Lecture 20

ArrayList Vector 2D Array

In this lecture we will discuss list provided by java. We will discuss where to use array and where to use list provided by java. Later we will learn where to use 2D array.

Starting our discussion there is a big issue of array that is size. We cannot declare array unless we know size of array or at least have some idea about approximate size. Otherwise we will either waste memory or have to change array size at run-time which is tricky. Consider example suppose out of a list of voters you want to separate males and females. In case of array first you will count males and females than you can declare array according to their count and finally sort them in separate lists. A similar example is an image has seven colors and we want to store positions of each color of pixels in separate list. Without knowing count we can't use array and counting pixel of each color is costly as small image of 100 by 100 pixels has 10000 pixels in total. Java provides 2 classes to store lists namely ArrayList & Vector. Though there is small difference between them but for this stage there is no difference between them so students may use any of them. The big advantage of using these classes is that size is not programmer's responsibility. Programmer will just add and remove values and size will be managed by the java classes automatically. Disadvantage of using these classes is that user programmer has to use functions of these classes. Just consider following code:

```
import java.util.*;
class TestArrayList{
  public static void main(String a[]){
    ArrayList<Integer> marks=new ArrayList<Integer>();
    int i,m,size;
    size=(int)(Math.random()*10+1);
    System.out.println("Size:"+size);
    for(i=0;i<size;i++){
      m=(int)(Math.random()*50);
      marks.add(m);
    }
    for(i=0;i<marks.size();i++)
      System.out.print (marks.get(i)+" ");
    System.out.println();
  }
}
```

Here we are declaring marks as list (where a list can store multiple values of same type normally), we can declare this list without giving size. Though we have not given size but size will be the number of elements in list, we may call size function to check current size for the purpose of loop execution. Here we are using ***add***, ***get*** and ***size*** function of class ArrayList. Students should notice that running this code again and again that each time we have different size of list.

Lastly if you just replace ArrayList with Vector and run same code you will get same result. Therefore, student may use either ArrayList or Vector. Again though we may use ArrayList always an alternate of Array but it should not be. Whenever we have fixed size of data we should use Array because manipulation is really faster. Some more examples where we prefer to use ArrayList. Dividing data into k groups on any basis where each group may have any number of elements.

2D Array

Normally it is said that a 2D array is use to handle table type of data where table has rows and columns but specifically a 2D array is use to handle table type of data where table has same type of values. Like score of 10 players in 6 matches or marks of 30 students in 3 subjects or temperature of a month divided in weeks or Sale of an item in a year divided in month that is sale of 10th of March. A 2D array is row dominated. Though we normally have rectangular array that is equal number of columns in all rows, we may have different number of columns in different rows. However, for the time being we will concentrate on rectangular array only. We will start our discussion from declaration of 2D array. A 2D array can be declared in all the ways a 1D array can be declared. The difference is:

- 2 square brackets used to declare a 2D array
- 2 sizes are to be given, one for rows and other for columns
- when array values are to be initialized at the time of declaration an outer curly braces will contain set of curly braces having values with ,

Now see examples:

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

in this case x has 4 rows and 3 columns

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

a multi-line declaration same as we do in 1D array.

```
int x[][]={{3,2,5,8},{2,4,1,7}};
```

in this case x has 2 rows and 4 columns. Where column1 has values 3, 2, 5, 8 and column 2 has values 2, 4, 1, 7.

Now we will discuss handling of 2D array. Each element of array requires 2 indexes, one for row other for column. Varying first index we can move in row and varying second index we can move in column. Hence nested loop is required to handle a 2D array. For example consider following code to print array x declared above:

```
int i,j;
for(i=0;i<x.length;i++)
  for(j=0;j<x[i].length;j++)
    System.out.print (x[i][j] +" ");
```

We will get output: 3 2 5 8 2 4 1 7. To display values in table we have to move to next line in outer loop:

```
for(i=0;i<x.length;i++){
   for(j=0;j<x[i].length;j++)
      System.out.print (x[i][j] +" ");
   System.out.println();
}
```

Here we will get output:

```
3 2 5 8
2 4 1 7
```

See another code to add 10 in all values:

```
   int i,j;
   for(i=0;i<x.length;i++)
     for(j=0;j<x[i].length;j++)
       x[i][j] = x[i][j] + 10;
```

I want to show that to handle 2D array we need nested loop and to access array elements we need 2 indexes in each case.