Announcements

- Program 2 Due 9/20 at 11:59PM
- Midterm 1 Date: 10/2
- TA Office hours still tbd… (there are TA's in the lab but I am not sure when)
  - TA Room: GMCS 425
- Sample code 2 is available

Last Week’s Quiz

Question

Don’t worry it’s not worth points...

Primitive Types vs Reference Types

- A **primitive type** variable directly stores the data for that variable type, such as int, double, or char. Ex: int numStudents = 20; declares an int that directly stores the data 20.
- A **reference type** variable can refer to an instance of a class, also known as an object, via the object's memory address

Pass by Value

- When assigning a variable to another variable, the value at the memory location of the first variable, gets **copied** to the memory location of the second variable
- This also occurs when passing a variable as an input to a method
- What happens when we pass an object as an input to a method? Does it get copied into the method parameter?

Lets see an example...

Question

Don’t worry it’s not worth points...

Look at this pass by value example, what is the result?

‘this’ implicit parameter

- In java, all non-static methods have an **implicit parameter** that is a reference to the member methods own object.
- The implicitly-passed object reference is accessible via the keyword: **this**
- A class member can be accessed as: **this.classmember**
- One of the primary uses of the implicit parameter, is to assign a member field to a method parameter of the same name (identifier)

Let’s look at an example...

Wrapper Classes

Turn primitive types into reference types (an object)

- int -> Integer
- char -> Character
- byte -> Byte
- short -> Short
- long -> Long
- float -> Float
- double -> Double
- boolean -> Boolean

Wrapper Classes (cont.)

Your primitive type becomes and reference type, and now has convenient methods available to it, example for the Integer class include:

- intValue() - returns the value as an int primitive type
- doubleValue() - returns the value as a double primitive type
- longValue() - returns the value as a long primitive type
- toString(Integer) - returns a string representation of the value
- parseInt(String) - returns the String as an int primitive type
- valueOf(String) - returns the String as an Integer reference type
- compareTo(Integer) - Compare two Integers

Autoboxing and Unboxing

Java's compiler will allow *syntactic sugar* for creating wrapper classes, in other words you do not need to explicitly instantiate a wrapper class.

- **Autoboxing** is the automatic conversion of primitive types to the corresponding wrapper classes
- **Unboxing** is the automatic conversion of wrapper class objects to the corresponding primitive types

Let's look at an example...

ArrayList

An **ArrayList** is an ordered list of reference type items. You can think of it as an array, but with more methods associated with it.

- Items in the ArrayList are known as: **Elements**
- An ArrayList will not hold primitive types, so for example you cannot have an ArrayList of booleans
- While you can think of an ArrayList functioning (somewhat) like an Array, you **CANNOT** index into it
    - i.e. in order to access an element the following syntax will NOT work: array[3] = 5;
- ArrayLists will dynamically grow

ArrayList (cont.)

- While an **ArrayList** cannot hold primitive types, we can however use Wrapper classes to accomplish the same functionality
- While we cannot index into an ArrayList to access its' elements, it does provide convenient methods to accomplish the same functionality:
  - add(element) - adds an element to the end of the list
  - get(index) - retrieves the element at the specified index
  - set(index, element) - sets the specified index with the specified element
  - size() - returns the number of element currently in the ArrayList

Let's look at an example...

ArrayList(cont.)

Where to find ArrayList documentation?? Google it of course…

Remember that you need to know your JDK / SEversion so you can know which version of the ArrayList to google.

Example: To find information on ArrayList in SE Version 9 , just Google:

Java ArrayList JDK 9

And click on the link the begins with:

https://docs.oracle.com …