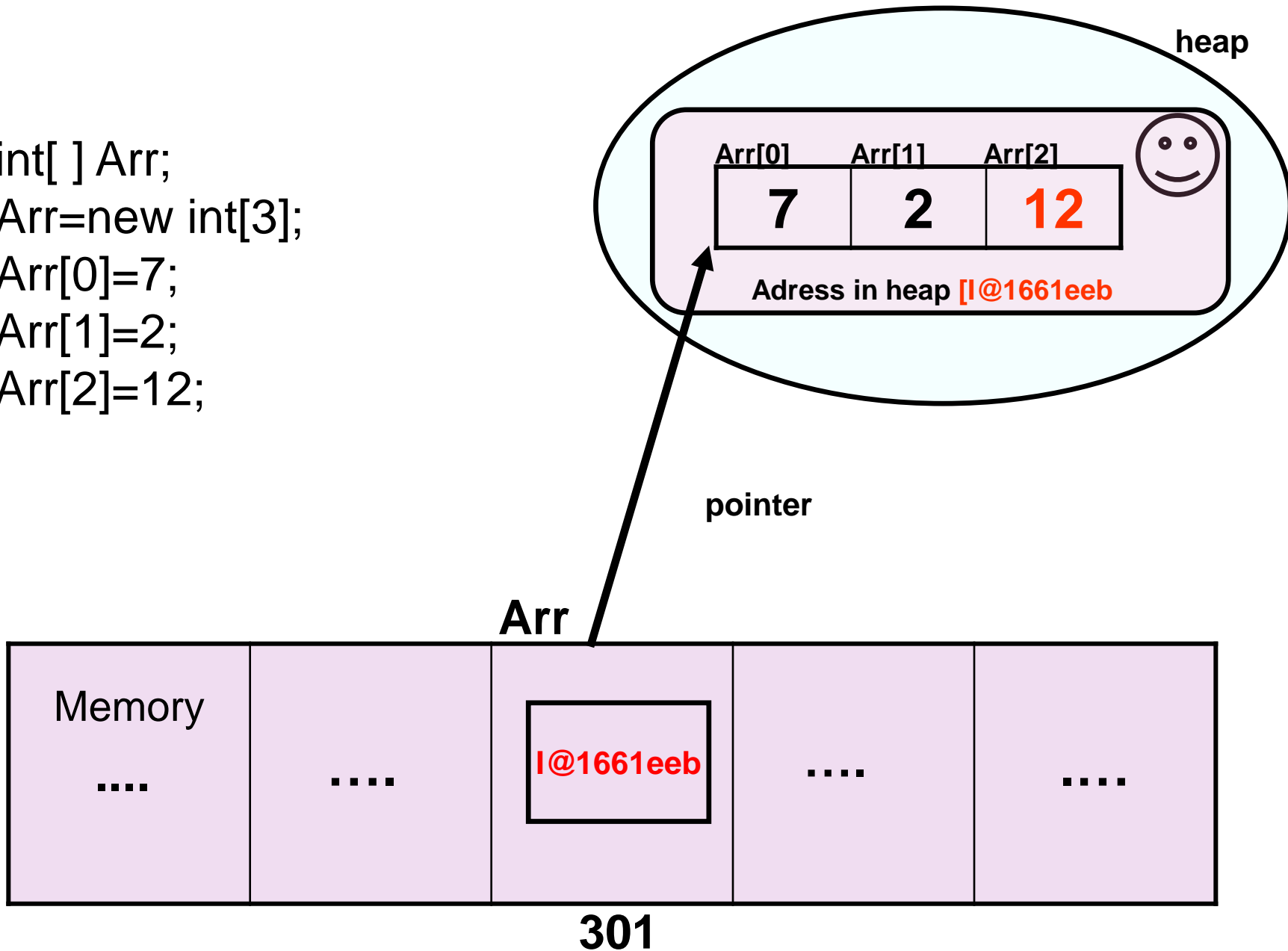
```
int[ ] Arr;  
Arr=new int[3];  
➔ Arr[0]=7;  
Arr[1]=2;  
Arr[2]=12;
```




```
int[ ] Arr;  
Arr=new int[3];  
Arr[0]=7;  
➔ Arr[1]=2;  
Arr[2]=12;
```

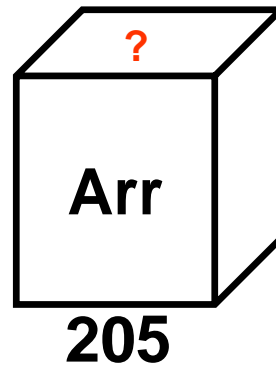


```
int[ ] Arr;  
Arr=new int[3];  
Arr[0]=7;  
Arr[1]=2;  
➡ Arr[2]=12;
```

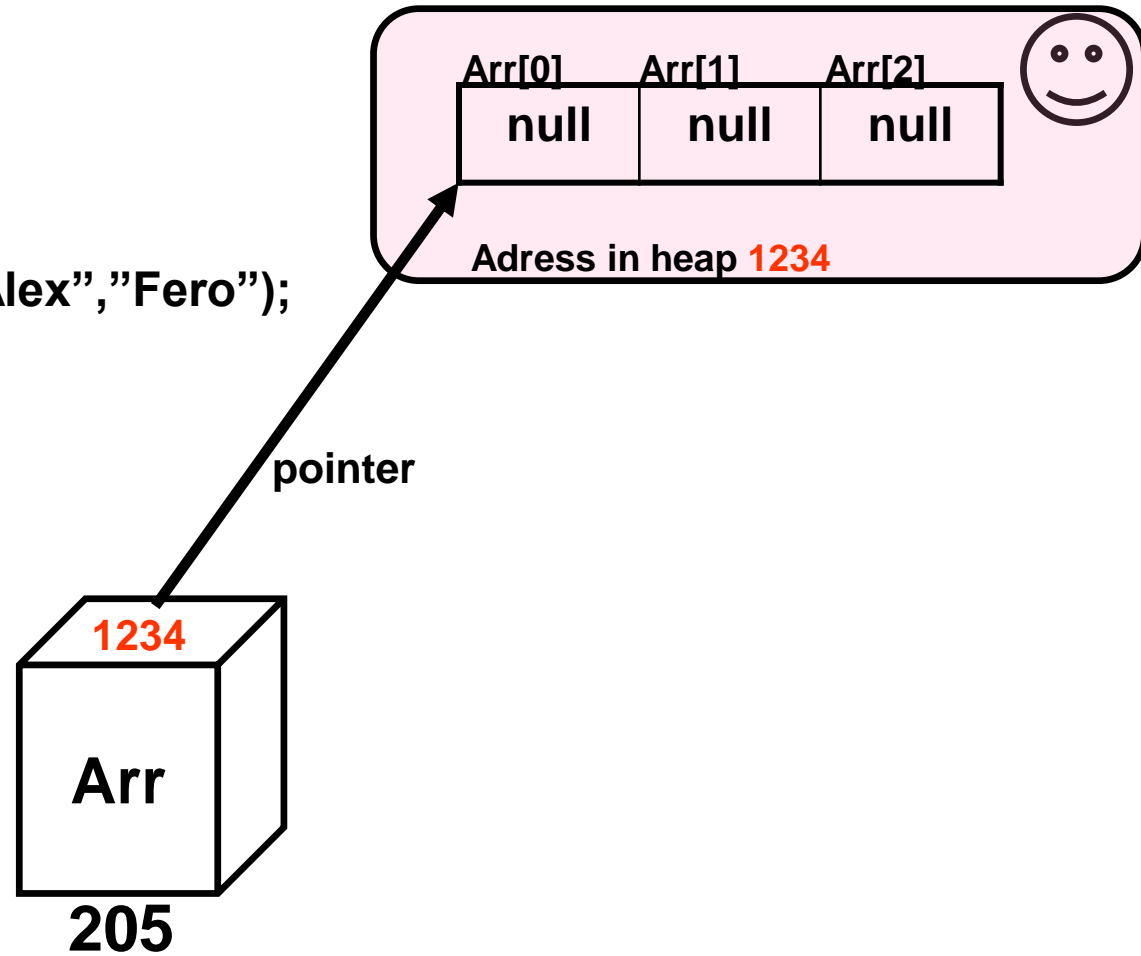


עוד דוגמא

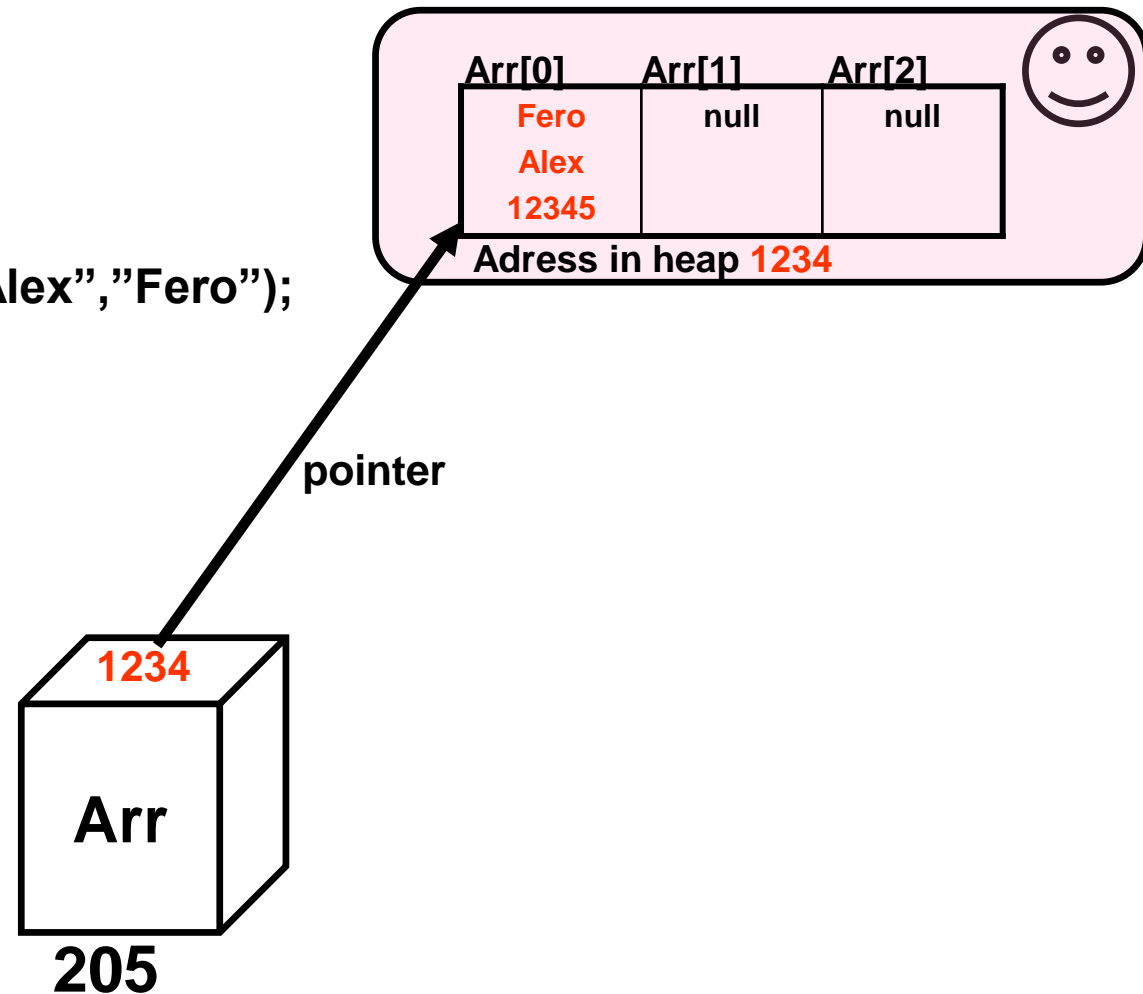
```
➡ Person[ ] Arr;  
Arr=new Person[3];  
Arr[0]=new Person(12345,"Alex","Fero");
```



```
Person[ ] Arr;  
➡ Arr=new Person[3];  
Arr[0]=new Person(12345,"Alex","Fero");
```

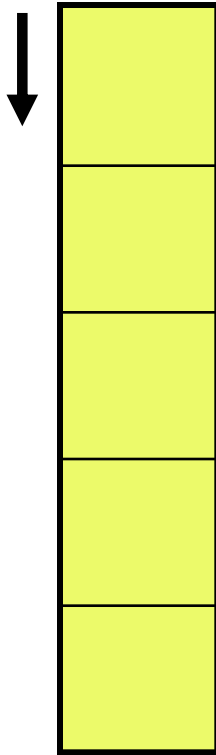


```
Person[ ] Arr;  
Arr=new Person[3];  
➡ Arr[0]=new Person(12345,"Alex","Fero");
```

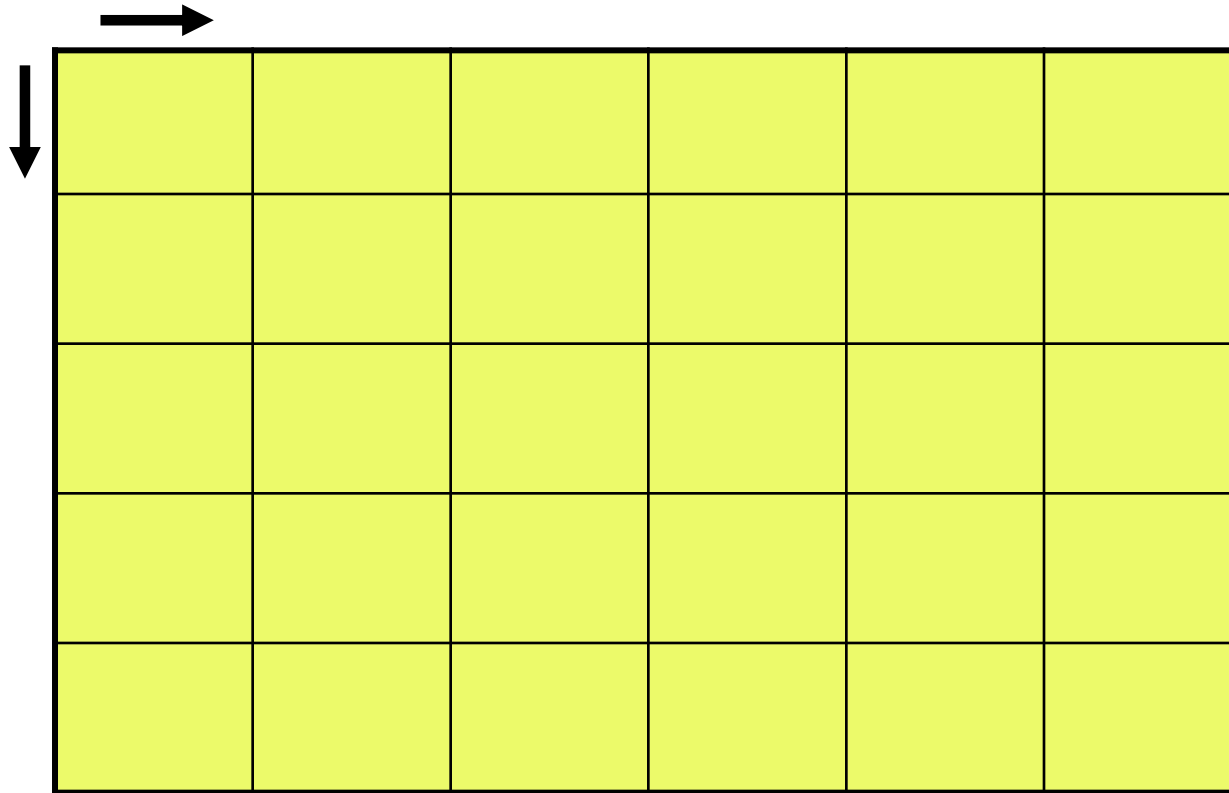


Tow Dimensional Array

One Dimension



Two Dimension



Declaration And Construction Of Two Dimensional Array

declaration

construction

`int[][] table;`
`table=new int[2][2];`

~~IX~~

`int[][] table=new int[2][2];`

0	0
0	0

declaration

construction

`String[][] table;`
`table=new String[2][2];`

~~IX~~

`String[][] table=new String[2][2];`

null	null
null	null

Assignment

Tow Dimensional Array

```
public static void main (String[] args ) {  
    int[ ][ ] table = new int[2][2];  
    arr[0][0] =1;  
    arr[0][1]=2;  
    arr[1][0]=3;  
    arr[1][1]=4;  
}
```

1	2
3	4

Declare + Construct + Initialize Two Dimensional Array - All In One Statement

```
public static void main (String[ ] args ) {  
    int[ ][ ] table={{1,2},  
                     {3,4}};  
}
```

1	2
3	4

Array Declaration VS. Two Dimensional Array Declaration

Array	Two Dimensional Array
<pre>int[] arr; arr=new int[2]; arr[0]=1;Arr[1]=2;</pre>	<pre>int[][] table; table = new int[2][2]; table[0][0]=1; table[0][1]=2; table[1][0]=3; table[1][1]=4;</pre>
<pre>int[] arr=new int[2]; arr[0]=1;Arr[1]=2;</pre>	<pre>int[][] table = new int[2][2]; table[0][0]=1; table[0][1]=2; table[1][0]=3; table[1][1]=4;</pre>
<pre>int[] arr={1,2};</pre>	<pre>int[][] table={ {1,2}, {3,4} };</pre>
<pre>----</pre>	<pre>int[][] table=new int[2][]; table[0]=new int[3]; table[1]=new int[5];</pre>

Printing Two Dimensional Array Using Loops

```
public static void main (String[ ] args ) {  
    int[ ][ ] table= { {1,2} , {3,4} };  
  
    for ( int i=0 ; i<table.length ; i++) {  
        for ( int j=0 ; j<table[0].length ;j++ ) {  
            System.out.println( table[ i ][ j ] );  
        }  
    }  
}
```

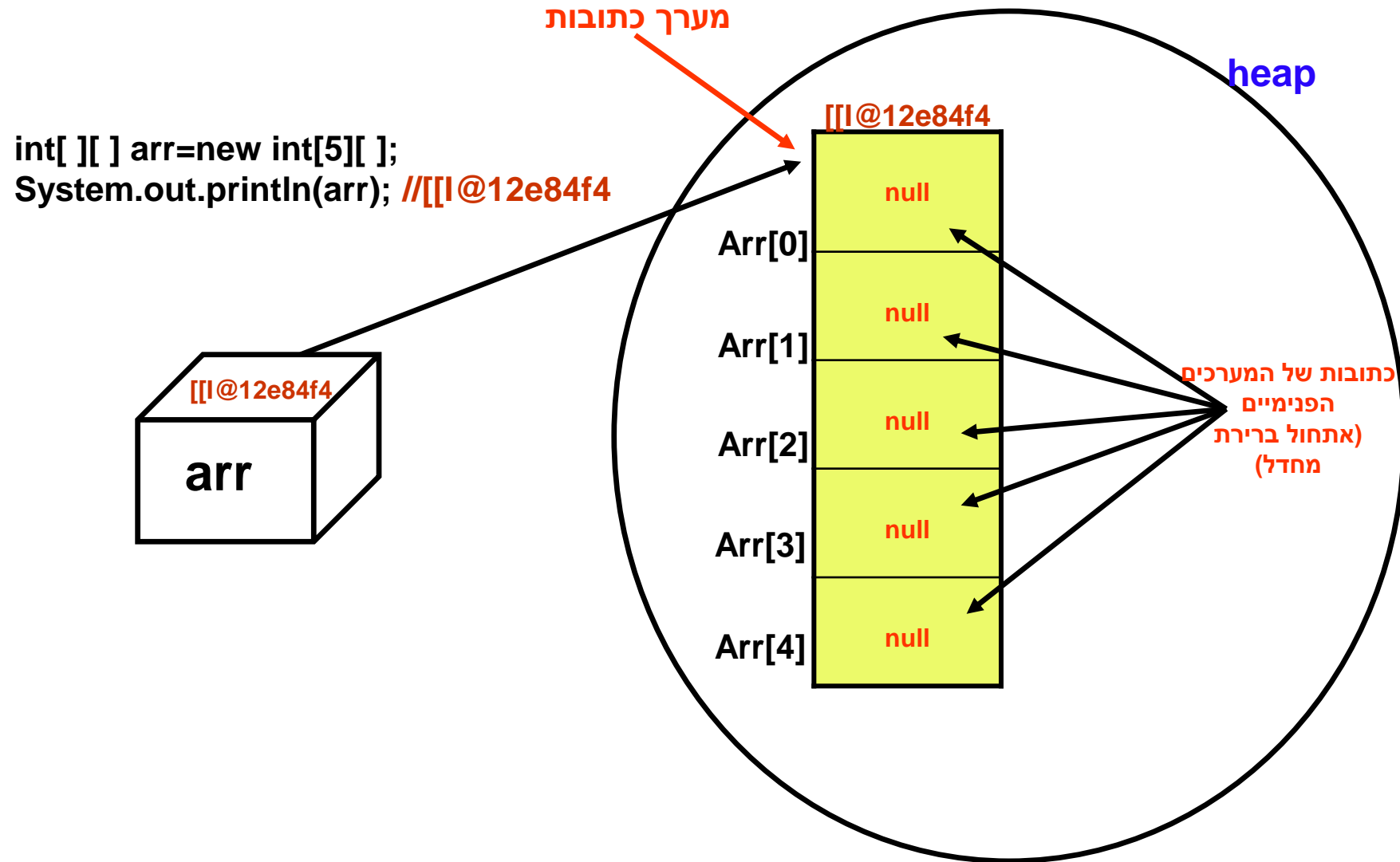
1	2
3	4

Using Loops To Assign Two Dimensional Array

```
public static void main (String[] args ) {  
    final int M=2 , N=2;  
    int[ ][ ] table=new int[ M ][ N ];  
    Scanner scan=new Scanner(System.in);  
  
    System.out.println("Insert:"+M*N+" numbers");  
    for ( int i=0 ; i<M ; i++ ) {  
        for ( int j=0 ; j<N ; j++ ) {  
            table[ i ][ j ] = scan.nextInt();  
        }  
    }  
}
```

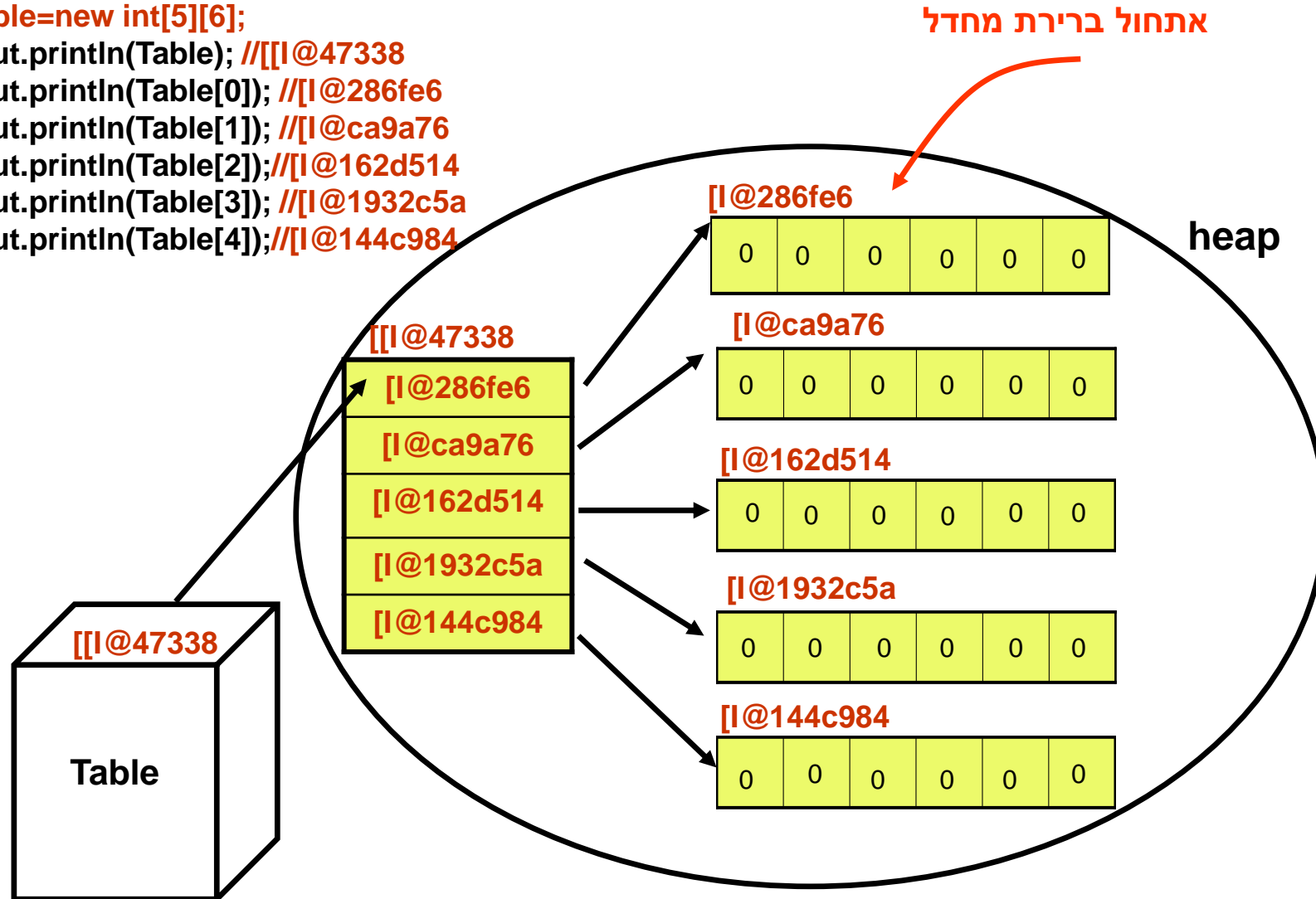
1	2
3	4

Array of Arrays



Array Of Arrays

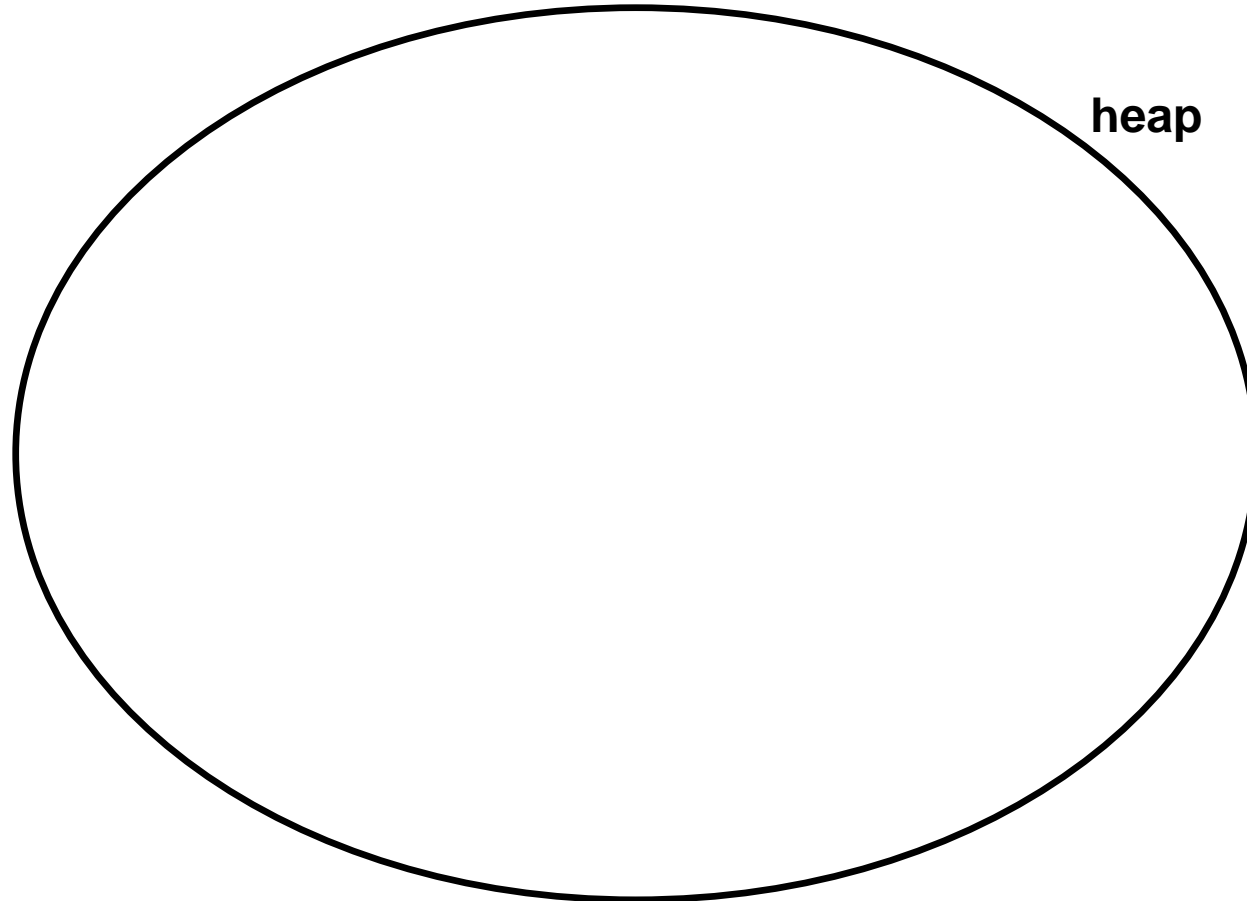
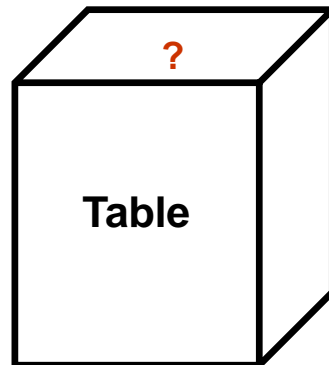
```
public class Tester{  
    public static void main(String[] args){  
        int[ ][ ] Table=new int[5][6];  
        System.out.println(Table); //[I@47338  
        System.out.println(Table[0]); //[I@286fe6  
        System.out.println(Table[1]); //[I@ca9a76  
        System.out.println(Table[2]); //[I@162d514  
        System.out.println(Table[3]); //[I@1932c5a  
        System.out.println(Table[4]); //[I@144c984  
    }  
}
```



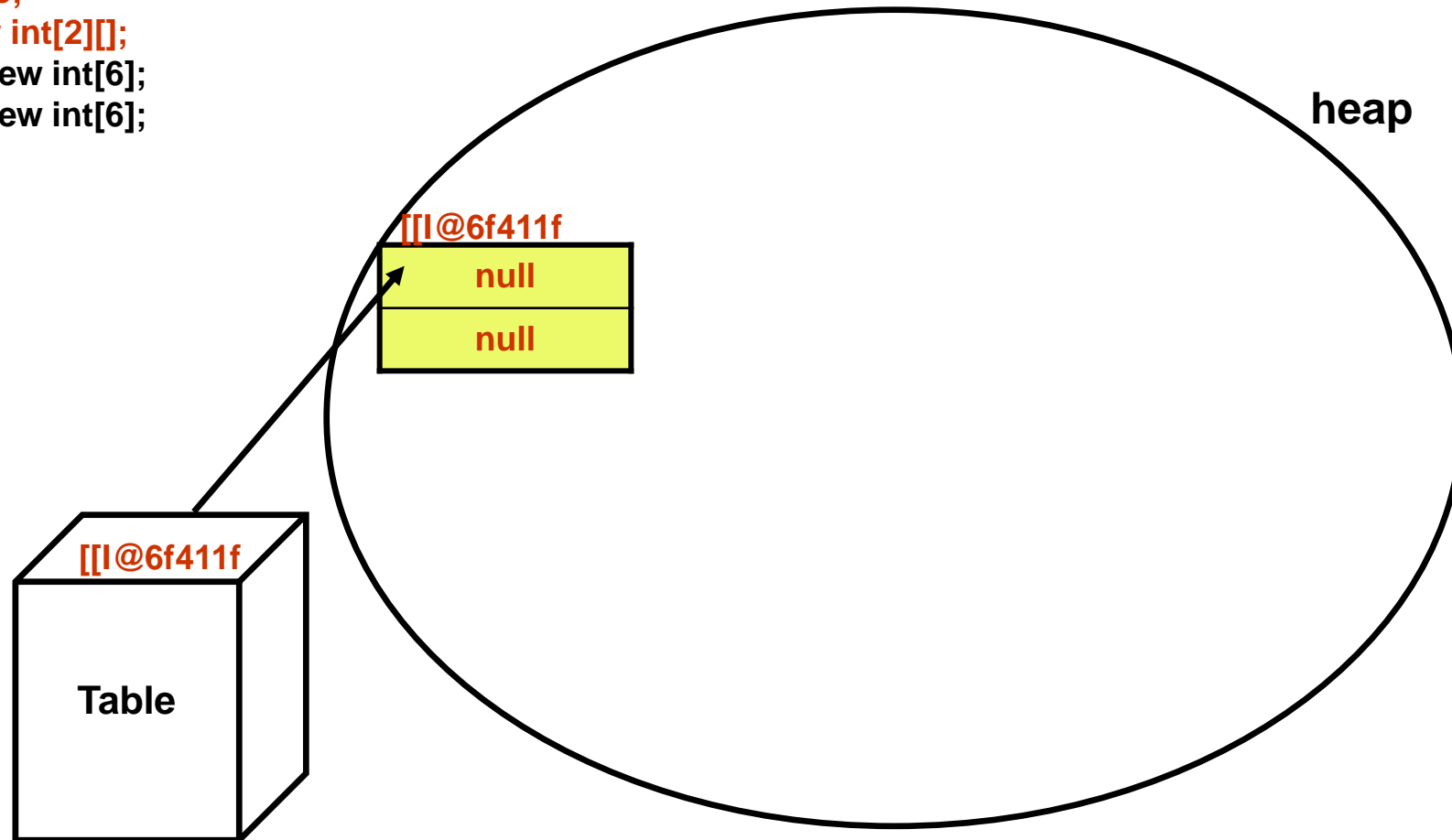
Array Of Arrays

מעקב

```
public class Tester{  
    public static void main(String[] args){  
        ➡ int[ ][ ] table;  
        table=new int[2][];  
        table[0]=new int[6];  
        table[1]=new int[6];  
    }  
}
```

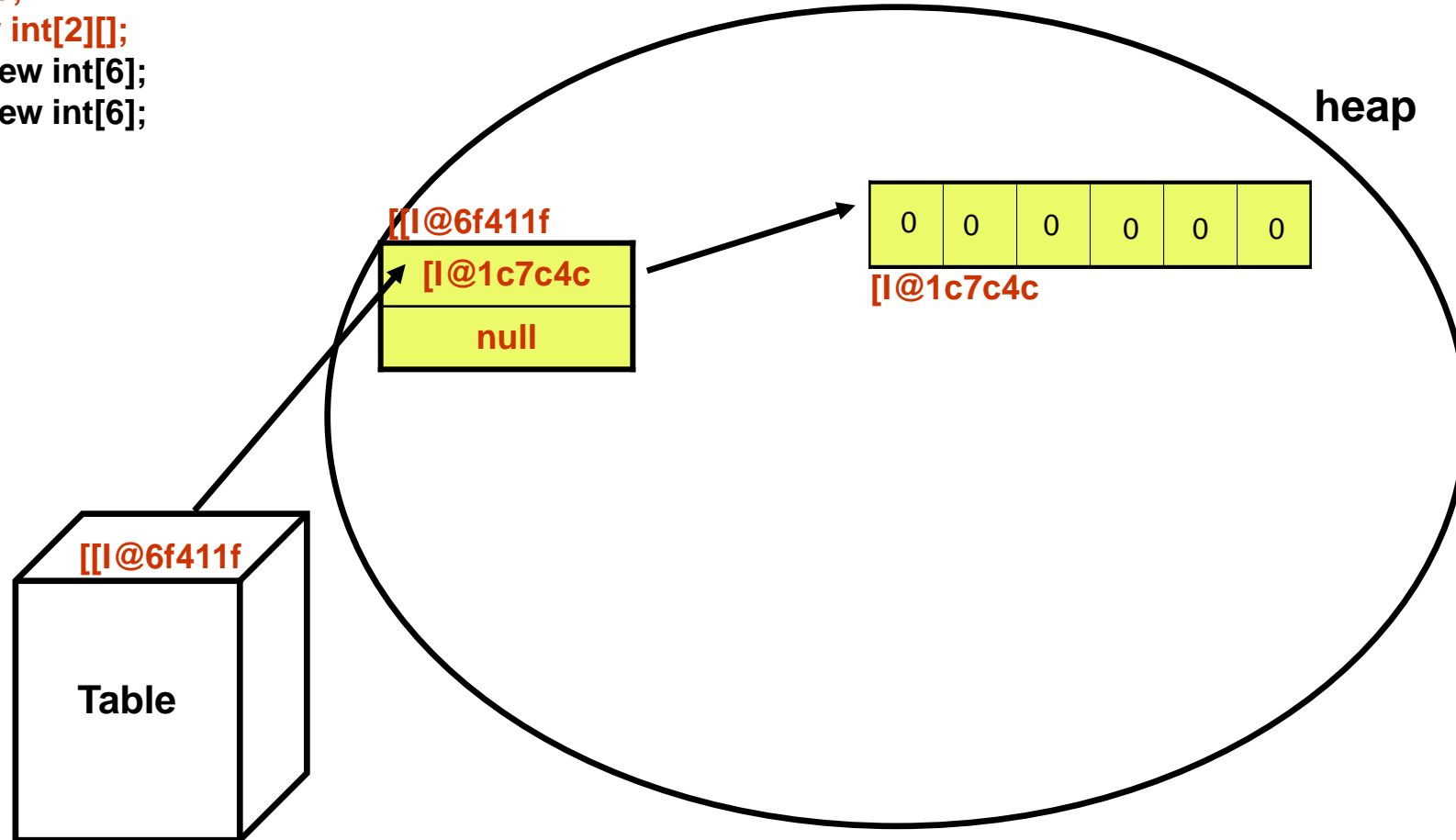


```
public class Tester{  
    public static void main(String[] args){  
        int[][] Table;  
        ➡ Table=new int[2][];  
        Table[0]=new int[6];  
        Table[1]=new int[6];  
    }  
}
```



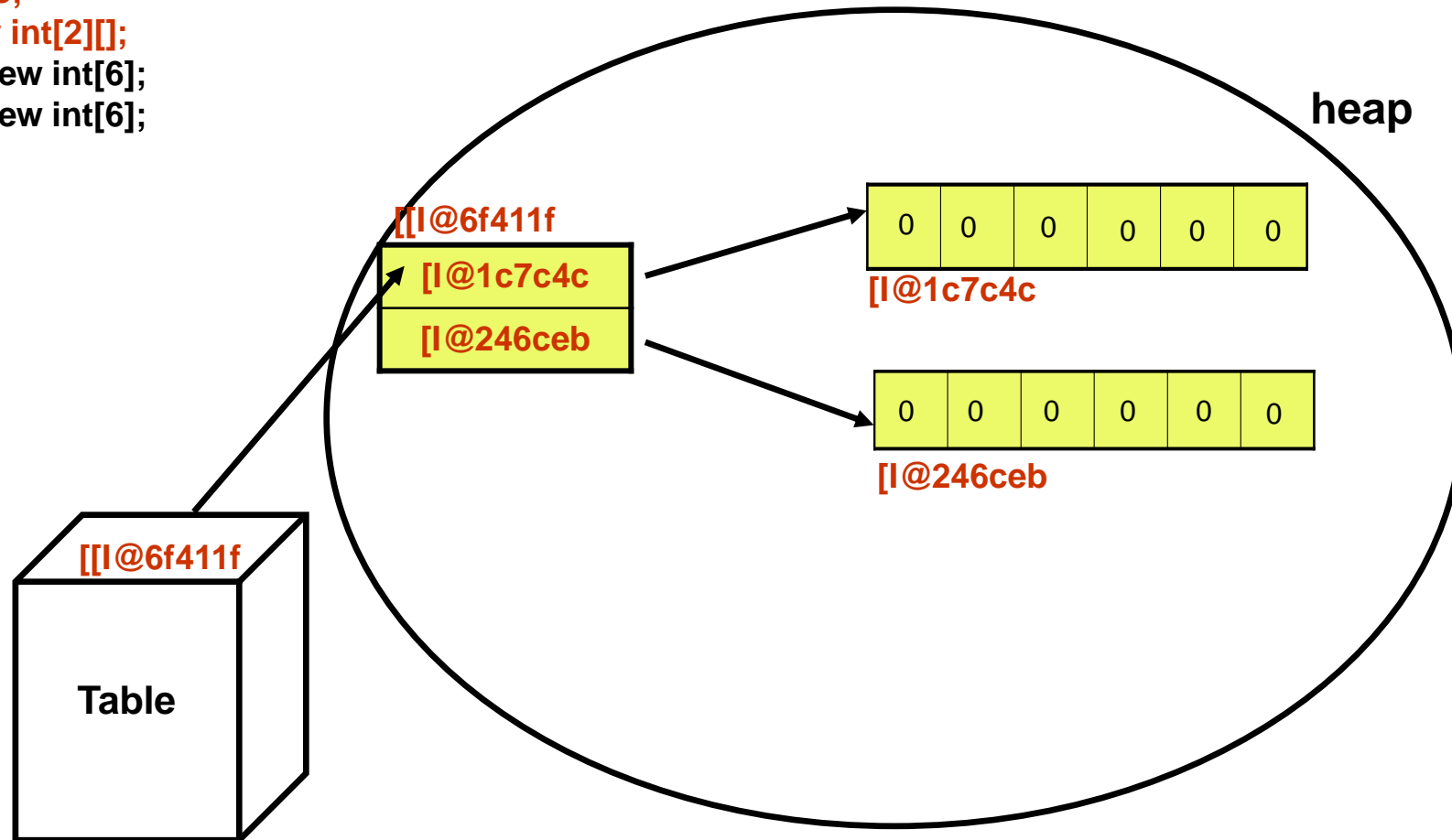
Array To Array

```
public class Tester{  
    public static void main(String[] args){  
        int[][] Table;  
        Table=new int[2][];  
        ➡ Table[0]=new int[6];  
        Table[1]=new int[6];  
    }  
}
```



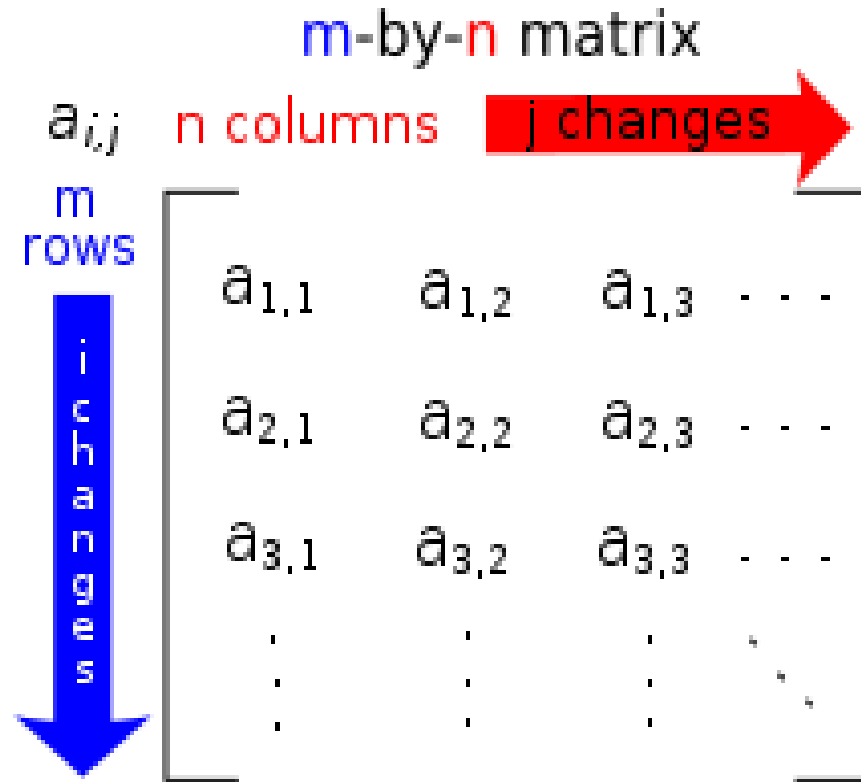
Array To Array

```
public class Tester{  
    public static void main(String[] args){  
        int[][] Table;  
        Table=new int[2][];  
        Table[0]=new int[6];  
        ➡ Table[1]=new int[6];  
    }  
}
```



Matrix

מטריצה



[http://en.wikipedia.org/wiki/Matrix_\(mathematics\)](http://en.wikipedia.org/wiki/Matrix_(mathematics))

- מטריצה מסדר n על m (ח- m טבעיים)
- היא מערך דו-ממדי שבו n שורות ו- m עמודות.
- רכיבי המטריצה הם בדרך כלל מספרים $a_{i,j}$
- את הרכיבים מסמנים בזוג אינדקסים
- מטריצה ריבועית היא מטריצה שבה מספר השורות שווה למספר העמודות כלומר $m=n$.

$$A = \begin{bmatrix} 9 & 13 & 5 \\ 1 & 11 & 7 \\ 3 & 7 & 2 \\ 6 & 0 & 7 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

לא ריבועית ריבועית

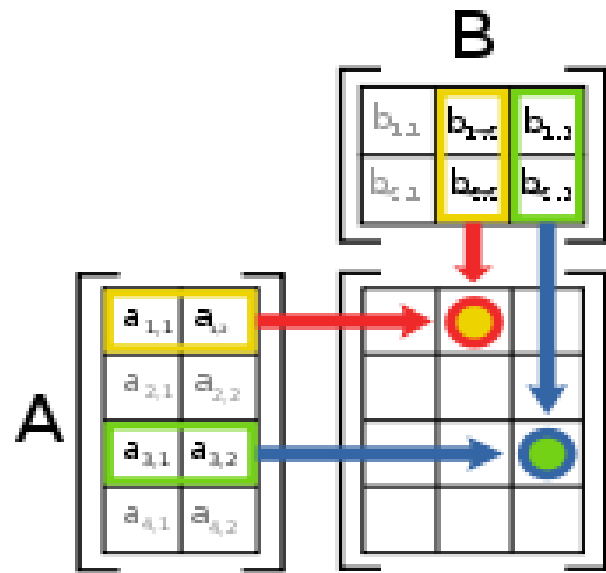
פעולות שניתן לבצע על מטריצות

Operation	Definition	Example
Addition	<p>The <i>sum</i> $A+B$ of two m-by-n matrices A and B is calculated entrywise:</p> <p>$(A+B)_{ij} = A_{ij} + B_{ij}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$.</p>	$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 1+5 \\ 1+7 & 0+5 & 0+0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 6 \\ 8 & 5 & 0 \end{bmatrix}$
Scalar multiplication	<p>The <i>scalar multiplication</i> cA of a matrix A and a number c (also called a <i>scalar</i> in the parlance of <i>abstract algebra</i>) is given by multiplying every entry of A by c:</p> <p>$(cA)_{ij} = c \cdot A_{ij}$.</p>	$2 \cdot \begin{bmatrix} 1 & 8 & -3 \\ 4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 2 \cdot 8 & 2 \cdot -3 \\ 2 \cdot 4 & 2 \cdot -2 & 2 \cdot 5 \end{bmatrix} = \begin{bmatrix} 2 & 16 & -6 \\ 8 & -4 & 10 \end{bmatrix}$
Transpose	<p>The <i>transpose</i> of an m-by-n matrix A is the n-by-m matrix A^T (also denoted A^{tr} or tA) formed by turning rows into columns and vice versa:</p> <p>$(A^T)_{ij} = A_{j,i}$.</p>	$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & 7 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 7 \end{bmatrix}$

[http://en.wikipedia.org/wiki/Matrix_\(mathematics\)](http://en.wikipedia.org/wiki/Matrix_(mathematics))

Matrix Multiplication

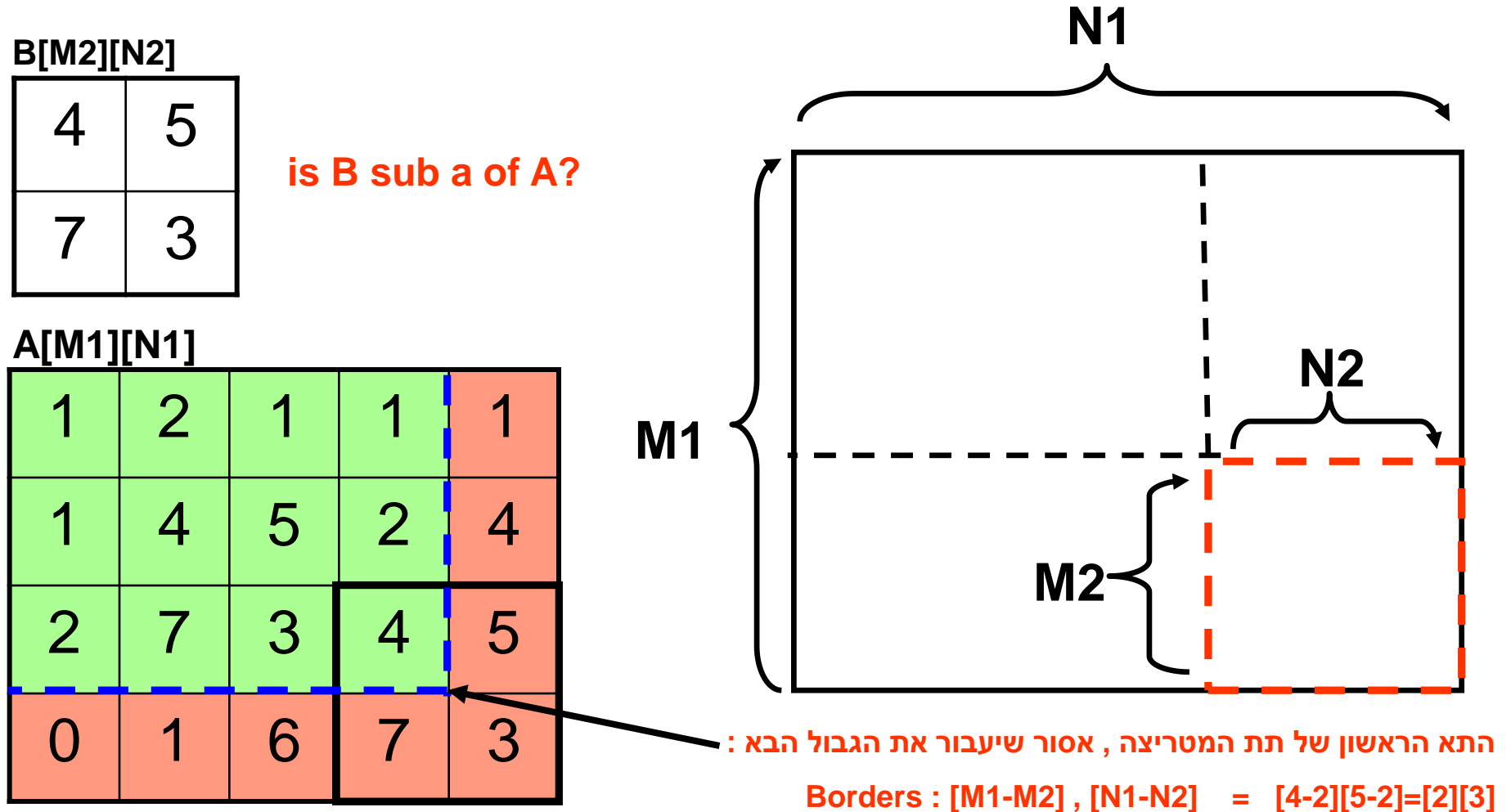
כפל מטריצות



$$\begin{bmatrix} \underline{2} & \underline{3} & \underline{4} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \underline{1000} \\ 1 & \underline{100} \\ 0 & \underline{10} \end{bmatrix} = \begin{bmatrix} 3 & \underline{2340} \\ 0 & 1000 \end{bmatrix}.$$

$$[AB]_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \cdots + A_{i,n}B_{n,j} = \sum_{r=1}^n A_{i,r}B_{r,j}$$

Borders of a Sub Matrix



תוכנית אשר מבצעת חיפוש

איבר במטריצה

```
public class Tester {  
    public static void main (String[ ] args ) {  
        int[ ][ ] table={{1,2,3,4},  
                           {5,1,13,19},  
                           {4,7,1,0},  
                           {2,5,8,1}};  
  
        boolean stop=true;  
        int num=19;  
        for ( int i=0 ; i<table.length && stop ; i++) {  
            for ( int j=0; j<table[0].length && stop ; j++) {  
                if ( table[ i ][ j ] == num ) {  
                    stop = false;  
                }  
            }  
        }  
        if ( stop == true )  
            System.out.println("no");  
        else  
            System.out.println("yes");  
    }  
}
```

תוכנית אשר מדפיסה את האלכסון הראשי במטריצה ריבועית

```
public class Tester {  
    public static void main (String[ ] args ) {  
        int[ ][ ] table={{1,2,3,4},  
                           {5,8,8,0},  
                           {4,7,2,0},  
                           {2,5,8,6}};  
        for ( int i=0 ; i<table.length ; i++) {  
            for (int j=0 ; j<table[0].length ; j++) {  
                if ( i==j ) {  
                    System.out.println ( table[ i ][ j ] );  
                }  
            }  
        }  
    }  
}
```


תוכנית אשר בודקת אם הערכים שיש על האלכסון הראשי במטריצה ריבועית שווים ל- 1

```
public class Tester{  
    public static void main (String[ ] args ) {  
        int[ ][ ] table={{1,2,3,4},  
                           {5,1,8,0},  
                           {4,7,1,0},  
                           {2,5,8,1}};  
  
        boolean stop=true;  
        for ( int i=0 ; i<table.length && stop ; i++ ) {  
            for(int j=0; j<table[0].length && stop ; j++ ) {  
                if ( i==j && table[ i ][ j ] !=1 ) {  
                    stop=false;  
                }  
            }  
        }  
        if ( stop==true )  
            System.out.println("yes");  
        else  
            System.out.println("no");  
    }  
}
```

תוכנית אשר בודקת אם הערכים שיש על האלכסון הראשי והמשני במטריצה ריבועית - שווים

```
public class Tester{  
    public static void main (String[ ] args ) {  
        int[ ][ ] table={{1,2,3,1},  
                           {5,2,2,19},  
                           {4,3,3,0},  
                           {4,5,8,4}};  
        boolean stop=true;  
        int num=19;  
        for ( int i=0 ; i<table.length && stop ; i++) {  
            for(int j=0 ; j<table[0].length && stop ; j++) {  
                if(i==j && table[ i ][ j ] != table[ i ][ table[0].length - 1 - j ] ) {  
                    stop=false;  
                }  
            }  
        }  
        if ( stop==true )  
            System.out.println("yes");  
        else  
            System.out.println("no");  
    }  
}
```

```

public class Matrix {
    private int [ ][ ] mat ;
    public Matrix ( int rows , int columns ) {...}
    public Matrix ( int[ ][ ] temp) {...}
    public Matrix ( Matrix other) {...}
    public boolean isSquare( ) {...}
    public int getElement ( int row , int col ) {...}
    public void setElement(int row ,int col ,int num){...}
    public String toString(){...}
    public boolean equals(Matrix other){ ...}
    public Matrix scalarMult(int num){...}
    public Matrix sum(Matrix other){...}
    public Matrix transpose(){...}
    public int[ ][ ] toInt(){...}
    public boolean isSub(Matrix other){...}
}

```

תרגיל

כתוב את המחלקה Matrix אשר מייצגת מטריצה של שלמים ע"י שימוש במערך דו-ממדי (מטיפוס int). למחלקה הוגדרו שלושה בנאים , הראשון מקבל שני מספרים שלמים שמייצגים את גודל המערך הדו מימדי שיש ליצור. השני מקבל מערך דו מימדי והשלישי בנאי העתקה. בנוסף יש לממש את הפעולות הבאות

- **equals** אשר משווה בין שתי מטריצות
- **sum** אשר מחזירה מטריצה שהיא תוצאת החיבור של שתי מטריצות
- **scalarMult** אשר מחזירה מטריצה שהיא תוצאת הכפל של מטריצה בסקלר
- **toString** אשר מחזירה יצוג של המטריצה כמחרוזת.
- **transpose** אשר מחזירה את המטריצה החלופית
- **toInt** שמחזירה מערך דו מימדי שהוא שכפול של המערך הדו מימדי שמייצג את המטריצה
- **isSub** בודקת אם המטריצה שהופעלה עליה השיטה היא תת מטריצה של זו שהתקבלה כפרמטר
- **isSquare** בודקת אם המטריצה היא ריבועית
- בנוסף הוגדרו שיטות **set** ו **get**

ראה טסטר בעמוד הבא

```

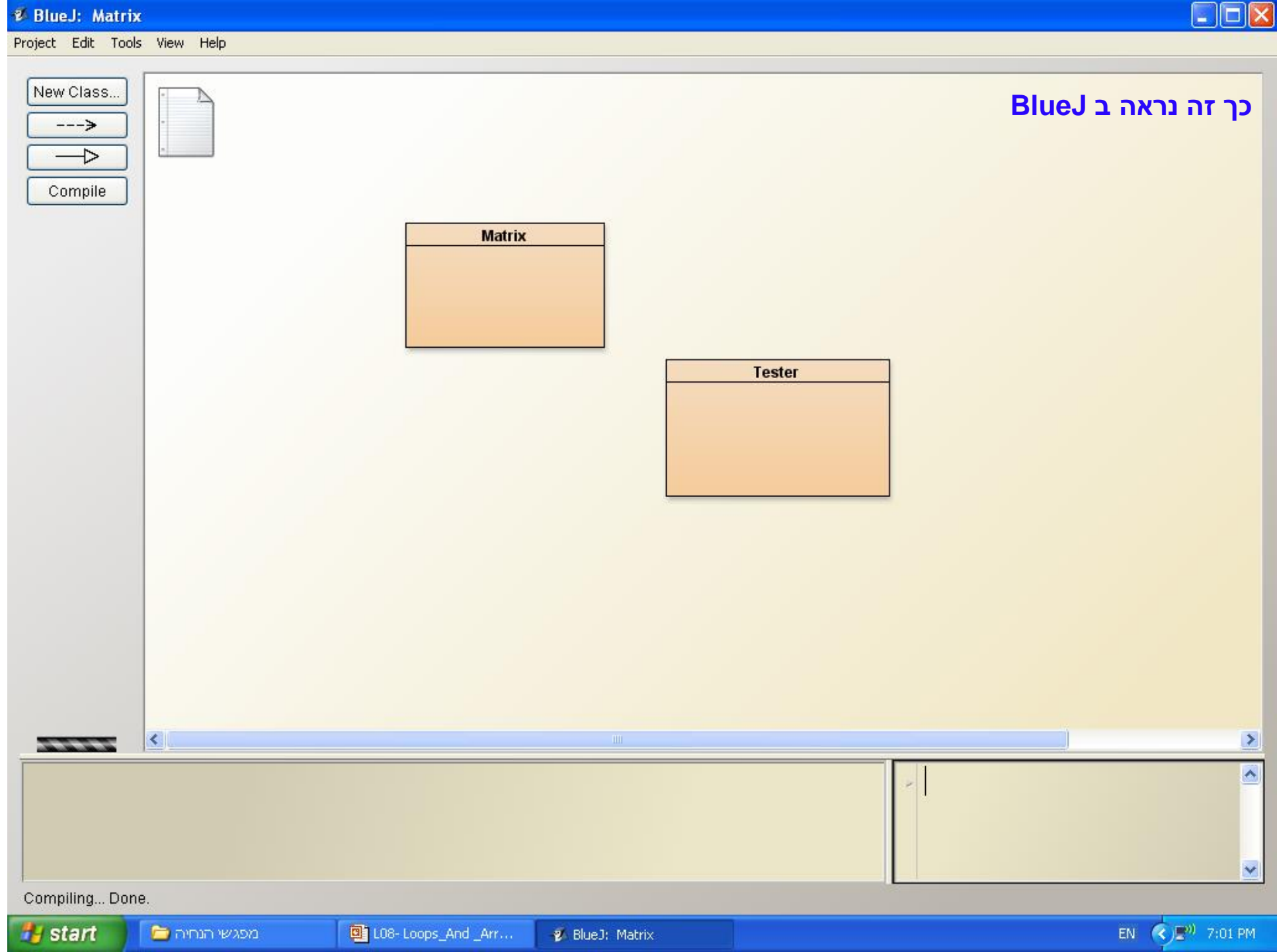
public class Tester {
    public static void main (String[] args ) {
        int [ ][ ] table1={ {1,2,3}, {4,5,6} , {7,8,9} };
        int [ ][ ] table2={ {5,6} , {8,9} };
        Matrix m1=new Matrix(2,3);
        Matrix m2=new Matrix(table1);
        Matrix m3=new Matrix(m2);
        System.out.println(m1+"\n"+m2+"\n"+ m3);
        System.out.println(m2.isSquare());
        System.out.println(m2.equals(m3));
        m1=m2.scalarMult(2);
        System.out.println(m1);
        m1=m2.sum(m3);
        System.out.println(m1);
        m1=m2.transpose();
        System.out.println(m1);
        m1=new Matrix(table2);
        System.out.println(m1);
        System.out.println(m1.isSub(m2));
        m1.setElement(0,0,-1);
        System.out.println(m1.getElement(0,0));
        int[ ][ ] temp = m1.toInt();
        Matrix m4=new Matrix(temp);
        System.out.println(m4);
    }
}

```

```

Size:2,3
0      0      0
0      0      0
Size:3,3
1      2      3
4      5      6
7      8      9
Size:3,3
1      2      3
4      5      6
7      8      9
true
true
Size:3,3
2      4      6
8      10     12
14     16     18
Size:3,3
2      4      6
8      10     12
14     16     18
Size:3,3
1      4      7
2      5      8
3      6      9
Size:2,2
5      6
8      9
true
-1
Size:2,2     -1      6
              8      9

```



כך זה נראה ב BlueJ

```

public Matrix(int rows , int columns) {
    if ( rows>0 && columns>0 ) {
        mat=new int[rows][columns];
    }
    else {
        mat=new int[1][1];
    }
}

```

```

public Matrix(int[ ][ ] temp) {
    if ( temp!=null ) {
        int rows=temp.length;
        int columns=temp[0].length;
        mat=new int[rows][columns];
        for( int i=0 ; i<rows ; i++) {
            for( int j=0; j<columns; j++) {
                mat[i][j] = temp[i][j];
            }
        }
    }
    else {
        mat=new int[1][1];
    }
}

```

```

public Matrix(Matrix other) {
    if( other!=null ) {
        int rows=other.mat.length;
        int columns=other.mat[0].length;
        mat=new int[rows][columns];
        for( int i=0 ; i<rows ; i++) {
            for( int j=0 ; j<columns ; j++) {
                mat[i][j]=other.mat[i][j];
            }
        }
    }
    else {
        mat=new int[1][1];
    }
}

```

```

public boolean isSquare(){
    return mat.length == mat[0].length;
}
public int getElement(int i,int j){
    return mat[i][j]; // i , j in range
}
public void setElement(int i,int j,int num){
    mat[i][j]=num; // i , j in range
}

```

```

public String toString(){
    int r=mat.length;
    int c=mat[0].length;
    String temp="";
    for ( int row=0 ; row<r ; row++ ) {
        for ( int col=0 ; col<c ; col++ ) {
            if ( col==0 )
                temp += mat[row][col];
            else
                temp += "\t" + mat[row][col];
        }
        temp += "\n";
    }
    return "Size:" + r + "," + c + "\n" + temp;
}

```

```

public boolean equals(Matrix other){
    if(other==null)
        return false;
    int rows=mat.length;
    int columns=mat[0].length;
    int orows=other.mat.length;
    int ocolumns=other.mat[0].length;
    if(rows!=orows || columns!=ocolumns)
        return false;
    for(int i=0;i<rows;i++){
        for(int j=0;j<columns;j++){
            if (mat[i][j]!=other.mat[i][j]){
                return false;
            }
        }
    }
    return true;
}

```

```

public Matrix scalarMult (int num) {
    int rows=mat.length;
    int columns=mat[0].length;
    Matrix temp=new Matrix(this);
    for ( int i=0 ; i<rows ; i++ ) {
        for ( int j=0 ; j<columns ; j++ ) {
            temp.mat[i][j] *= num;
        }
    }
    return temp;
}

```

```

public Matrix transpose( ) {
    int rows=mat.length;
    int columns=mat[0].length;
    Matrix temp=new Matrix(columns,rows);
    for ( int i=0 ; i<rows ; i++ ) {
        for ( int j=0 ; j<columns ; j++ ) {
            temp.mat[ j ][ i ] = mat[ i ][ j ];
        }
    }
    return temp;
}

```

```

public Matrix sum( Matrix other ) {
    int rows=mat.length;
    int columns=mat[0].length;
    Matrix temp=new Matrix(this);
    if(other!=null && rows==other.mat.length && columns==other.mat[0].length){
        for ( int i=0 ; i<rows ; i++ ) {
            for ( int j=0 ; j<columns ; j++ ) {
                temp.mat[i][j] += other.mat[i][j];
            }
        }
    }
    return temp;
}

```



```

public boolean isSub(Matrix other){
    if(other==null) return false;
    int m=other.mat.length-mat.length;
    int n=other.mat[0].length-mat[0].length;
    if(m<0||n<0) return false;
    for (int i=0 ; i<=m ; i++ ) {
        for ( int j=0 ; j<=n ; j++ ) {
            if ( isSub(i , j , other) ) {
                return true;
            }
        }
    }
    return false;
}

```

```

public int[ ][ ] toInt( ) {
    int rows=mat.length;
    int columns=mat[0].length;
    int[ ][ ] temp=new int[rows][columns];
    for ( int i=0 ; i<rows ; i++ ) {
        for ( int j=0 ; j<columns ; j++ ) {
            temp[ i ][ j ] = mat[ i ][ j ];
        }
    }
    return temp;
}

```

```

private boolean isSub(int r , int c , Matrix other){
    int rows=mat.length;
    int columns=mat[0].length;
    for ( int i=0 ; i<rows ; i++ ) {
        for ( int j=0 ; j<columns ; j++ ) {
            if ( mat[i][j] != other.mat[ i+r ][ j+c ] ) {
                return false;
            }
        }
    }
    return true;
}

```

Method Overloading



```

public class Fraction {
    private int numerator;
    private int denominator; //!=0
    public Fraction() {...}
    public Fraction (int n, int d) {...}
    public Fraction (Fraction f) {...}
    public boolean equals (Fraction f) {...}
    public Fraction sum (Fraction f) {...}
    public Fraction multScalar (int k) {...}
    public String toString() {...}
}

```

```

public class Tester {
    public static void main(String[ ] args) {
        Fraction f1=new Fraction();
        Fraction f2=new Fraction(1,2);
        Fraction f3=new Fraction(1,5);
        System.out.println(f1 + "\n" + f2 + "\n" + f3);
        f1=new Fraction(f2);
        System.out.println(f1);
        System.out.println(f1.equals(f2));
        f1=f2.sum(f3);
        System.out.println(f1);
        f1=f3.multScalar(2);
        System.out.println(f1);
    }
}

```

$\frac{3}{4}$

Fraction

שבר

תרגיל

במתמטיקה , שבר הוא מספר המוצג כחילוק של מספר שלם אחד במספר שלם שני שאיננו 0. יש ליישם את המחלקה Fraction אשר מתארת שבר עשרוני.

יש לממש את הפעולות הבאות

- equals אשר משווה בין שני שברים

- sum אשר מחברת שני שברים

- multScalar אשר מכפילה שבר בסקלר

- toString אשר מדפיסה את השבר

ניתן להיעזר בטסטר מלמטה

$$\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d + c \cdot b}{b \cdot d}$$

$$\frac{a}{b} \cdot k = \frac{a \cdot k}{b}$$

$$\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$$

Output:

0/1
1/2
1/5
1/2
true
7/10
2/5

```

public class Fraction {
    private int numerator;
    private int denominator;
    public Fraction ( ) { numerator = 0 ; denominator = 1 ; }
    public Fraction ( int n , int d ) { numerator=n; denominator=d; }
    public Fraction ( Fraction f ) {
        if ( f!=null ) {
            numerator = f.numerator; denominator = f.denominator;
        }
        else { numerator = 0; denominator = 1; }
    }
    public boolean equals ( Fraction f ) {
        return (f!=null) && ( numerator == f.numerator ) && ( denominator == f.denominator );
    }
    public Fraction sum ( Fraction f ) {
        if ( f!=null )
            return new Fraction ( numerator * f.denominator + denominator * f.numerator ,
                                   denominator * f.denominator );
        else return new Fraction(this);
    }
    public Fraction multScalar ( int k ) {
        return new Fraction ( numerator * k , denominator );
    }
    public String toString ( ) { return numerator + "/" + denominator; }
}

```

תרגיל

```
public class MatrixOfFraction {  
    private Fraction mat[ ][ ];  
    public MatrixOfFraction (int rows , int cols){...}  
    public MatrixOfFraction (Fraction [ ][ ] temp){...}  
    public MatrixOfFraction (MatrixOfFraction other){...}  
    public boolean equals(MatrixOfFraction other){...}  
    public MatrixOfFraction multScalar ( int num ){...}  
    public MatrixOfFraction sum(MatrixOfFraction other){...}  
    public double[ ][ ] toDoubleMat( ){...}  
    public Fraction[ ][ ] toFractionMat( ){...}  
    public String toString(){...}  
}
```

כתוב את המחלקה MatrixOfFraction אשר מייצגת מטריצה של Fraction-ים. למחלקה הוגדרו שלושה בנאים , הראשון מקבל שני מספרים שלמים שמייצגים את גודל המערך הדו מימדי שיש ליצור. השני מקבל מערך דו מימדי מסוג Fraction והשלישי בנאי העתקה. בנוסף יש לממש את הפעולות הבאות

- **equals** אשר משווה בין שתי מטריצות
- **sum** אשר מחזירה מטריצה שהיא תוצאת החיבור של שתי מטריצות
- **multScalar** אשר מחזירה מטריצה שהיא תוצאת הכפל של מטריצה בסקלר
- **toString** אשר מחזירה יצוג של המטריצה כמחרוזת.
- **toDoubleMat** שמחזירה מערך דו מימדי מסוג Double שאבריו הם מספרים ממשיים
- **toFractionMat** שמחזירה מערך דו מימדי מסוג Fraction

-ראה טסטר בעמוד הבאר

```

public class Tester{
    public static void main (String[] args ) {
        Fraction f1=new Fraction(1,2);
        Fraction f2=new Fraction(1,4);
        Fraction[ ][ ] table3={{f1,f2},
                                {f2,f1}};

        MatrixOfFraction mf1=new MatrixOfFraction(3,2);
        MatrixOfFraction mf2=new MatrixOfFraction(table3);
        MatrixOfFraction mf3=new MatrixOfFraction(mf2);
        System.out.println(mf1);
        System.out.println(mf2);
        System.out.println(mf3);
        System.out.println(mf2.equals(mf3));
        mf1=mf2.multScalar(2);
        System.out.println(mf1);
        mf1=mf2.sum(mf3);
        System.out.println(mf1);
        double [ ][ ] tmp1;
        tmp1=mf2.toDoubleArray();
        Fraction [ ][ ] tmp2;
        tmp2=mf2.toFractionArray();
    }
}

```

Output:

Size:3,2
(0/1) (0/1)
(0/1) (0/1)
(0/1) (0/1)

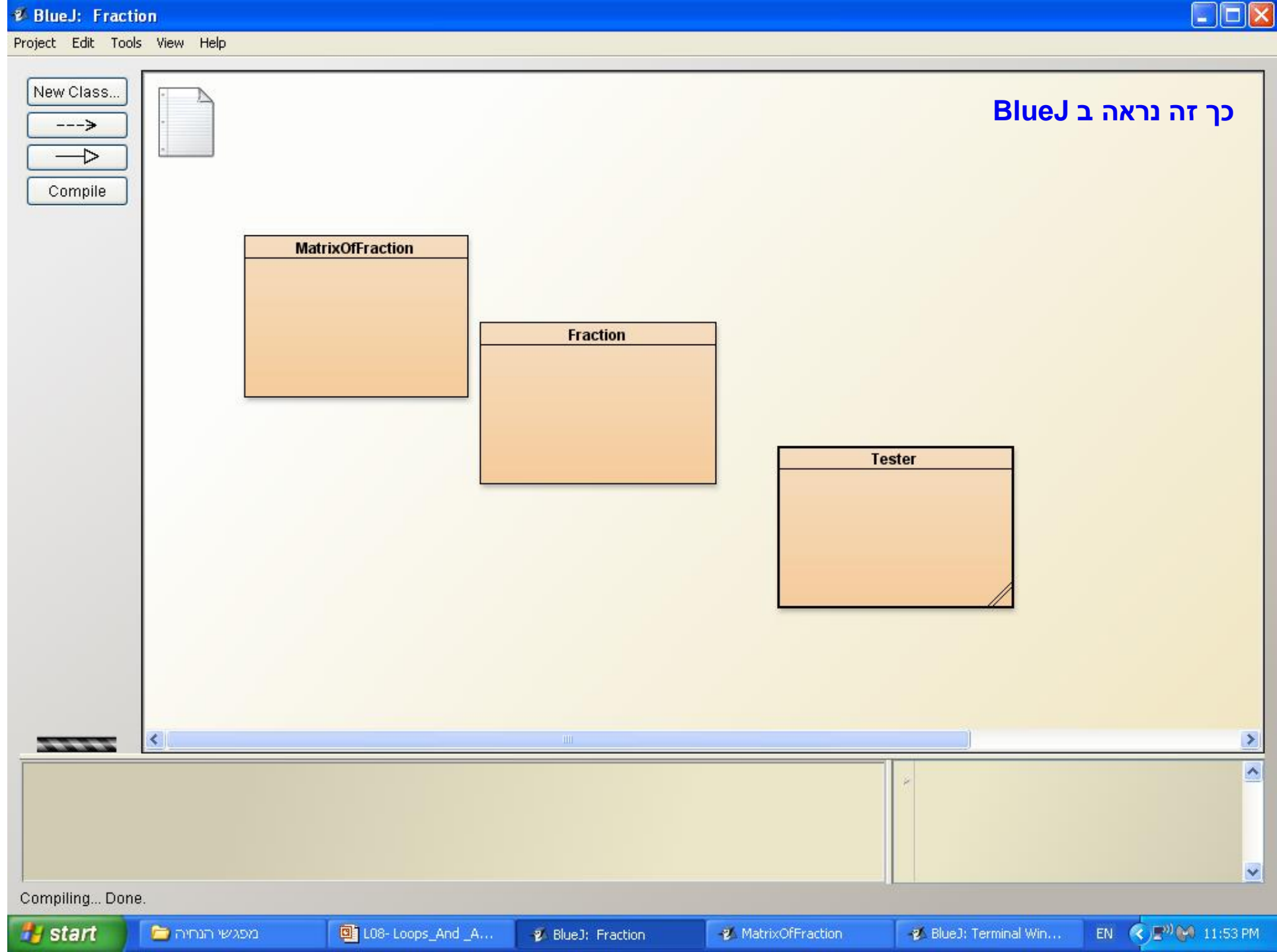
Size:2,2
(1/2) (1/4)
(1/4) (1/2)

Size:2,2
(1/2) (1/4)
(1/4) (1/2)

true

Size:2,2
(2/2) (2/4)
(2/4) (2/2)

Size:2,2
(4/4) (8/16)
(8/16) (4/4)



```

public MatrixOfFraction(int rows,int cols){
    if(rows>0 && cols>0){
        mat=new Fraction [rows][columns];
        for(int i=0 ; i<rows ; i++){
            for(int j=0 ; j<columns ; j++){
                mat[i][j]=new Fraction();
            }
        }
    }
    else {
        mat=new Fraction [1][1];
    }
}

```

```

public MatrixOfFraction(Fraction[ ][ ] temp) {
    if(temp!=null){
        int rows=temp.length;
        int columns=temp[0].length;
        mat=new Fraction [rows][columns];
        for(int i=0 ; i<rows ; i++){
            for(int j=0 ; j<columns ; j++){
                mat[i][j]=new Fraction(temp[i][j]);
            }
        }
    }
    else { mat=new Fraction [1][1]; }
}

```

```

public MatrixOfFraction (MatrixOfFraction other){
    if(other!=null){
        int rows=other.mat.length;
        int columns=other.mat[0].length;
        mat=new Fraction [rows][columns];
        for(int i=0;i<rows;i++){
            for(int j=0;j<columns;j++){
                mat[i][j]=new Fraction(other.mat[i][j]);
            }
        }
    }
    else { mat=new Fraction [1][1]; }
}

```

```
public boolean equals(MatrixOfFraction other) {  
    if(other==null)  
        return false;  
    int rows=mat.length;  
    int columns=mat[0].length;  
    int orows=other.mat.length;  
    int ocolumns=other.mat[0].length;  
    if(rows!=orows || columns!=ocolumns)  
        return false;  
    for(int i=0 ; i<rows ; i++){  
        for(int j=0 ; j<columns ; j++){  
            if (!mat[ i ][ j ].equals(other.mat[ i ][ j ])) {  
                return false;  
            }  
        }  
    }  
    return true;  
}
```



```

public MatrixOfFraction sum(MatrixOfFraction other) {
    MatrixOfFraction temp=new MatrixOfFraction(this);
    if(other==null)
        return temp;
    int rows=mat.length;
    int columns=mat[0].length;
    int orows=other.mat.length;
    int ocolumns=other.mat[0].length;
    if(rows!=orows || columns!=ocolumns)
        return temp;
    for(int i=0 ; i<rows ; i++) {
        for(int j=0 ; j<columns ; j++) {
            temp.mat[i][j] = temp.mat[ i ][ j ].sum(other.mat[ i ][ j ]);
        }
    }
    return temp;
}

```

```
public String toString() {  
    int rows=mat.length;  
    int columns=mat[0].length;  
    String temp="";  
    for(int i=0 ; i<rows ; i++) {  
        for(int j=0 ; j<columns ; j++) {  
            if ( j==0 )  
                temp += mat[i][j];  
            else  
                temp += " " + mat[i][j];  
        }  
        temp += "\n";  
    }  
    return "Size:" + rows + "," + columns + "\n" + temp;  
}
```

```
public MatrixOfFraction multScalar(int num){  
    int rows=mat.length;  
    int columns=mat[0].length;  
    MatrixOfFraction temp=new MatrixOfFraction(this);  
    for(int i=0;i<rows;i++){  
        for(int j=0;j<columns;j++){  
            temp.mat[i][j]=temp.mat[i][j].multScalar(num);  
        }  
    }  
    return temp;  
}
```

```
public double[ ][ ] toDoubleMat(){  
    int rows=mat.length;  
    int columns=mat[0].length;  
    double[ ][ ] temp=new double[rows][columns];  
    for(int i=0;i<rows;i++){  
        for(int j=0;j<columns;j++){  
            temp[i][j]=(double)mat[i][j].getNumerator()/  
                mat[i][j].getDenominator();  
        }  
    }  
    return temp;  
}
```

```
public Fraction[ ][ ] toFractionMat(){  
    int rows=mat.length;  
    int columns=mat[0].length;  
    Fraction[ ][ ] temp=new Fraction[rows][columns];  
    for(int i=0;i<rows;i++){  
        for(int j=0;j<columns;j++){  
            temp[i][j]=new Fraction(mat[i][j]);  
        }  
    }  
    return temp;  
}
```



END