Objective of Lecture 20

Arrays

- Introduce another Java container class called Array.
- Compare Arrays with Vectors
- See some examples with Arrays.

Course Content

- Introduction
- Objects
- Methods
- Tracing Programs
- Object State
- Sharing resources
- Selection
- Repetition
- Arrays
- Vectors
- Testing/Debugging
- Searching
- Files I/O
- Sorting
- Inheritance
- Recursion

Outline of Lecture 20

- Arrays
- Arrays versus Vectors
- Two simple array examples

Containers - Review

- An object’s state consists of instance variables that are bound to other objects or values.
- Sometimes it is useful for an object’s state to include an arbitrary number of other objects.
- An object that remembers an arbitrary number of other objects is called a container or a collection.

Vector: an Indexed Collection

- Vectors are containers whose elements are indexed by integers are called indexed containers.
- A Vector, can hold any kind of Objects, but not values.
- The integer indexes are the object references.
Vector: Condition of use

- A Vector is indexed by non-negative ints so it can be accessed by position;
- The first position is 0, not 1;
- A Vector knows its current size;
- A vector is initially empty and of size 0;
- The size of a vector is not known in advance.
- A Vector can be iterated by index;
- When you access an Object in a Vector, you must cast its type to use it.

Java Arrays - Declarations

- In Java there is a container called an array that can hold an arbitrary number of Objects or values.
- Since arrays can contain values, they can sometimes be used when Vectors cannot.
- An array is declared using brackets:
  ```java
  int markArray[];
  Person personArray[];
  String stringArray[];
  ```

Java Arrays - Constructors

- When an array is created, its size must be specified and the size cannot change.
- Since the size of an array is fixed when it is created, Vectors can sometimes be used when arrays cannot.
- An array is created using an array constructor:
  ```java
  markArray = new int[30];
  personArray = new Person[30];
  stringArray = new String[10];
  ```

Java Arrays - Accessing

- Array elements can be accessed using brackets: `markArray[3].`
- The length of an array can be obtained using the public length attribute (not a message): `markArray.length.`
- Since arrays are indexed, starting at zero, the indexes go from: 0 to length - 1.

```java
// add 5 to all elements of an array
for (index = 0; index < markArray.length; index++)
    markArray[index] = markArray[index] + 5;
```

Java Arrays - Literal Initializers

- Literal values can be put into an array using braces.
  ```java
  int markArray[] = {10, 20, 30, 40, 50};
  String stringArray[] = {"Fred", "Barney"};
  ```

Outline of Lecture 20

- Arrays
  - Arrays versus Vectors
  - Two simple array examples
Multidimensional Arrays

- We can have vectors inside a vector;
- Arrays can be multidimensional;

```
int numList[2][3];
```

```
int numbers[][] = {{1,2,3},{4,5,6}};
int oneNumber = numbers[1][2];
```

**Warning:** Do not try to access an element that does not exist (i.e., index beyond the size of the array or array dimension)

Vectors and Arrays

- An array is a list (multidimensional lists) of object/values of the same type, while a vector is a list of objects of any type;
- An array is a structure, not an object, and thus can be accessed relatively faster while an vector is an object accessible only by sending messages;
- Arrays are fixed sized and cannot grow while vectors are dynamic collections of objects.

Accessing Vectors vs. Accessing Arrays

- **Arrays:** reference the arrays at a given index;
  
  `myVariable = myArray[index];`

- **Vectors:** send a message `elementAt` to vector.
  
  `myObject = (objectClass)myVector.elementAt(index);`

Outline of Lecture 20

- Arrays
- Arrays versus Vectors
- Two simple array examples

Array Example

```
// Find the largest element in an array of ints
int markArray[] = {50, 37, 71, 99, 63};
int index;
int max;
index = 0;
max = markArray[index];
for (index = 1; index < markArray.length; index++)
  if (markArray[index] > max) max = markArray[index];
System.out.println(max);
```

Array Example2

```
// Find the index of the largest element in an array of ints
int markArray[] = {50, 37, 71, 99, 63};
int index;
int indexOfMax;
index = 0;
indexOfMax = 0;
for (index = 1; index < markArray.length; index++)
  if (markArray[index] > markArray[indexOfMax])
    indexOfMax = index;
System.out.println(indexOfMax);
```
Java Multidimensional Array Example

```java
// multidimensional array
int matrix[][] = {{0, 1, 2, 3}, {1, 0, 3, 2}, {2, 3, 0, 1}, {3, 2, 1, 0}};
int row, column;
for (row = 0; row < 4; row++) {
    for (column = 0; column < 4; column++)
        System.out.print(matrix[row][column] + " ");
    System.out.println();
}
```

```
0 1 2 3
1 0 3 2
2 3 0 1
3 2 1 0
```