Array, ArrayList, and Vector

2 – 4 pm Tuesday 7/29/2008 @JD2211

1

Agenda

- Review
  - Abstract classes and abstract methods
  - Overriding and overloading
  - Static method sand static variables
  - Final variables, final methods and final classes
- Array
- ArrayList
- Vector

2

Array

- A program language construct used to group and organized data
- In Java, array indexes always begin at zero
- In Java, an array is an object that must be instantiated

3

Array (cont'd)

- Two ways to declare an array reference in Java
  - `int[] grades;`
  - `int grades[];`
- Arrays must be instantiated
  - `int[] grades = new int[10];`
  - `int grades[] = new int[10];`

4

Array (cont'd)

- The length of the array is the number of elements it can hold, thus the maximum index of an array is `length` – 1
- `int[] number = new int[10];`
- Off-by-one error

5

Initializer Lists

- Instantiate an array and provide the initial values for the elements of the array
- The size of the array is determined by the number of items in the initializer list
- `int[] numbers = {1, 2, 3, 4, 5}`

Command-Line Arguments

- The formal parameter to the main method of a Java application is always an array of String objects
- The `String[]` parameter, which we call `args`, represents command-line arguments that are provided when the interpreter is invoked

7

Command-Line Arguments(cont'd)

- Any extra information on the command line when the interpreter is invoked is stored in the `args` array for use by the program
- This technique is another way to provide input to a program

8

Command-Line Arguments(cont'd)

```
public class NameTag{
   public static void main(String[] args){
      System.out.println();
      System.out.println("\t" + args[0]);
      System.out.println("My name is " + args[1]);
   }
}
```

9

Two-Dimensional Arrays

- Use two indexes to refer to a value in a two-dimensional array, one specifying the row, and another the column
- `int[][] table = new int[5][10];`

10

Exercise 1

- Find and correct the error of the following program segment:

```
final int ARRAY_SIZE = 5;
ARRAY_SIZE = 10;
```

11

Exercise 2

- Find and correct the error of the following program segment:

```
int b[] = new int[10];
for (int i = 0; I <= b.length; i++)
  b[i] = 1;
```

12

Exercise 3

- Find and correct the error of the following program segment:

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

13

The `ArrayList` Class

- An `ArrayList` object is similar to an array, but it **dynamically changes size** as needed, and element can be inserted and removed
- The `ArrayList` class is part of the `java.util` pacakge
- http://java.sun.com/j2se/1.4.2/docs/api/java/util/ArrayList.html

14

The `ArrayList` Class (cont'd)

- The ArrayList class is part of the Collection API
- Java Collection API
  - Set, List, Map, SortedSet, SortedMap, HashSet, TreeSet, ArrayList, LinkedList, Vector, Collections, Arrays, AbstractCollection
  - http://java.sun.com/j2se/1.4.2/docs/api/java/util/Collection.html

15

Some Methods of `ArrayList`

ArrayList ()
: Constructor: creates an initially empty list

boolean add (Object obj)
: Inserts the specified object to the end of this list

void add (int index, Object obj)
: inserts the specified object into this list at the specified index

void clear ()
: removes all elements from this list

Object remove (int index)
: removes the element at the specified index in this list and returns it.

Object get (int index)
: returns the object at the specified index in this list without removing it

Int indexOf (Object obj)
: returns the index of the first occurrence of the specified object

int size()
: returns the number of elements in this list

boolean isEmpty()
: returns true if this list contains no elements

16

```
import java.util.ArrayList;

public class Presidents{
   public static void main (String[] args){
      ArrayList presidents = new ArrayList();

      presidents.add("Jimmy Carter");
      presidents.add("Ronald Reagan");
      presidents.add("G.H. Bush");
      presidents.add("Bill Clinton");
      presidents.add("George Bush");

      System.out.println(presidents);

      int location = presidents.indexOf("Ronald Reagan");
      System.out.println(presidents.get(location));

      presidents.remove(location);

      System.out.println(presidents);
      System.out.println("At index 1: " + presidents.get(1));

      presidents.add(3, "Ronald Regan");
      System.out.println(presidents);

      System.out.println("Size of the presidents: " + presidents.size());
   }
}
```

17

The `Vector` Class

- `Vector` is similar to `ArrayList`
- The `Vector` class is part of the `java.util` pacakge
- The `Vector` class is part of the Collection API

18

Some Methods of `Vector`

ArrayList ()
: Constructor: creates an initially empty list

boolean add (Object obj)
: Inserts the specified object to the end of this list

void add (int index, Object obj)
: inserts the specified object into this list at the specified index

void clear ()
: removes all elements from this list

Object remove (int index)
: removes the element at the specified index in this list and returns it.

Object get (int index)
: returns the object at the specified index in this list without removing it

Int indexOf (Object obj)
: returns the index of the first occurrence of the specified object

int size()
: returns the number of elements in this list

boolean isEmpty()
: returns true if this list contains no elements

19

```
import java.util.Vector;

public class PresidentsVector{
   public static void main (String[] args){
      Vector presidents = new Vector();

      presidents.add("Jimmy Carter");
      presidents.add("Ronald Reagan");
      presidents.add("G.H. Bush");
      presidents.add("Bill Clinton");
      presidents.add("George Bush");

      System.out.println(presidents);

      int location = presidents.indexOf("Ronald Reagan");
      System.out.println(presidents.get(location));

      presidents.remove(location);

     System.out.println(presidents);
      System.out.println("At index 1: " + presidents.get(1));

      presidents.add(3, "Ronald Regan");
      System.out.println(presidents);

      System.out.println("Size of the presidents: " + presidents.size());
   }
}
```

20

`ArrayList` vs. `Vector`

- `ArrayLists` behave like unsynchronized `Vectors` and therefore execute faster than `Vectors`, because `ArrayLists` do not have the overhead thread synchronization.

21

Lab Assignment

- http://www.csun.edu/~twang/Java/Lab/7-29-Lab.pdf

22