COMP1409: Introduction to Software Development I

Mike Mulder (mmulder10@bcit.ca)

Week 10

Agenda

- Week 9 Review
- Quiz
  - Quiz 9
  - Review Answers
- Logistics
- Lesson 10
  - ArrayList
  - For Each Loops
- Course Evaluation
- Lab 10
  - Due Thursday

Review

1. How do you declare a 2D array variable?

```
int[][] a;  // local variable
private String[][] b; // instance variable
```

2. How do you initialize a 2D array?

```
a = new int[3][4];
b = new String[5][5];
private String[][] b = new String[5][5]; // Declare and Initialize
```

3. How do you use a 2D array?

```
b[2][3] = “Hello World”;
System.out.println(b[2][3]);
a[0][0] = 5; a[0][1] = 3; a[0][2] = 2; a[0][3] = 4;
int sum = a[0][0] + a[0][1] + a[0][2] + a[0][3];
```

Review

How can we visualize the 2D array?

With a table (or matrix)

Can we have different length rows in 2D arrays?

Yes. Example:

```
int [][] numbers;
numbers = new int[3][];    // defines three int arrays

numbers[0] = new int[4];   // first row holds 4 ints
numbers[1] = new int[500]; // second row holds 500 ints
numbers[2] = new int [7];  // third row holds 7 ints
```

Review

How do we get the number of rows in a 2D array?

```
int [][] numbers;
numbers = new int[3][4];
int rows = numbers.length;
```

How do we get the length of a row for a 2D array?

```
int columnsRow1 = numbers[0].length;
int columnsRow2 = numbers[1].length;
...
```

Review

- while loop

```
int i = 0;

while(i < names.length){
        System.out.println(names[i]);
        i++;
}
System.out.println(i); // ok
```

- for loop

```
for(int i = 0; i < names.length; i++){
        System.out.println(names[i]);
}
System.out.println(i); // error; undefined
// Local variable i only exists within the
// for loop
```

```
int i = 0;  // What is this called?

i < names.length; // What is this called?

i++; // What is this called?
```

Initialization

Condition

Update

Review

What is this called?

```
public class Demo{
    private int[][] numbers = {{1,2,3},{4,5,6}};
    public Demo(){
        for(int row = 0;row<numbers.length;row++){
            // loop through each row
            for(int column = 0;column < numbers[row].length;column++){
                // loop through columns of current row
                System.out.print(numbers[row][column]+" ");
             }
             System.out.println();
         }
    }
}
```

Nested For Loops

Questions

When would we use 2D arrays?

- Large data sets of fixed size, i.e. csv comma separated values) files
- Mathematics – such as matrices

What is a limitation of 2D arrays?

- The size has to be known upfront (i.e., when we initialize and before we use it).
- Same type for all elements

Quiz 9

Closed book, laptop, phone, etc.

You have a maximum of 20 minutes to complete

Raise your hand when you are done, and I will retrieve your paper

We will review the answers afterwards

Logistics – Remaining Classes

Week	Date	Topics	Comments
2Course Evaluation: To be conducted online during Session 11 prior to the lab.			
12	Nov. 24	ArrayList class Enhanced for (foreach) loop	Quiz 9
13	Dec. 1	Iterators	Quiz 10
14	Dec. 8	Final Exam	Assignment 3 Due

Assignment 2

Mark out of 106

Based off of the Test Runner

Almost everyone received 106/106

Assignment 3

Due Dec. 5th at midnight

Two classes: Package and Warehouse

Same format – unit tests and test runner

Everyone should be able to get 100% if they pass the unit tests.

Assignment 3

Hints on Warehouse:

- Takes in an array of Packages to the constructor (i.e., Package[])
- Return value is an array for some of the public methods (i.e., Package[])
- Internally an ArrayList should be used to store the Packages in the Warehouse
- The “ship” methods return the Package or Packages shipped (i.e., deleted) from the Warehouse

Final Exam Format

Total 3 hours

- First 90 minutes – Written/Closed Book (i.e., big quiz)
- Second 90 minutes – Coding/Open Book (i.e., small assignment with test)

Next week we will do a review of relevant topics and I will provide you will some study guidance.

Final Exam Format

Total 3 hours

- First 90 minutes – Written/Closed Book (i.e., big quiz)
- Second 90 minutes – Coding/Open Book (i.e., small assignment)

Collections

ArrayList

Autoboxing

For-Each Loop

Collections

- Java collections are sets, lists, and maps
- A Collection is a container that groups similar elements
- Examples would include a list of bank accounts, a set of students, a group of telephone numbers
- Collections are used to store groups of related objects in memory

Collections

- Collection classes provide efficient methods to organize, store and retrieve data

- One type of Collection is ArrayList

ArrayList

array

- Not a java Collection
- Fixed size
- Primitives or references
- All languages
- .length

ArrayList

- Java Collection
- Size can change
- Only stores references
- Only Java
- .size()

Some Advantages of ArrayList:

- Do not need to know size upfront
- The list is ordered (same order as inserted in the list)
- No gaps in the list when elements are removed (i.e., null checks not required)
- Collections provides utilities, such as sort

Wrapper Classes

- Remember the primitive types: int, boolean, double, char, …
- Java provides "wrapper classes" that hold an a primitive type. Examples:
  - Integer – for int
  - Boolean – for boolean
  - Double – for double
  - Character – for char
- This provides useful methods (i.e., type conversions) and allows you to *treat the primitive types as objects*

Ref: https://www.w3resource.com/java-tutorial/java-wrapper-classes.php

Autoboxing

Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on.

If the conversion goes the other way, this is called unboxing.

Ref: https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html

Autoboxing

- An ArrayList is a *list,* not an *array.* It works only with objects, not primitives.
  - Autoboxing can make it appear that ArrayLists can hold primitives
  - ArrayList<Integer> myArrayList;
  - myArrayList.add(5);
  - Actually becomes
  - myArrayList.add(new Integer(5)); automatically

Autoboxing

- ArrayList<Integer> myArrayList;
- myArrayList.add(5);
- Actually becomes
- myArrayList.add(new Integer(5)); automatically

- The int 5 was promoted to an Integer object
- Integer is a wrapper class for int

- int x = myArrayList.get(0); // unboxed back to int
- Integer x = myArrayList.get(0); // still an Integer object

Using Arraylist

- In order to make use of the ArrayList class, it should first be imported from the java.util package

```
import java.util.ArrayList;
public class ClassRoom
{
```

- The ArrayList is declared and created in the same way as an object of any class, except that you should specify the generic (or "parameterized") type as follows

```
private ArrayList<Student> students;

public ClassRoom()
{
    students = new ArrayList<Student>();
}
```

Using Arraylist

```
private ArrayList<Student> students;

public ClassRoom()
{
    students = new ArrayList<Student>();
}
```

- students can store ONLY references to Student objects now

Using ArrayList

- Every type that uses “<T>” notation is called a parametrized type or generic type
- This means that every time you use that type, you can specify what kind of object references it holds inside the “< >” notation
- Keep in mind only object reference types can be used in generic classes (not primitive types)

Using ArrayList

- Each element in the ArrayList is accessible by a zero-based index
- The ArrayList class provides a selection of powerful methods that can do many things
- Some of the methods that we will work with are:
  - remove(int index)
  - add(Element e)
  - get(int index)
  - size()

  - “RAGS”

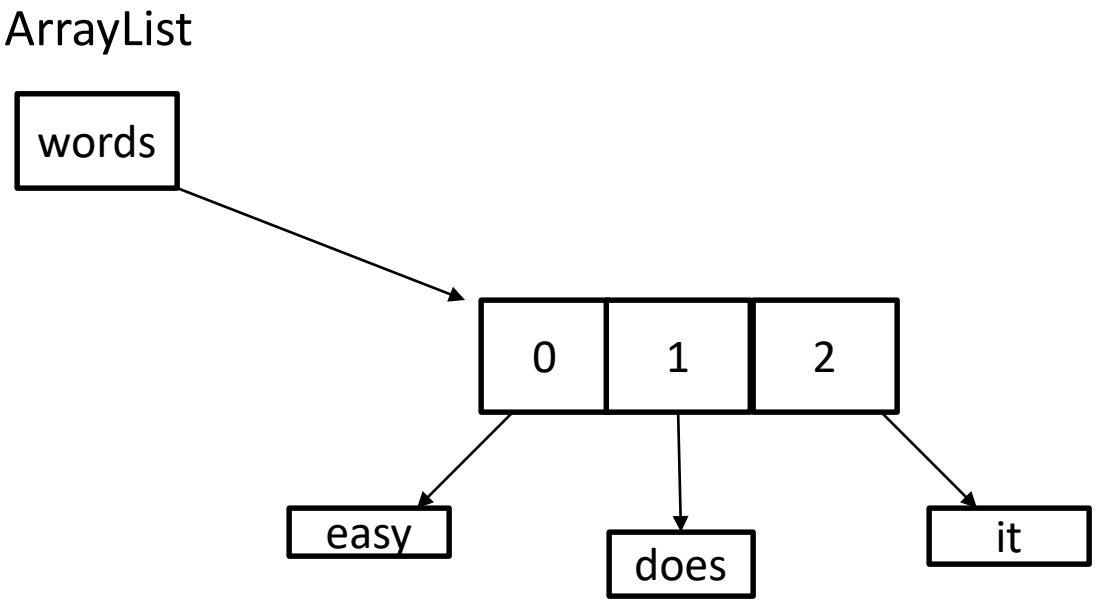
Using ArrayList

- The add() method adds elements to the ArrayList:

```
// make a new arraylist (initially empty)
ArrayList words = new ArrayList<String>();
// add three String objects to the ArrayList
words.add("easy");
words.add("does");
words.add("it");
```

Using ArrayList

- After the code on the previous slide is executed, the ArrayList *words* will have three references, each to a String object:



Using ArrayList

- Method remove() can take an int as a parameter.
- This int parameter represents the index of the element to remove
- Make sure that the passed index is a valid one before using it

```
if(index >= 0 && index < words.size()){
    words.remove(index);
 }
```

Using ArrayList

- Method get() takes an int parameter indicating which element we want, and returns a reference to that element from the ArrayList
- The valid index numbers that correspond to actual references in the ArrayList are always in the range from zero to size()-1

Using ArrayList

```
public void getAWord(int index){
    if(index >= 0 && index < words.size()){
        String theWord = words.get(index);
        System.out.println(theWord);
     }
 }
```

Using ArrayList

- Method size() takes no parameters and returns the current number of elements in the ArrayList.

```
System.out.println(words.size());
```

- The output of the above statement is : 3

Example

```
import java.util.ArrayList;

class BookStore
{
   private ArrayList<Book> books;

   public BookStore(){
      books = new ArrayList<>();

      books.add(new Book("harry potter", 1995));
      books.add(new Book("lord of the rings", 1954));

      books.remove(0);
      System.out.println(books.get(0).getTitle()); // lord of the rings

      System.out.println(books.size()); // 1, now
   }
}
```

Declare, Initialize, Use

Declare

- ArrayList<Book> books;
- private ArrayList<Book> books;

Initialize

- books = new ArrayList<Book>();
- books = new ArrayList<>(); // Type is optional

Use

- books.add(myBook); // Add a new book
- books.get(0);   // Get book at element 0
- books.size(); // Get the number of books
- books.remove(0); // Remove book at element 0 from list

Conversions (Advanced)

Adding an ArrayList to an Array:

```
Book[] bookArr = books.toArray(
                    new Book[books.size()]);
```

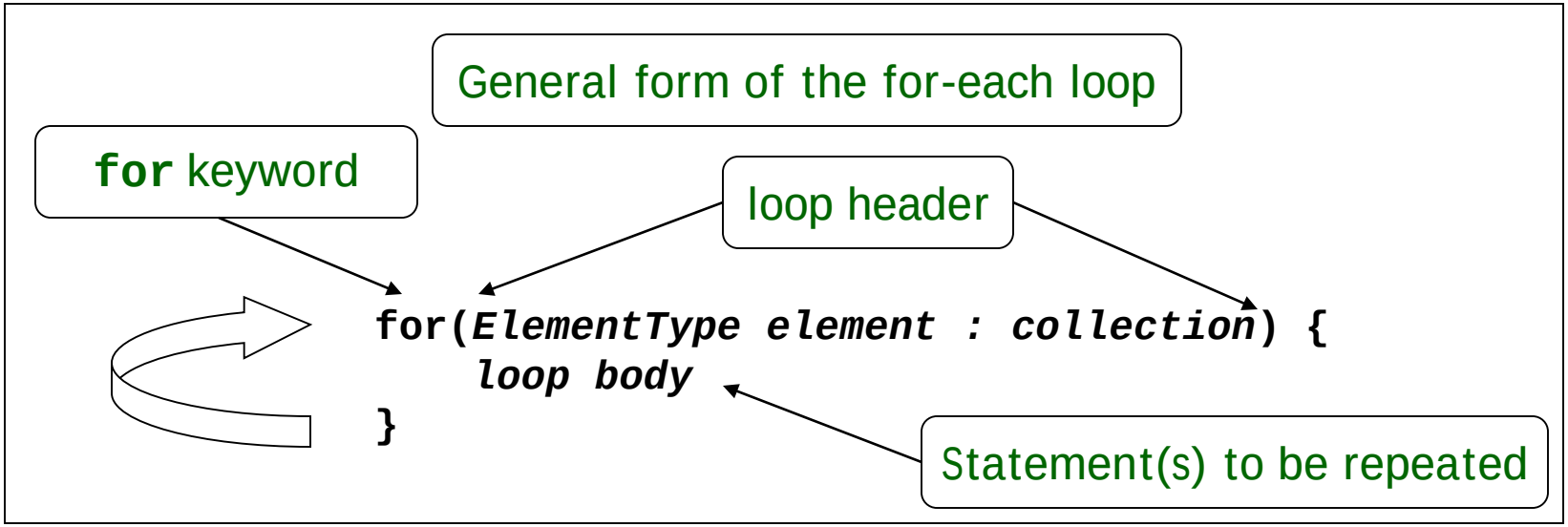
Converting an Array to an ArrayList:

```
ArrayList<Book> books =
      new ArrayList<Book>(
              Arrays.asList(bookArr));
```

The for-each loop

- With collections, we often want to repeat things once for every element in the collection
- Java has different kind of loop called a "*for-each*" loop or "*enhanced for*" loop
- It doesn't use the index directly
- This kind of loop has been designed to cycle through all the elements in collections such the ArrayList

for-each loop pseudo code



Pseudo-code expression of the actions of a for-each loop

For each *element* in *collection*, do the things in the *loop body*.

for-each loop Example

```
for(String aWord: words){
    System.out.println(aWord);
}
```

- To print each word in the ArrayList
- NOTE: if words == null, this would crash

Question

What are some advantages of the ArrayList of an array?

- Do not need to know size upfront
- The list is ordered (same order as inserted in the list)
- No gaps in the list when elements are removed (i.e., null checks not required)
- Collections provides utilities, such as sort

Question

What is Autoboxing?

Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on.

Question

What methods does an ArrayList provide?

- `remove(int index)`
- `add(Element e)`
- `get(int index)`
- `size()`

Question

What's an advantage of the for each loop?

You get the object directly as a variable within your loop – do not have to access through an index on an array.

References

Wrapper Classes:
https://www.w3resource.com/java-tutorial/java-wrapper-classes.php

Autoboxing:
https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html

ArrayList and For Each Loop:
https://beginnersbook.com/2013/12/java-arraylist/

Lab 10

Refactor of Lab 8 (ProvinceTerritory and Canada).

- Code is available in Week 7 course content.
- Rewrite to use an ArrayList rather than an Array

Next Week

Topics:

- Quiz 10
- Class Topic: Iterators
  - Lab – Refactor of Canada class to use Iterators
  - Let's try to get done in-class as a group
- Course Review and Final Exam Hints
  - Practice questions