Lesson 9:
Array Lists and JOptionPane

February 14, 2012
Array Lists and JOptionPane

- The for-each loop
- The Array List
- The JOptionPane for simple input-output
The for-each Loop

- Suppose we had the following code that would print the contents of the array `myArray`:
  ```java
double [] myArray = new double[50];
  // Code that would fill the array
  ...
  for (int i=0; i<myArray.length; i++)
    System.out.println(myArray[i]);
```

- Using the for-each construct, we can rewrite the above code as:
  ```java
double [] myArray = new double[50];
  // Code that would fill the array
  ...
  for (double element : myArray)
    System.out.println(element);
```
The for-each Loop

- Is an enhanced version of for loop with following syntax:

  ```java
  for (variable : collection) statement
  ```

- The for-each construct allows you to loop through each element in an array, as well as other things, without having to worry about the index values.

- Less error prone than for statement for iterating through contents of array:
  - don’t have to worry about beginning and ending index values
  - can’t cause “array index out of bounds” error
**ArrayList**

- Once array created, size is fixed. Can’t be changed.
  
  Ex: `int size = 10;`  
  `Circle circleArray[] = new Circle[size];`

- **ArrayList** class provides structure that can grow and shrink to meet storage requirements. Automatically adjusts capacity as you add and remove elements.

- Syntax for constructing ArrayList containing Circle objects:
  
  ```java
  ArrayList<Circle> circleArrayList = new ArrayList<Circle>();
  ```
I could add a Circle to the circleArrayList by

```java
Circle c = new Circle(5.0); // Circle of radius 5.0

circleArrayList.add(c);
```

- The `add` method will append the new element to the end of the array list.
- Each time a Circle is added to the end of the array list, the JVM will find memory that will hold the elements currently in `circleArrayList` plus the element being added.
(ArrayList Methods

- In addition to **add** method, the **ArrayList class** also provides a **remove()** method to remove an element from the **ArrayList** and automatically move other methods up.

- The **ArrayList size()** method will return the number of elements stored in the array list.

- Unfortunately, you cannot use the normal array syntax of [ ] to access an individual element. But you can do the following.

```java
Circle temp = circleArrayList.get(i) ;
```

and you will have the equivalent of

```java
Circle temp = circleArray[i] ;
```
ArrayList Methods (cont.)

- Optionally, you can specify the size of the array list when creating it:
  
  ```java
  ArrayList<Circle> circleArrayList = new ArrayList<Circle>(50);
  ```

- Then, once you’ve added all of the elements needed, method `trimToSize()` reduces the storage capacity of the array list to its current size.

- See java API for details of additional ArrayList methods.
FileReader Example

_last lesson we demonstrated a program that read file students.txt, breaking each line of input into individual tokens (studentID, last name, first name, major) and displaying the contents to the console (see next slide).

<table>
<thead>
<tr>
<th>students.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAJ70066 AGUILERA ALEJANDRO INFO MANAGEMENT</td>
</tr>
<tr>
<td>DBA80270 ARONHALT DRAKE COMPUTER SCIENCE</td>
</tr>
<tr>
<td>EMB60390 BARNES ELIZABETH INFO MANAGEMENT</td>
</tr>
<tr>
<td>MAB50654 BOSWORTH MICHAEL COMPUTER SCIENCE</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Next step

- Create a class called Student
- Instantiate a Student object for each line of data and store it in an ArrayList.
import java.util.*;
public class InputDemoScanner {

    public InputDemoScanner() {
    }

    public void readFile() {
        String studentID, first, last, major;
        try {
            File inFile = new File("students.txt");
            Scanner myScanner = new Scanner(inFile);
            while (myScanner.hasNext()) {
                studentID = myScanner.next();
                last = myScanner.next();
                first = myScanner.next();
                major = myScanner.nextLine();
                major = major.trim(); // Get rid of the \n
                System.out.println(first + ' ' + last + " ID#: " + studentID);
            }
            catch (FileNotFoundException ex) {
                System.out.println("File not found");
            }
            catch (IOException ex) {
                System.out.println("Unable to read from file");
            }
        }
    }

    public static void main(String[] args) {
        InputDemoScanner id = new InputDemoScanner();
        id.readFile();
    }
}
public class Student {
    private String studentID, last, first, major;

    public Student(String id, String l_name, String f_name, String maj) {
        studentID = id;
        last = l_name;
        first = f_name;
        major = maj;
    }

    public String getStudentID() {
        return studentID;
    }

    public String getLast() {
        return last;
    }

    public String getFirst() {
        return first;
    }

    public String getMajor() {
        return major;
    }

    public void setMajor(String major) {
        this.major = major;
    }

    public String toString() {
        return first + " " + last + " " + major;
    }
}
import java.io.*;
import java.util.*;

public class InputDemoScanner_ArrayList {

    ArrayList<Student> studentBody = new ArrayList<Student>();

    public InputDemoScanner_ArrayList() {} 

    public void readFile() {
        String studentID, first, last, major; try {
            File inFile = new File("students.txt");
            Scanner myScanner = new Scanner(inFile);
            while (myScanner.hasNext()) {
                studentID = myScanner.next();
                last = myScanner.next();
                first = myScanner.next();
                major = myScanner.nextLine();
                major = major.trim(); // Get rid of the \n
                Student m = new Student(studentID, last, first, major);
                studentBody.add(m);
            }
        }
    }

    public void displayByMajor(String major) {
        for(Student m: studentBody) {
            if (m.getMajor().equalsIgnoreCase(major))
                System.out.println(m);
        }
    }

    public static void main(String[] args) {
        InputDemoScanner_ArrayList id = new InputDemoScanner_ArrayList();
        id.readFile();
        id.displayByMajor("CompSci");
    }
}
JOptionPane - showInputDialog

- Create message or dialog boxes without overhead of full GUI application
- Get input from the user with an input dialog box

```java
String name = JOptionPane.showInputDialog("What's your name?");
```

- Part of the javax.swing package
Getting numeric input…must still use an input dialog box

```
String input = JOptionPane.showMessageDialog("Guess a number from 0 to 100");
```

Convert String input to numeric using wrapper classes

```
int number = Integer.parseInt(input);
```
JOptionPane- showMessageDialog

Display a message with a message dialog box

JOptionPane.showMessageDialog(null, "Welcome to CSCI3381!");
Getting a confirmation (yes/no) with a confirmation dialog box

```java
int done = JOptionPane.showConfirmDialog(null, "Are you done?");
```

Returns an int equal to one of three static constants:

- `JOptionPane.YES_OPTION`
- `JOptionPane.NO_OPTION`
- `JOptionPane.CANCEL_OPTION`
JOptionPane Example
(Number Game)

- Guess a number from 0 to 100
  - Version using console input (Scanner)
  - JOptionPane version
In-Class Practice

- See link on today’s lesson outline for a sample problem for you to try using File I/O and ArrayLists.
Summary

- The for-each loop
- The Array List
- The JOptionPane for simple input-output