دانشکده مهندسی و علوم کامپیوتر

برنامه نویسی پیشرفته

وحیدی اصل

آرایه ها -۲

انگیزه

- در اسلاید اول آرایه ها، از آرایه های تک بعدی برای نمایش دنباله ای ترتیبی از داده ها استفاده کردیم.
- فرض کنید بخواهیم یک جدول یا ماتریس را نگهداری کنیم. در این مواقع استفاده از آرایه های تک بعدی مناسب نخواهد بود!
- جدول زیر فاصله میان شهرهای مختلف را برحسب مایل، نشان می دهد که با یک آرایه دوبعدی نگهداری می شود:

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

اعلان/ایجاد آرایه های دوبعدی

//اعلان یک ارایه دوبعدی

```
dataType[][] array_name;
```

//ایجاد یک آرایه دوبعدی در heap و انتساب آن به متغیر ارجاعی (ریموت کنترل) به نام array_name

```
array_name = new dataType[10][10];
```

// ترکیب اعلان و و ایجاد آرایه در یک دستور

```
dataType[][] array_name = new dataType[10][10];
```

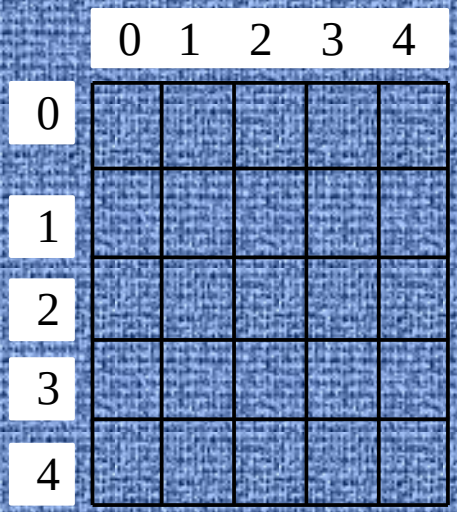
// روش جایگزین

```
dataType array_name[][] = new dataType[10][10];
```

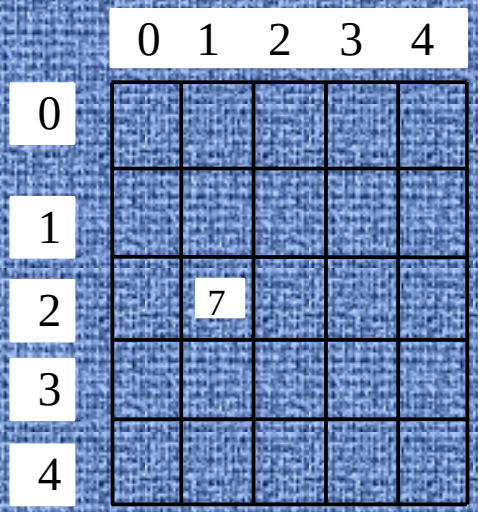
اعلان متغیرهایی از آرایه های دو بعدی و ایجاد آرایه های دو بعدی

```
int[][] matrix = new int[10][10];
 یا
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random()*1000);

double[][] x;
```

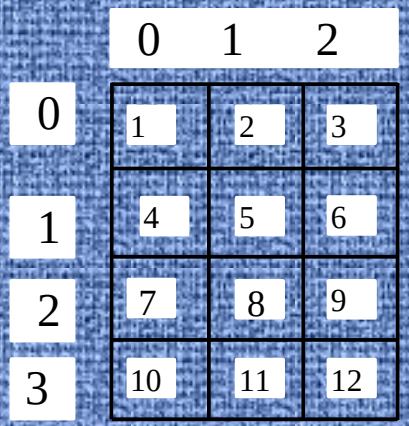
نمایی از آرایه دو بعدی




```
matrix = new int[5][5];
```




```
matrix[2][1] = 7;
```




```
int[][] array = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
};
```

matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

اعلان، ایجاد و مقداردهی اولیه یک آرایه دو بعدی می تواند به یکباره با استفاده از یک مقداردهنده (initializer) انجام شود.

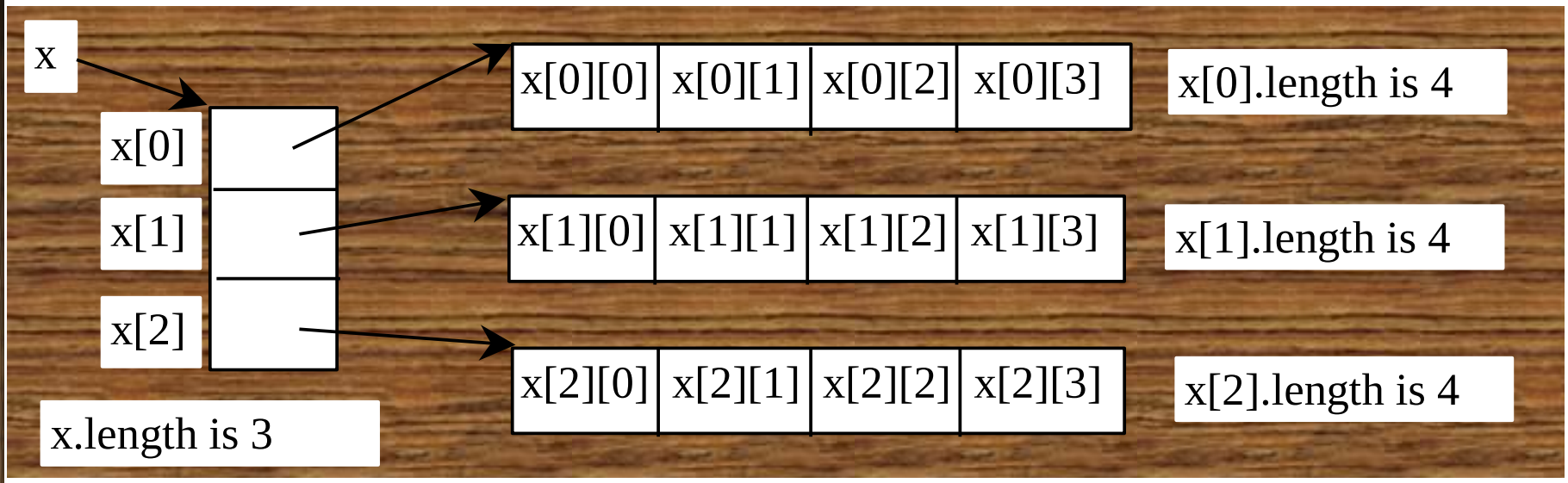
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

معادل با

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

طول آرایه های دو بعدی

```
int[][] x = new int[3][4];
```



طول آرایه های دو بعدی–ادامه

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

array.length

array[0].length

array[1].length

array[2].length

array[3].length

array[4].length ArrayIndexOutOfBoundsException

جمع دو ماتریس

```
class Testarray5{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,3,4},{3,4,5}};
int b[][]={{1,3,4},{3,4,5}};

//creating another matrix to store the sum of two matrices
int c[][]=new int[2][3];

//adding and printing addition of 2 matrices
for(int i=0;i<2;i++){
for(int j=0;j<3;j++){
c[i][j]=a[i][j]+b[i][j];
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}

}}
```

Test it Now

```
Output:2  6  8
       6  8  10
```

آرایه های نامنظم (Ragged arrays)

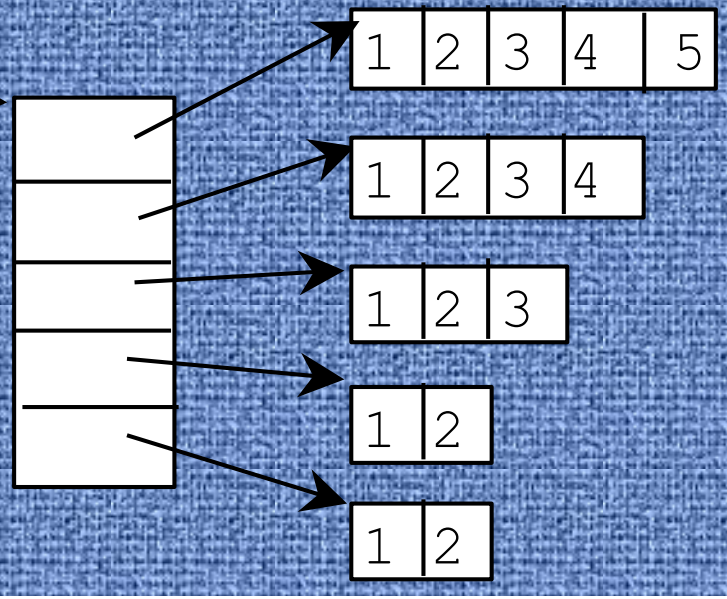
- هر سطر در یک آرایه دوبعدی، خود کنترل کننده (ریموت) یک آرایه است. در نتیجه، تعداد ستونهای هر سطر می تواند با تعداد ستونهای سطر دیگر متفاوت باشد. به چنین آرایه ای، اصطلاحاً نامنظم گفته می شود. برای مثال:

```
int[][] matrix = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

آرایه های نامنظم (Ragged arrays)–ادامه

```
int[][] triangleArray = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```



- در اسلایدهای بعدی مثالهایی از روشهای مختلف پردازش آرایه های چندبعدی ارایه شده است:

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Which row has the largest sum)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)

مقداردهی آرایه ها با داده های ورودی کاربر

```
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
  matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = input.nextInt();
  }
}
```

مقداردهی آرایه ها با داده های تصادفی

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = (int)(Math.random() * 100);
  }
}
```

چاپ آرایه ها

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    System.out.print(matrix[row][column] + " ");
  }

  System.out.println();
}
```

استفاده از Enhanced loop برای آرایه های دوبعدی

```
public class EnhancedLoop
 {
     public static void main(String[] args) {
     int Site[][] = new int[10][10];
      for (int[] i : Site)
       {
          for (int j : i){
             i[j] = 1;
             System.out.print(""+i[j]);
         }
       System.out.println();
    }
   }
 }
```

مجموع همه عناصر آرایه

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    total += matrix[row][column];
  }
}
```

محاسبه مجموع عناصر هر ستون

```
for (int column = 0; column < matrix[0].length; column++) {
  int total = 0;
  for (int row = 0; row < matrix.length; row++)
    total += matrix[row][column];
  System.out.println("Sum for column " + column +" is   “ +
        total);
}
```

بُرزدن تصادفی (شافلینگ)

```
for (int i = 0; i < matrix.length; i++) {
  for (int j = 0; j < matrix[i].length; j++) {
    int i1 = (int)(Math.random() * matrix.length);
    int j1 = (int)(Math.random() * matrix[i1].length);
    // Swap matrix[i][j] with matrix[i1][j1]
    int temp = matrix[i][j];
    matrix[i][j] = matrix[i1][j1];
    matrix[i1][j1] = temp;
  }
}
```

ارسال آرایه های دو بعدی به متدها

```
import java.util.Scanner;

public class PassTwoDimensionalArray {
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Enter array values
    int[][] m = new int[3][4];
    System.out.println("Enter " + m.length + " rows and “  + m[0].length + " columns: ");

    for (int i = 0; i < m.length; i++)
      for (int j = 0; j < m[i].length; j++)
        m[i][j] = input.nextInt();

    // Display result
    System.out.println("\nSum of all elements is " + sum(m));
  }
    public static int sum(int[][] m) {
    int total = 0;
    for (int row = 0; row < m.length; row++) {
      for (int column = 0; column < m[row].length; column++) {
        total += m[row][column];
      }
    }
    return total;
  }
}
```

نمره دادن به یک تست چند گزینه ای

- هدف: برنامه ای بنویسید که یک تست چند گزینه ای را تصحیح کند!

Students' Answers to the Questions:

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

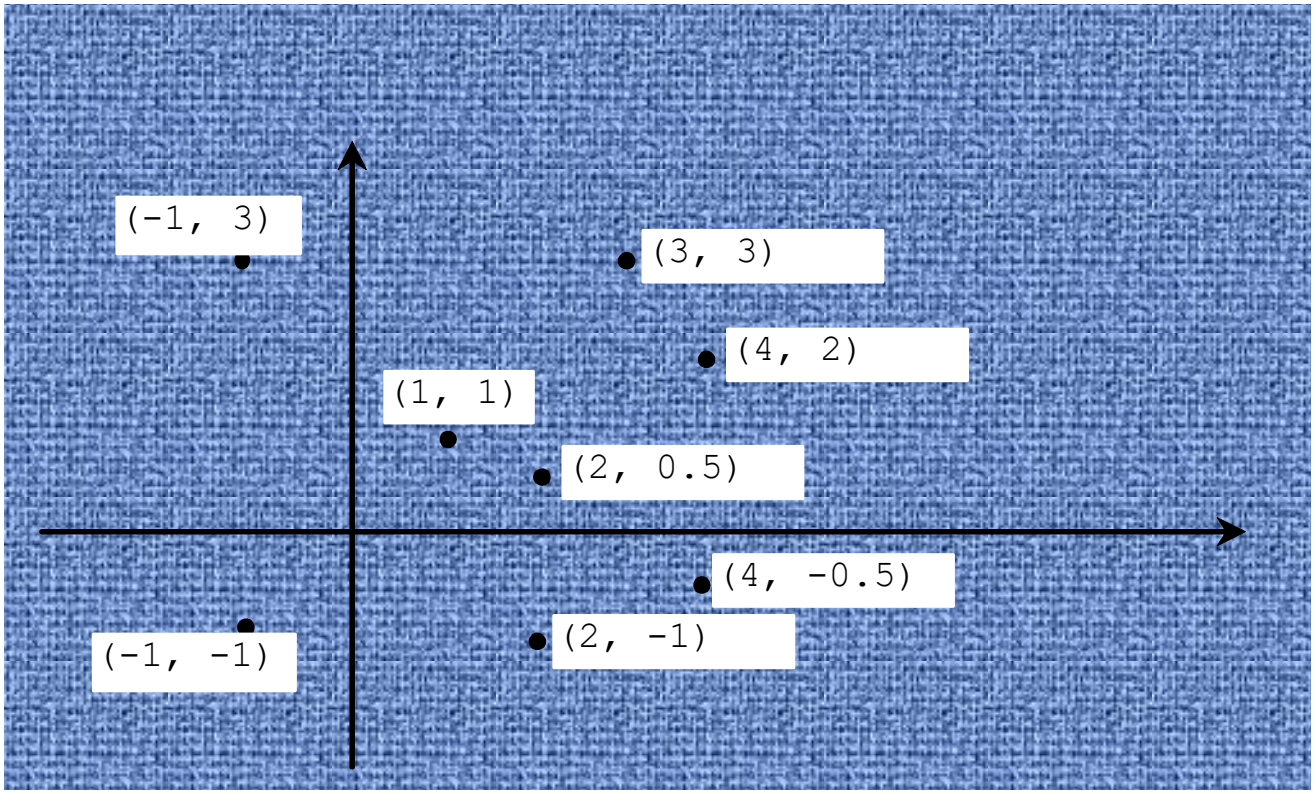
Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

نمره دادن به یک تست چند گزینه ای

```
public class GradeExam {
  /** Main method */
  public static void main(String args[]) {
    // Students' answers to the questions
    char[][] answers = {
      {'A', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'D', 'B', 'A', 'B', 'C', 'A', 'E', 'E', 'A', 'D'},
      {'E', 'D', 'D', 'A', 'C', 'B', 'E', 'E', 'A', 'D'},
      {'C', 'B', 'A', 'E', 'D', 'C', 'E', 'E', 'A', 'D'},
      {'A', 'B', 'D', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'B', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'B', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'E', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'}};

    // Key to the questions
    char[] keys = {'D', 'B', 'D', 'C', 'C', 'D', 'A', 'E', 'A', 'D'};

    // Grade all answers
    for (int i = 0; i < answers.length; i++) {
      // Grade one student
      int correctCount = 0;
      for (int j = 0; j < answers[i].length; j++) {
        if (answers[i][j] == keys[j])
          correctCount++;
      }
      System.out.println("Student " + i + "'s correct count is " +
        correctCount);
    }
  }
}
```

مسئله: یافتن نزدیکترین دو نقطه به یکدیگر



	x	y
0	-1	3
1	-1	-1
2	1	1
3	2	0.5
4	2	-1
5	3	3
6	4	2
7	4	-0.5

مسئله: یافتن نزدیکترین دو نقطه به یکدیگر

```
import java.util.Scanner;

public class FindNearestPoints {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter the number of points: ");
    int numberOfPoints = input.nextInt();

    // Create an array to store points
    double[][] points = new double[numberOfPoints][2];
    System.out.print("Enter " + numberOfPoints + " points: ");
    for (int i = 0; i < points.length; i++) {
      points[i][0] = input.nextDouble();
      points[i][1] = input.nextDouble();
    }

    // p1 and p2 are the indices in the points array
    int p1 = 0, p2 = 1; // Initial two points
    double shortestDistance = distance(points[p1][0],
        points[p1][1],
      points[p2][0], points[p2][1]); // Initialize shortestDistance
```

```
    // Compute distance for every two points
    for (int i = 0; i < points.length; i++) {
      for (int j = i + 1; j < points.length; j++) {
        double distance = distance(points[i][0], points[i][1],
          points[j][0], points[j][1]); // Find distance

        if (shortestDistance > distance) {
          p1 = i; // Update p1
          p2 = j; // Update p2
          shortestDistance = distance; // Update
shortestDistance
        }
      }
    }
    // Display result
    System.out.println("The closest two points are " +
      "(" + points[p1][0] + ", " + points[p1][1] + ") and (" +
      points[p2][0] + ", " + points[p2][1] + ")");
  }

  /** Compute the distance between two points (x1, y1)
and (x2, y2)*/
  public static double distance(
      double x1, double y1, double x2, double y2) {
    return Math.sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 -
y1));
  }
}
```

سودوکو (Sudoku) چیست؟

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

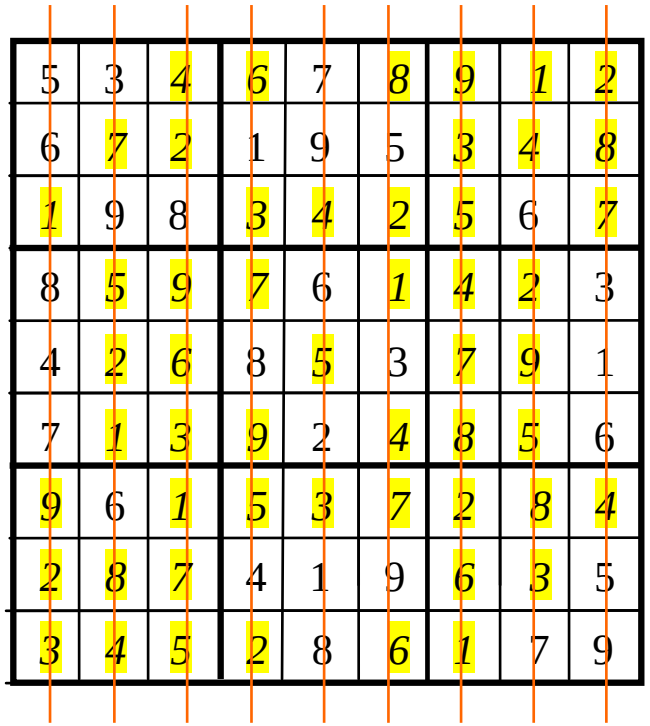
هر سطر شامل عناصری از ۱ تا ۹ است

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

هر ستون شامل عناصری از ۱ تا ۹ است

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9



هر جعبه ۳x۳ حاوی عناصری از ۱ تا ۹ است

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

بررسی صحیح بودن راه حل ارایه شده

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

آرایه های چندبعدی

- در برخی مواقع لازم است آرایه هایی با بیش از دو بعد تعریف کنیم.
- در جاوا مجازیم آرایه ای با هر تعداد بعد تعریف کنیم (آرایه n بعدی که n یک عدد صحیح باشد).
- روش تعریف آرایه چند بعدی مشابه آرایه های دو بعدی می باشد.
- برای مثال، آرایه ۳بعدی را به شکل زیر تعریف می کنیم:

  - `double[][][] scores = new double[10][5][2];`

مسئله: محاسبه مجموع امتیازها

- می خواهیم برنامه ای بنویسیم که مجموع امتیازات را برای دانشجویان یک کلاس محاسبه می کند.
- فرض کنید امتیازات در یک آرایه سه بعدی به نام scores ذخیره سازی شده اند و اولین اندیس در این آرایه، دانشجو را مشخص می کند، دومین اندیس یک امتحان را مشخص می کند و اندیس سوم یک بخش در امتحان را مشخص کند.
- فرض کنید ۷ دانشجو و ۵ امتحان داریم و هر امتحان دو بخش دارد: یک بخش چندگزینه ای و یک بخش تشریحی.
- بنابراین scores[i][j][0] بیانگر امتیاز امتیاز دانشجوی i در امتحان j در بخش چندگزینه ای می باشد.
- برنامه اسلاید بعدی امتیاز کل را برای هر دانشجو نمایش می دهد.

مسئله: محاسبه مجموع امتیازها

```
public class TotalScore {
  /** Main method */
  public static void main(String args[]) {
    double[][][] scores = {
      {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
      {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
      {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
      {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
      {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
      {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}},
      {{1.5, 29.5}, {6.4, 22.5}, {14, 30.5}, {10, 30.5}, {16, 6.0}}};

    // Calculate and display total score for each student
    for (int i = 0; i < scores.length; i++) {
      double totalScore = 0;
      for (int j = 0; j < scores[i].length; j++)
        for (int k = 0; k < scores[i][j].length; k++)
          totalScore += scores[i][j][k];

      System.out.println("Student " + i + "'s score is " +    totalScore);
    }
  }
}
```

متدهایی با لیست متغیر ارگومانها (varargs)

- ساختار varargs (لیست متغیر آرگومانها) به یک متد امکان می دهد، صفر یا چند آرگومان ورودی بپذیرد.
- در صورت عدم استفاده از varargs می‌توانیم از متد سربارگذاری‌شده استفاده کنیم یا یک آرایه را به عنوان پارامتر ارسال نماییم که به علت مشکلات نگهداری، روشی مناسب به حساب نمی‌آید.
- در مواقعی که ندانیم چه تعداد آرگومان را به یک متد ارسال کنیم، varargs روشی مناسب به حساب می‌آید.

مزیت varargs

- به متد سربارگذاری شده نیاز نداریم.
- در نتیجه میزان کدنویسی کم خواهد شد.
- قاعده نحوی:

– Varargs از سه نقطه پشت سر هم "..." پس از نوشتن نوع داده به صورت زیر، استفاده می‌کند:

```
return_type method_name(data_type... variableName){}
```

مثالی ساده از varargs در جاوا

```
class VarargsExample1{

 static void display(String... values){
  System.out.println("display method invoked ");
 }

 public static void main(String args[]){

 display();//zero argument
 display("my","name","is","varargs");//four arguments
 }
}
```

Test it Now

```
Output:display method invoked
       display method invoked
```

مثالی دیگر از varargs در جاوا

```
class VarargsExample2{

 static void display(String... values){
  System.out.println("display method invoked ");
  for(String s:values){
   System.out.println(s);
  }
 }

 public static void main(String args[]){

 display();//zero argument
 display("hello");//one argument
 display("my","name","is","varargs");//four arguments
 }
}
```

Test it Now

```
Output:display method invoked
       display method invoked
       hello
       display method invoked
       my
       name
       is
       varargs
```

قواعد varargs

- در هنگام استفاده از varargs، باید قواعد مربوط به آن رعایت شود؛ در غیراینصورت کد کامپایل نخواهد شد.
- قواعد استفاده:
- بیش از یک آرگومان متغیر (varargs) در متد وجود نداشته باشد.
- در صورت استفاده از چند آرگومان، آرگومان متغیر (varargs) باید آخرین آرگومان باشد.

خطای کامپایل در صورت استفاده نادرست از varargs

```
void method(String... a, int... b){}//Compile time error

void method(int... a, String b){}//Compile time error
```

مثالی از یک متد که varargs آخرین آرگومان آن باشد

```
class VarargsExample3{

 static void display(int num, String... values){
  System.out.println("number is "+num);
  for(String s:values){
   System.out.println(s);
  }
 }

 public static void main(String args[]){

 display(500,"hello");//one argument
 display(1000,"my","name","is","varargs");//four arguments
 }
}
```

Test it Now

```
Output:number is 500
       hello
       number is 1000
       my
       name
       is
       varargs
```

مثالی دیگر از varargs

```
public class VarargsTest
{
   // calculate average
   public static double average( double... numbers )
   {
      double total = 0.0; // initialize total

      // calculate total using the enhanced for statement
      for ( double d : numbers )
         total += d;

      return total / numbers.length;
   } // end method average

   public static void main( String[] args )
   {
      double d1 = 10.0;
      double d2 = 20.0;
      double d3 = 30.0;
      double d4 = 40.0;

      System.out.printf( "d1 = %.1f\nd2 = %.1f\nd3 = %.1f\nd4 = %.1f\n\n",
         d1, d2, d3, d4 );

      System.out.printf( "Average of d1 and d2 is %.1f\n",
         average( d1, d2 ) );
      System.out.printf( "Average of d1, d2 and d3 is %.1f\n",
         average( d1, d2, d3 ) );
      System.out.printf( "Average of d1, d2, d3 and d4 is %.1f\n",
         average( d1, d2, d3, d4 ) );
   } // end main
} // end class VarargsTest
```

```
d1 = 10.0
d2 = 20.0
d3 = 30.0
d4 = 40.0

Average of d1 and d2 is 15.0
Average of d1, d2 and d3 is 20.0
Average of d1, d2, d3 and d4 is 25.0
```

Command line arguments– main آرگومانهای متد

- با استفاده از خط فرمان می‌توانیم به برنامه جاوا آرگومان ورودی یدهیم.
- این‌کار با استفاده از یک آرایه از رشته‌های String[] درقالب لیست پارامترها در متد main امکانپذیر می‌شود.
- طبق قرارداد این آرایه args نام‌گذاری شده است.
- هرگاه برنامه جاوا را در command prompt اجرا می‌کنید، در صورت نیاز، آرگومانهای ورودی برنامه را بعد از نام کلاس (حاوی بایت کد) در خط فرمان قرار می‌دهید و ماشین مجازی جاوا این آرگومانها را به عنوان پارامترهای ورودی از جنس String به متد main کلاس شما می‌فرستد.
- تعداد آرگومانهای ورودی با استفاده از متد length آرایه محاسبه می‌شود.
- آرگومانهای ورودی، اغلب حاوی گزینه‌های انتخابیِ (options) کاربر و یا اسامی فایلها به برنامه‌های جاوا هستند.
- مثال زیر نشان می‌دهد چگونه برای کلاس InitArray سه آرگومان ورودی در خط فرمان نوشته می‌شود.
  - پس از کامپایل برنامه با javac و ایجاد فایل بایت کد به نام InitArray.class ، آرگومانهای ورودی در هنگام اجرای برنامه با فراخوانی java و پس از نام فایل حاوی بایت کد (فایل با پسوند کلاس) نوشته می‌شوند.

```
java InitArray 5 0 4
```

  - این دستور می‌گوید به ترتیب آرگومانهای ۵، ۰ و ۴ در args[0]، args[1] و args[2] قرار گیرد.

آرگومانهای متدmain –مثال

```
public class InitArray
{
```

```
java InitArray 5 0 4
Index   Value
    0       0
    1       4
    2       8
    3      12
    4      16
```

```
crement." );

   int[] array = new int[ arrayLength ];        // create array

    int initialValue = Integer.parseInt( args[ 1 ] );     // get initial value and increment from command-line arguments
   int increment = Integer.parseInt( args[ 2 ] );

   for ( int counter = 0; counter < array.length; counter++ ) // calculate value for each array element
      array[ counter ] = initialValue + increment * counter;

   System.out.printf( "%s%8s\n", "Index", "Value" );
         // display array index and value
   for ( int counter = 0; counter < array.length; counter++ )
      System.out.printf( "%5d%8d\n", counter, array[ counter ] );
  } // end else
 } // end main
} // end class InitArray
```

کلاس Arrays

- کلاس Arrays به شما امکان می‌دهد بدون اختراع دوباره چرخ(!!!) از متدهای استاتیک برای دستکاری برروی آرایه ها بهره ببرید.
- متدهای موجود در این کلاس، عناصر آرایه را مرتب‌سازی می کنند (مثلا به ترتیب صعودی)
- از جستجوی دودویی برای یافتن یک عنصر در آرایه استفاده می‌کنند.
- از متد equals برای مقایسه آرایه‌ها استفاده می‌کنند و قادرند در خانه‌های آرایه، مقدار مشخصی قرار دهند.
- این متدها برای آرایه‌هایی از انواع اصلی و انواع ارجاعی، سربارگذاری شده‌اند.

Arrays کلاس

```
import java.util.Arrays;

public class ArrayManipulations {
  public static void main( String[] args )  {
    // sort doubleArray into ascending order
    double[] doubleArray = { 8.4, 9.3, 0.2, 7.9, 3.4 };
    Arrays.sort( doubleArray );
    System.out.printf( "\ndoubleArray: " );
     for ( double value : doubleArray )
       System.out.printf( "%.1f ", value );

    // fill 10-element array with 7s
    int[] filledIntArray = new int[ 10 ];
    Arrays.fill( filledIntArray, 7 );
    displayArray( filledIntArray, "filledIntArray" );
     // copy array intArray into array intArrayCopy
    int[] intArray = { 1, 2, 3, 4, 5, 6 };
    int[] intArrayCopy = new int[ intArray.length ];
    System.arraycopy( intArray, 0, intArrayCopy, 0, intArray.length );
    displayArray( intArray, "intArray" );
    displayArray( intArrayCopy, "intArrayCopy" );
     // compare intArray and intArrayCopy for equality
    boolean b = Arrays.equals( intArray, intArrayCopy );
    System.out.printf( "\n\nintArray %s intArrayCopy\n",
      ( b ? "==" : "!=" ) );
     // compare intArray and filledIntArray for equality
    b = Arrays.equals( intArray, filledIntArray );
    System.out.printf( "intArray %s filledIntArray\n",
      ( b ? "==" : "!=" ) );
```

```
// search intArray for the value 5
    int location = Arrays.binarySearch( intArray, 5 );

    if ( location >= 0 )
      System.out.printf(
        "Found 5 at element %d in intArray\n", location );
    else
      System.out.println( "5 not found in intArray" );

    // search intArray for the value 8763
    location = Arrays.binarySearch( intArray, 8763 );

    if ( location >= 0 )
      System.out.printf(
        "Found 8763 at element %d in intArray\n", location );
    else
      System.out.println( "8763 not found in intArray" );
  } // end main
)
}
```

```
doubleArray: 0.2 3.4 7.9 8.4 9.3
filledIntArray: 7 7 7 7 7 7 7 7 7 7
intArray: 1 2 3 4 5 6
intArrayCopy: 1 2 3 4 5 6

intArray == intArrayCopy
intArray != filledIntArray
Found 5 at element 4 in intArray
8763 not found in intArray
```

ption

Collections و کلاس ArrayList

- API جاوا چند ساختمان‌داده از پیش‌تعریف‌شده به نام collections فراهم کرده که با استفاده از آنها گروههایی از اشیای مرتبط، ذخیره‌سازی می‌شوند.
- این کلاسها متدهای کارآمدی تعریف کرده‌اند که با استفاده از آنها می‌توانید داده‌ها را بدون نیاز به دانستن نحوه ذخیره‌سازی آنها، ذخیره و بازیابی کنید.
- به‌این‌ترتیب در میزان برنامه‌نویسی شما صرفه‌جویی خواهد شد.
- قبلاً برای ذخیره دنباله‌ای از اشیا از آرایه استفاده می‌کردیم.
- آرایه‌ها در زمان اجرا برای اضافه‌نمودن عناصر جدید، به طور خودکار اندازه خود را تغییر نمی‌دهند.
- کلاس collection به نام ArrayList<T> (از پکیج java.util) یک راه‌حل مناسب برای ذخیره‌سازی داده‌ها می‌باشد.
- کلاس ArrayList می‌تواند به طور پویا (دینامیک) اندازه خود را تغییر دهد تا عناصر بیشتری را در خود جای دهد.

- به T اصطلاحاً نگهدارنده مکان (placeholder) گفته می‌شود.
- هرگاه یک ArrayList جدید تعریف می‌کنید، به جای T نوع عناصری را که arrayList شما قرار است نگهداری کند، بنویسید.
- اینکار مشابه این است که در هنگام اعلان یک آرایه معمولی، نوع عناصر آن را تعیین کنیم.
- تفاوت مهم این است که کلاسهای collection (مانند arrayList) تنها انواع غیراصلی (reference) را نگهداری می‌کنند.

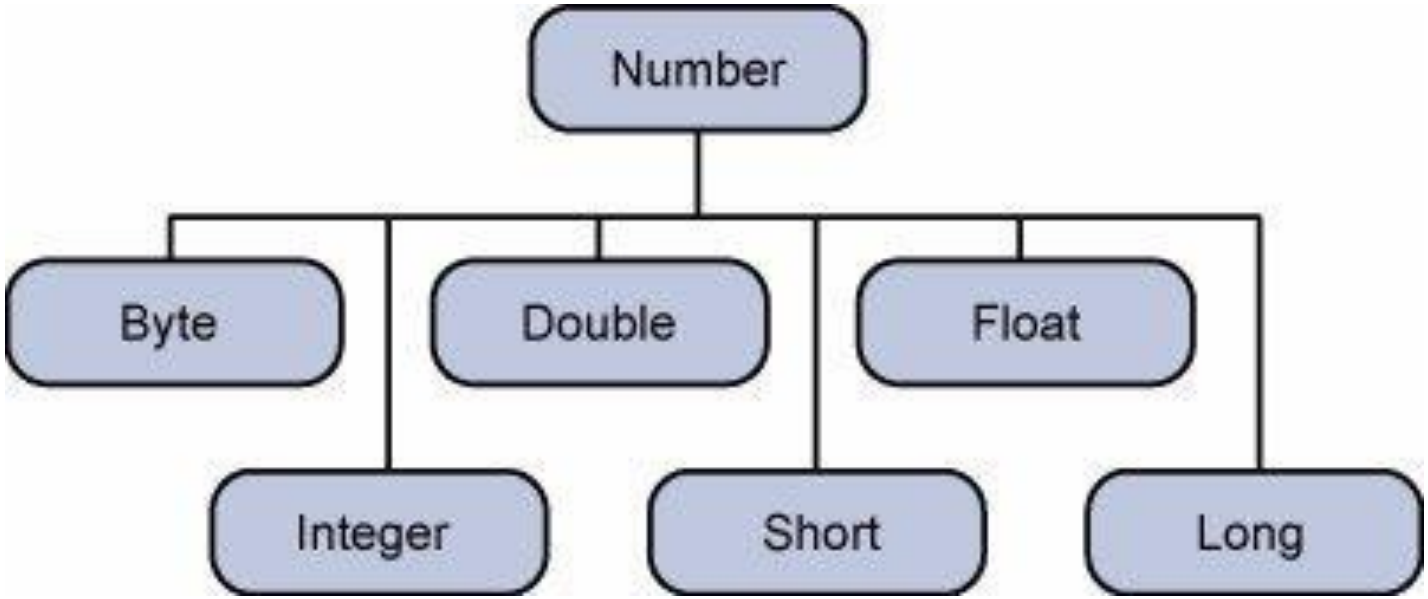
نحوه اعلان یک ArrayList

- در مثال زیر یک ArrayList اعلان شده که هر عنصر آن یک رشته می‌باشد.

```
ArrayList<String> al=new ArrayList<String>();//creating new generic arraylist
```

کلاس Number

- از آنجایی که یک ArrayList نمی‌تواند مقادیری از انواع اصلی (primitive type) را نگهداری کند، برای نگهداری این مقادیر از کلاسهای ویژه‌ای استفاده می‌کنیم که از یک کلاس انتزاعی (Abstract class) به نام Number ارث‌بری می‌کنند و قادرند مانند یک wrapper لفافه‌ای از جنس کلاس برروی مقدار نوع اصلی قرار دهند. برای مثال، شیئی از کلاس Integer قادر است مقداری از نوع int را در خود نگهداری کند.



مثالی از کلاس ArrayList جاوا

```
import java.util.*;
class TestCollection1{
 public static void main(String args[]){

  ArrayList<String> al=new ArrayList<String>();//creating arraylist
  al.add("Ravi");//adding object in arraylist
  al.add("Vijay");
  al.add("Ravi");
  al.add("Ajay");

  Iterator itr=al.iterator();//getting Iterator from arraylist to traverse elements
  while(itr.hasNext()){
   System.out.println(itr.next());
  }
 }
}
```

Test it Now

```
Ravi
Vijay
Ravi
Ajay
```

- در حالت کلی به دو شیوه می‌توان به عناصر کلاس Collection و به طور خاص کلاس ArrayList دسترسی داشت:
- به صورت تکراری با تعریف واسط Iterator (interface) و فراخوانی متد hasNext در آن مانند اسلاید قبلی
- با استفاده از ساختارحلقه for-each

مثال دسترسی به ArrayList باساختارحلقه for-each

```
import java.util.*;
class TestCollection2{
 public static void main(String args[]){
  ArrayList<String> al=new ArrayList<String>();
  al.add("Ravi");
  al.add("Vijay");
  al.add("Ravi");
  al.add("Ajay");
  for(String obj:al)
    System.out.println(obj);
 }
}
```

Test it Now

```
Ravi
Vijay
Ravi
Ajay
```

کارهایی که می توان با ArrayList انجام داد–۱

1. Make one

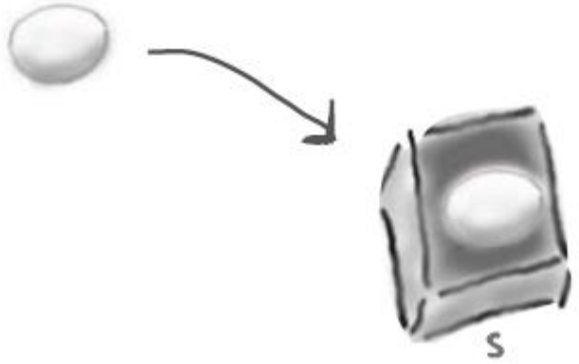
```
ArrayList<Egg> myList = new ArrayList<Egg>();
```

Don't worry about this new <Egg> angle-bracket syntax right now; it just means "make this a list of Egg objects".

A new ArrayList object is created on the heap. It's little because it's empty.

2. Put something in it

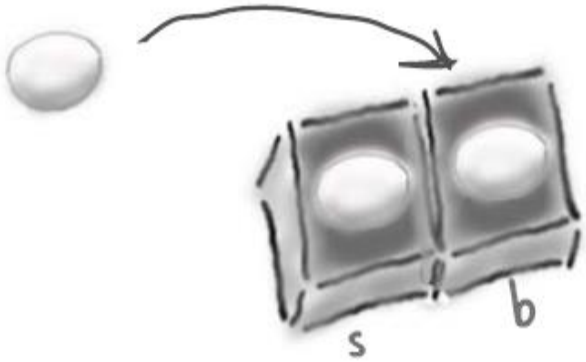
```
Egg s = new Egg();

myList.add(s);
```



Now the ArrayList grows a "box" to hold the Egg object.

3. Put another thing in it

```
Egg b = new Egg();

myList.add(b);
```



The ArrayList grows again to hold the second Egg object.

کارهایی که می توان با ArrayList انجام داد–۲

4 Find out how many things are in it

```
int theSize = myList.size();
```

The ArrayList is holding 2 objects so the size() method returns 2

5 Find out if it contains something

```
boolean isIn = myList.contains(s);
```

The ArrayList DOES contain the Egg object referenced by 's', so contains() returns true

6 Find out where something is (i.e. its index)

```
int idx = myList.indexOf(b);
```

ArrayList is zero-based (means first index is 0) and since the object referenced by 'b' was the second thing in the list, indexOf() returns 1

کارهایی که می توان با ArrayList انجام داد-۳

⑦ Find out if it's empty

```
boolean empty = myList.isEmpty();
```

it's definitely NOT empty, so isEmpty() returns false

⑧ Remove something from it

```
myList.remove(s);
```

Hey look — it shrank!

مثال ArrayList با عناصر تعریف شده توسط کاربر

```
class Student{
  int rollno;
  String name;
  int age;
  Student(int rollno,String name,int age){
   this.rollno=rollno;
   this.name=name;
   this.age=age;
  }
}
```

مثال ArrayList با عناصر تعریف شده توسط کاربر

```
import java.util.*;
public class TestCollection3{
 public static void main(String args[]){
  //Creating user-defined class objects
  Student s1=new Student(101,"Sonoo",23);
  Student s2=new Student(102,"Ravi",21);
  Student s2=new Student(103,"Hanumat",25);

  ArrayList<Student> al=new ArrayList<Student>();//creating arraylist
  al.add(s1);//adding Student class object
  al.add(s2);
  al.add(s3);

  Iterator itr=al.iterator();
  //traversing elements of ArrayList object
  while(itr.hasNext()){
    Student st=(Student)itr.next();
    System.out.println(st.rollno+" "+st.name+" "+st.age);
  }
 }
}
```

Test it Now

```
101 Sonoo 23
102 Ravi 21
103 Hanumat 25
```

مثال ArrayList با عناصر تعریف شده توسط کاربر

- توجه کنید که با اینکه اشیای موجود در ArrayList از کلاس Student هستند: چون ریموت کنترل itr از نوع Iterator تعریف شده که یک Interface است، نمی‌تواند به صفات ویژه شیئ Student دسترسی پیدا کند؛
- در نتیجه ابتدا یک downcasting انجام می‌دهیم تا ریموت کنترل بتواند به همه فیلدهای شیئ Student دسترسی پیدا کند. این downcasting به کلاس شیئ موجود در ArrayList همواره باید انجام شود.

```
import java.util.*;
public class TestCollection3{
 public static void main(String args[]){
  //Creating user-defined class objects
  Student s1=new Student(101,"Sonoo",23);
  Student s2=new Student(102,"Ravi",21);
  Student s2=new Student(103,"Hanumat",25);

  ArrayList<Student> al=new ArrayList<Student>();//creating arraylist
  al.add(s1);//adding Student class object
  al.add(s2);
  al.add(s3);

  Iterator itr=al.iterator();
  //traversing elements of ArrayList object
  while(itr.hasNext()){
    Student st=(Student)itr.next();
    System.out.println(st.rollno+" "+st.name+" "+st.age);
  }
 }
}
```

- نکته: قرار ندادن palceholder یا همان نوع generic در اعلان یک ArrayList به معنای استفاده ازArrayList غیرژنریک یا سنتی است. در این حالت، عناصر ArrayList از نوع کلاس Object می‌باشند.
- اگر عناصر یک ArrayList از نوع Object باشند، می‌توانند اشیای مختلفی در heap را کنترل کنند، اما ریموت کنترلهای ذخیره شده در ArrayList همگی از یک نوع Object بوده و برای دسترسی به صفتهای ویژه اشیای کنترل شده توسط آنها باید downcasting صورت گیرد.

addAll method مثالی از متد

```
import java.util.*;
class TestCollection4{
 public static void main(String args[]){

  ArrayList<String> al=new ArrayList<String>();
  al.add("Ravi");
  al.add("Vijay");
  al.add("Ajay");

  ArrayList<String> al2=new ArrayList<String>();
  al2.add("Sonoo");
  al2.add("Hanumat");

  al.addAll(al2);

  Iterator itr=al.iterator();
  while(itr.hasNext()){
   System.out.println(itr.next());
  }
 }
}
```

Test it Now

```
Ravi
Vijay
Ajay
Sonoo
Hanumat
```

مثالی از متد removeAll در ArrayList

```
import java.util.*;
class TestCollection5{
 public static void main(String args[]){

  ArrayList<String> al=new ArrayList<String>();
  al.add("Ravi");
  al.add("Vijay");
  al.add("Ajay");

  ArrayList<String> al2=new ArrayList<String>();
  al2.add("Ravi");
  al2.add("Hanumat");

  al.removeAll(al2);

  System.out.println("iterating the elements after removing the elements of al2...");
  Iterator itr=al.iterator();
  while(itr.hasNext()){
   System.out.println(itr.next());
  }

  }
}
```

Test it Now

```
iterating the elements after removing the elements of al2...
Vijay
Ajay
```

مثالی از متد retainAll در ArrayList

```
import java.util.*;
class TestCollection6{
 public static void main(String args[]){

  ArrayList<String> al=new ArrayList<String>();
  al.add("Ravi");
  al.add("Vijay");
  al.add("Ajay");

  ArrayList<String> al2=new ArrayList<String>();
  al2.add("Ravi");
  al2.add("Hanumat");

  al.retainAll(al2);

  System.out.println("iterating the elements after retaining the elements of al2...");
  Iterator itr=al.iterator();
  while(itr.hasNext()){
   System.out.println(itr.next());
  }
 }
}
```

Test it Now

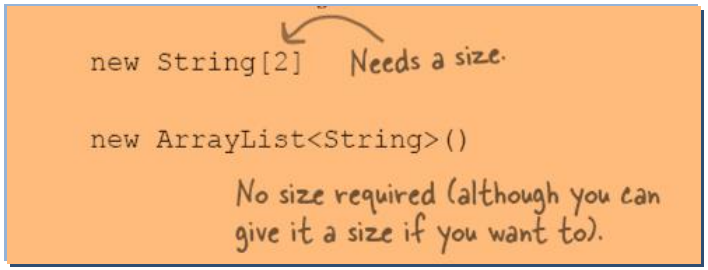
```
iterating the elements after retaining the elements of al2...
Ravi
```

جمعبندی-مجموعه متدهای موجود در ArrayList

java.util.ArrayList	
+ArrayList()	Creates an empty list.
+add(o: Object): void	Appends a new element o at the end of this list.
+add(index: int, o: Object): void	Adds a new element o at the specified index in this list.
+clear(): void	Removes all the elements from this list.
+contains(o: Object): boolean	Returns true if this list contains the element o.
+get(index: int): Object	Returns the element from this list at the specified index.
+indexOf(o: Object): int	Returns the index of the first matching element in this list.
+isEmpty(): boolean	Returns true if this list contains no elements.
+lastIndexOf(o: Object): int	Returns the index of the last matching element in this list.
+remove(o: Object): boolean	Removes the element o from this list.
+size(): int	Returns the number of elements in this list.
+remove(index: int): boolean	Removes the element at the specified index.
+set(index: int, o: Object): Object	Sets the element at the specified index.

مقایسه آرایه معمولی و ArrayList-۱

۱) در هنگام ایجاد یک آرایه معمولی باید اندازه آن را مشخص کنیم.
اما برای ArrayList شما هربار تنها یک شیئ از نوع ArrayList ایجاد می کنید. نیاز نیست که اندازه آن را مشخص کنید، چون: با اضافه شدن یا حذف اشیا از آن، بزرگ یا کوچک می‌شود!



مقایسه آرایه معمولی و ArrayList-۲

۲) برای قراردادن یک شیئ در یک آرایه معمولی، باید اندیسی از آرایه را که قرار است آن مقدار(یا شیئ) را نگهداری کند، مشخص نمایید.
اگر اندیسی که می‌خواهید مقدار(یا شیئ) در آن خانه قرار گیرند، خارج از محدوده تعریف شده آرایه باشد، با خطای زمان اجرا مواجه خواهید شد!
اما وقتی از ArrayList استفاده می‌کنیم، هم می‌توانید اندیس آرایه را با دستور add(anInt,anObject) مشخص کنید و یا اصولا اندیس را مانند دستور زیر مشخص نکنید:

myList.add(b); //no index specified

```
myList[1] = b;
```

Needs an index.