COMP1409: Introduction to Software Development I

Week 11

Mike Mulder (mmulder10@bcit.ca)

Agenda

- Week 10 Review
- Quiz
  - Quiz 10
  - Review Answers
- Lesson 11
  - Iterators
  - Assignment 3 Hint
- Lab 11
  - In Class
- Final Exam Review
- Course Evaluation (Last 15 minutes)

Review - ArrayList

How Do You Declare an ArrayList

```
ArrayList<Book> books;
private ArrayList<Book> books;
```

How Do You Initialize and ArrayList

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

How Do You Use an ArrayList

```
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
```

Review - ArrayList

What are four key methods available on an ArrayList object?

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

“RAGS”

Review – ArrayList (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

Review - ArrayList

What are wrapper classes?

Built-in classes in Java that hold an a primitive type. Examples:

Integer – for int
Boolean – for boolean
Double – for double
Character – for char

What is Autoboxing?

The automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. Autoboxing can make it appear that ArrayLists can hold primitives. Example:

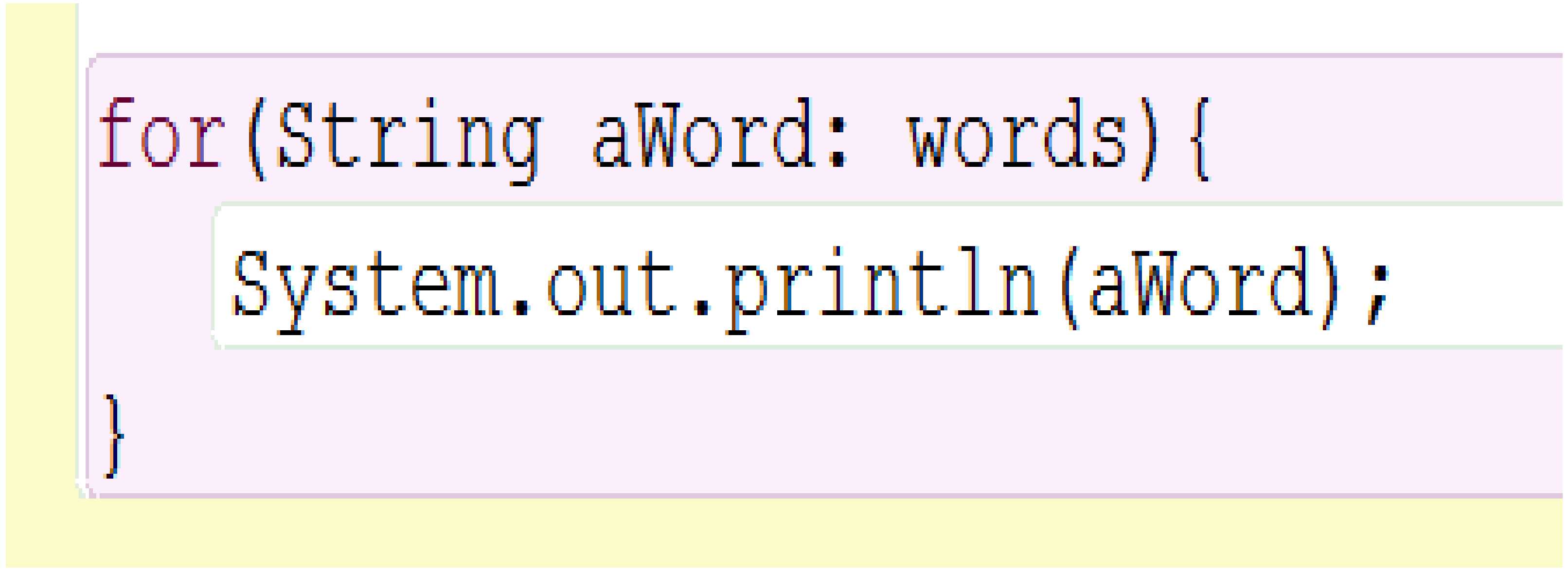
- ArrayList<Integer> myArrayList;
- myArrayList.add(5);

Review - ArrayList

What’s are some of the advantages of the ArrayList over 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

Review – for each loop

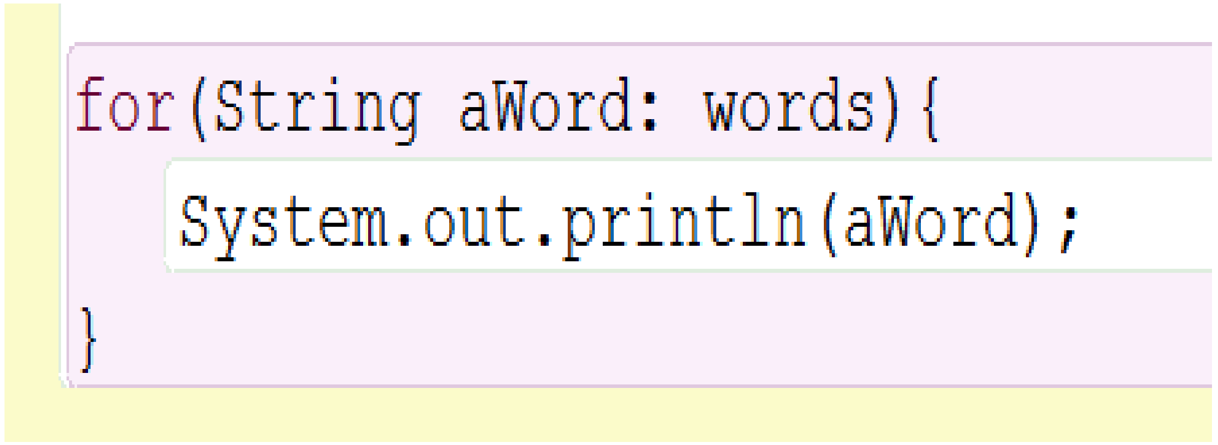


```
for(String aWord: words){
    System.out.println(aWord);
}
```

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

Review – for each loop

```
ArrayList<String> words = new ArrayList<String>();
words.add(“easy”);
words.add(“does”);
words.add(“it);
```



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

What is printed to the console?

```
easy
does
it
```

Review – for each loop

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.

Quiz 10

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 a solution afterwards

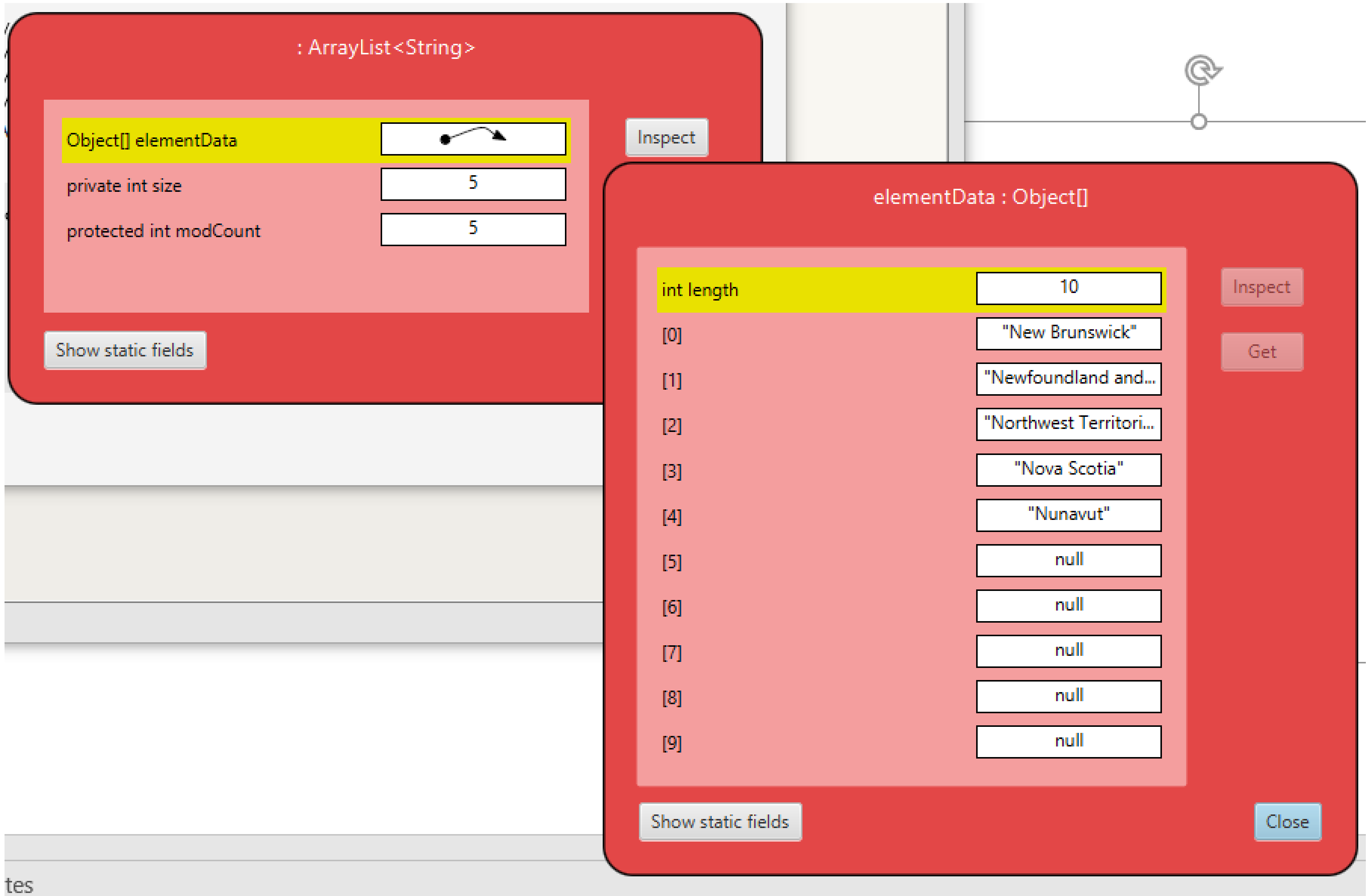
Lab 10 (Last Week)

ArrayList and for each loop

Will be marked this evening

Solution on D2L (Week 10) that we will use for Lab 11 today

Lab 10 – What is this?



: ArrayList<String>

Object[] elementData	
private int size	5
protected int modCount	5

elementData : Object[]

int length	10
[0]	"New Brunswick"
[1]	"Newfoundland and...
[2]	"Northwest Territori...
[3]	"Nova Scotia"
[4]	"Nunavut"
[5]	null
[6]	null
[7]	null
[8]	null
[9]	null

- This is our ArrayList<String> object.
- It holds an array of "Objects" internally.
- It manages the size and contents in the array.
- You don't need to worry about this as a user of the ArrayList.
- You use the public methods (add, get, remove, size, etc.)

Assignment 3

Due Wednesday, Dec. 5 at midnight.

One more hint in today’s lecture.

Learning Outcomes: Lesson 11

- Iterators
- Traversal with Collections
- The four Iterator methods
- Usage with while loop

Iterators

All Collection classes use Iterators

- In Java, Collections include **sets**, **lists** and **maps**
- ArrayLists are lists and therefore use iterators
- Iterator is imported from the java.util package, too
- This allows for granular traversal control
- Collections have an iterator() method which returns an Iterator reference.

Iterators

- Obtain an iterator reference from the ArrayList

```
Iterator<Person> it = personCollection.iterator();
```

- **it** can now be used to traverse through the ArrayList
- **it** will always start traversing from beginning to end of a collection.

The four Iterator methods

- it.hasNext()

  Returns true if there is a next element

- it.next()

  Returns the next element (and also advances the iterator’s position)

- it.remove()

  Removes the current element from the collection

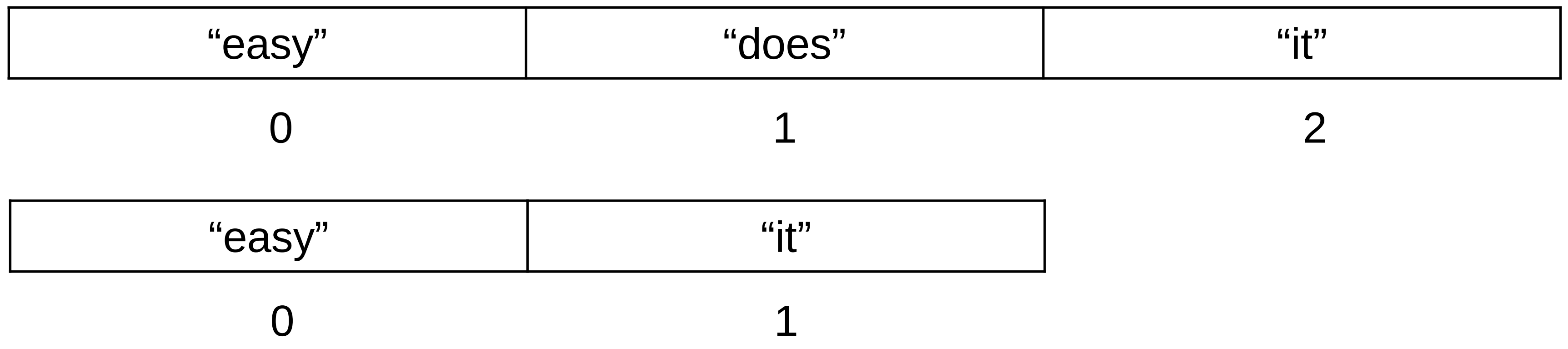
- it.forEachRemaining(action)

  Execute a method on each element that hasn’t been iterated (advanced - not in scope)

The Problem

You cannot remove an element from an Array while using an for loop or a while loop with an index.

Why? You get unexpected results iterating through an ArrayList when the size changes via remove.



What if we remove element at index 1?

The element previously at index 2 is now at index 1

The problem

```
ArrayList<String> words = new ArrayList<String>();
words.add(“easy”);
words.add(“does”);
words.add(“it”);
```

```
for (String word : words) {
    if (word.equals(“does”)) {
        words.remove(word);
    }
}
```

```
for (int i = 0; i < words.size(); i++) {
    if (words.get(i).equals(“does”)) {
        words.remove(i);
    }
}
```

What might happen if we remove the element in the above loops?

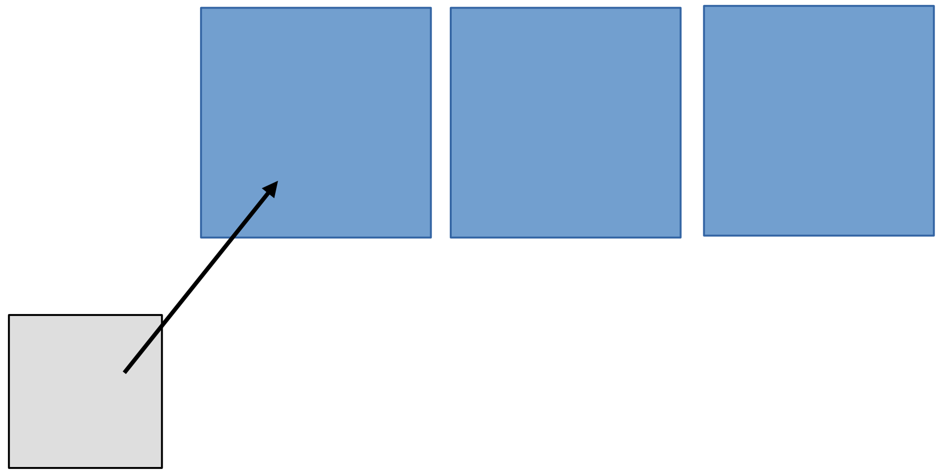
Answer: Depends. May get an exception, may get unexpected results, may work fine.

Iterator usage with while loop

```
Iterator<Person> it = personCollection.iterator();  // declare your Iterator LOCALLY

while(it.hasNext()){                    // check if there’s a next element
    Person p = it.next();               // cache the reference for next element

    System.out.println(                 // print details about current element
        p.getName() + “ is age ” +
        p.getAgeYears()
    );

    if(p.getAgeYears() > 10){
        it.remove();                    // remove element from the collection
    }
}
```

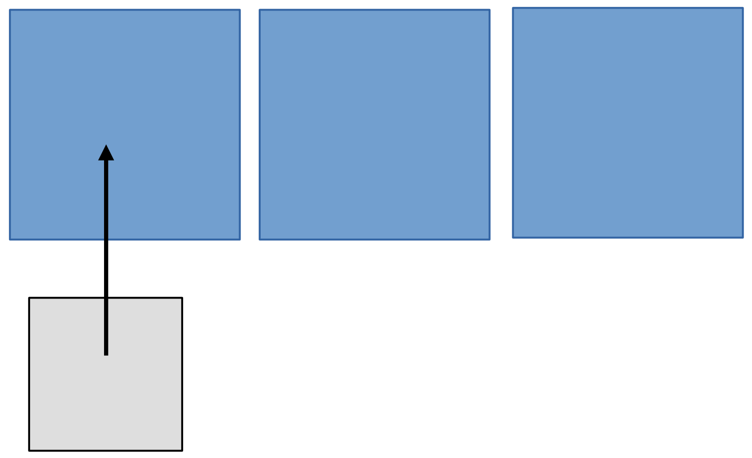
Iterators allow you to remove elements while iterating through a collection.

Usage with while loop



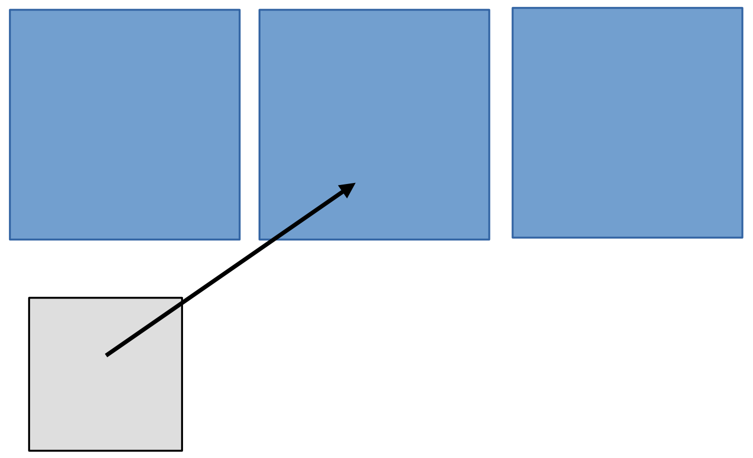
```
it.hasNext();      // true
```

Usage with while loop



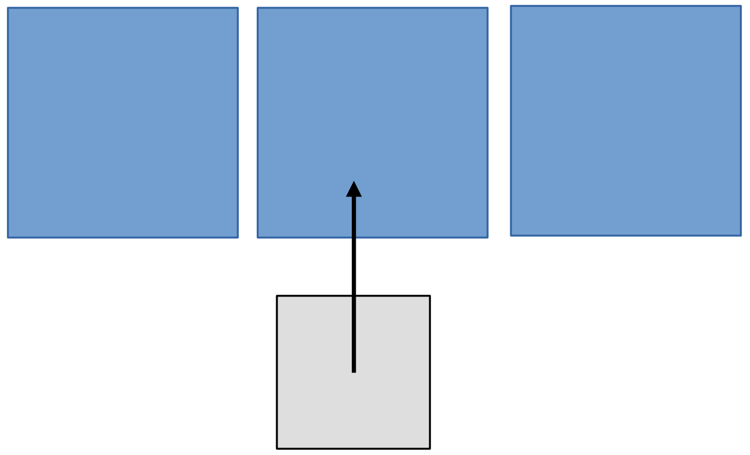
it.next();// first reference

Usage with while loop



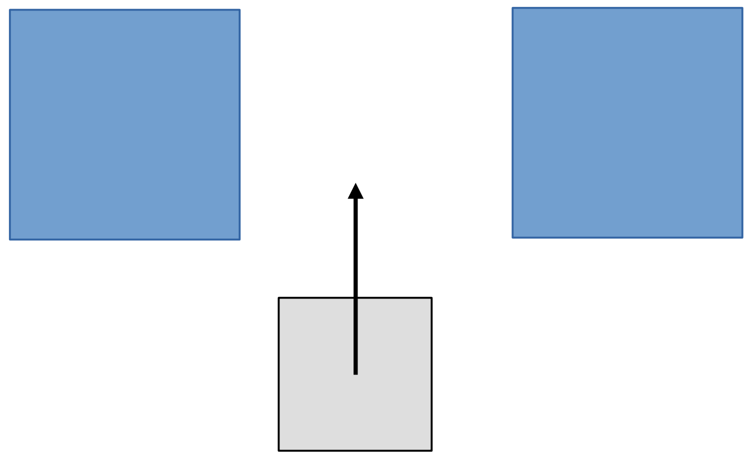
```
it.hasNext();      // true
```

Usage with while loop



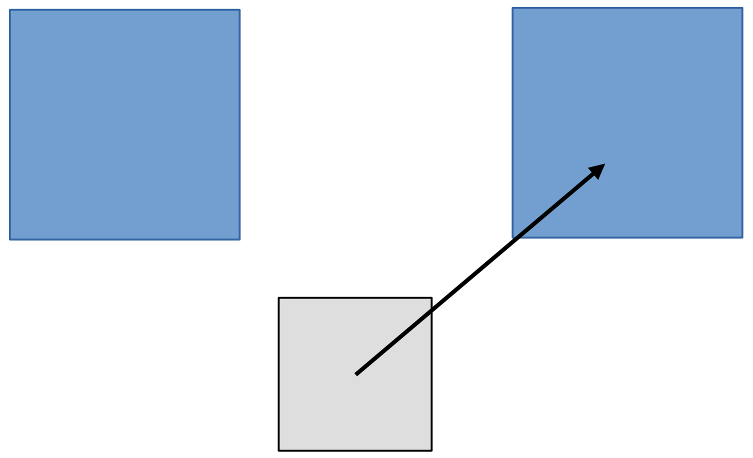
it.next();// second reference

Usage with while loop



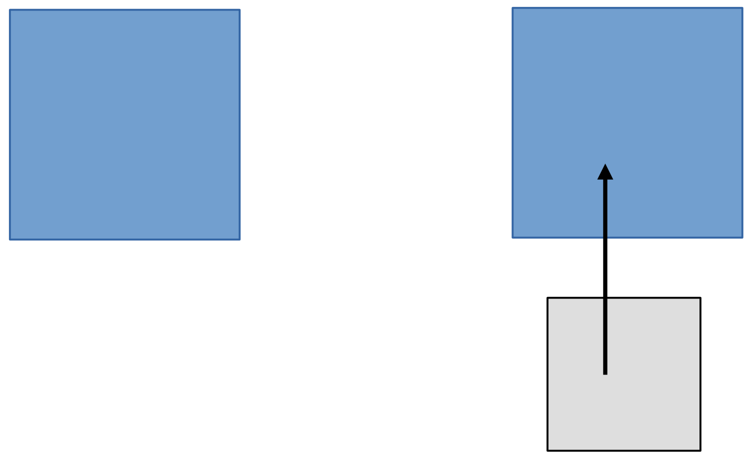
```
it.remove();        // remove current reference
```

Usage with while loop



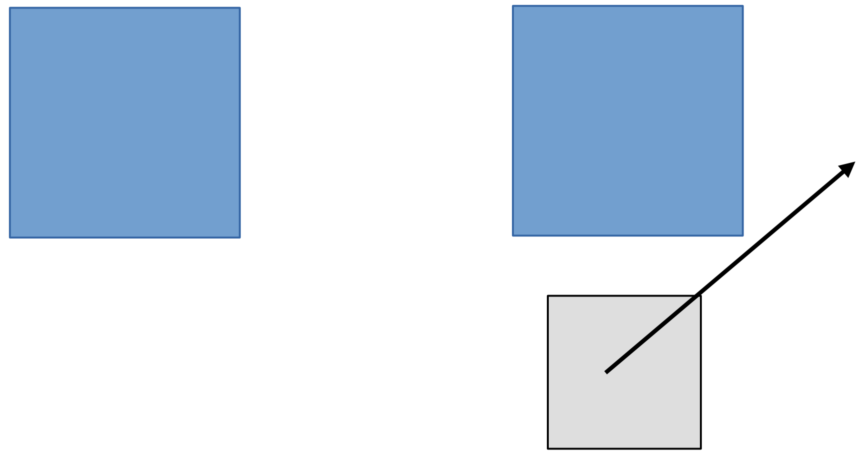
```
it.hasNext();      // true
```

Usage with while loop



it.next();// return next reference

Usage with while loop



```
it.hasNext();      // false
```

Iterator vs. for-each

- Use an Iterator when <u>removing</u> from an ArrayList

- Use an Iterator for more complex Collection classes than ArrayList

Iterator rules

1. import java.util.Iterator
2. make a <u>local</u> Iterator instance
3. use the Collection’s iterator() method
4. while(it.hasNext()){
5. Element e = it.next()

Summary: For Each vs. Iterator

```
for (String aWord : words) {
    System.out.println(aWord);
    words.remove(aWord); // throws exception
}
```

Don’t do this

```
Iterator it = words.iterator();
while (it.hasNext()) {
    String aWord = it.next();
    System.out.println(aWord);
    it.remove(); // okay here
}
```

Do this

Hint: Assignment 3

public Package[] shipPackagesByDestinationCity(String theDestCity)

Removes multiple packages from the warehouse.

Can use an iterator to remove the packages.

An ArrayList has a remove method that takes in the element to remove.

    i.e., packages.remove(package);

Hint: Assignment 3

```
Iterator<Package> it = packages.iterator();

while (it.hasNext()) {
    Package package = it.next();
    if (package != null && <condition>)
    {
        it.remove();
    }
}
```

What's an alternate way to safely remove the packages from your package ArrayList?

Hint: Assignment 3

Remember to include the imports for Assignment 3 in the Warehouse.java class:

```
import java.util.ArrayList;
import java.util.Iterator;
```

Lab 11

Read the lab write-up

Create a new BlueJ project and load in ProvinceTerritory.java and Canada.java from Week 10 in D2L.

I will demo one example

Do the first part individually or in partners

Then we will go through the lab together as a group

Final Exam Review

Final Exam Format

In class next week, 3 hours in total

First 90 minutes – Written/Closed Book (i.e., big quiz)

Second 90 minutes – Coding/Open Book (i.e., small assignment with test)

You cannot start the second part until after the first 90 minutes are complete.

Course Topics

Week	Topics
1	Bluej Definitions: Classes, objects, data types, state, methods Naming conventions A simple java class
2	Comments Visibility modifiers Methods: return types and parameters if statements null Constructors Parameters Assignment statement Logical operators Relational operators
3	Accessors Mutators Console output String concatenation public static void main()
4	Logical operators Local variables Static variables and methods Constants

Course Topics

Week	Topics
5	Thanksgiving – No Class
6	Arithmetic operators Overloading Switch/case
7	Composition (object interaction and external method calls)
8	References; identity versus equality null (again) this Debugging techniques
9	Arrays while loops
10	Remembrance Day – No Class
11	More arrays for loops
12	ArrayList class Enhanced for (foreach) loop
13	Iterators
14	Final Exam

Final Exam Review - Topics

What is a class:

- java file
- describes data and behaviors of a general category

What is an object:

- one instance of a class

Final Exam Review - Topics

Constructor:

- called automatically by java when an object is created
- initialize instance variables properly

Final Exam Review - Topics

Typical Mutator Method:

```
public void setFirstName(String firstName){
	if(firstName != null){
		this.firstName = firstName;
	}else{
		throw new IllegalArgumentException();
	}
}
```

Typical Accessor Method:

```
public String getFirstName(){
	return firstName;
}
```

Final Exam Review - Topics

String Concatenation

```
String s = 4 + 4 + "4" + 4 + (4 + 4) + 4;
84484
```

Main Method

```
public static void main(String[] args)
```

Final Exam Topics

Instance Variables versus:

- local variables (live one method run; no default set by java; only accessible in that method; no access modifier)
- static variables (one copy per class; no object required)
- constants (set once only)
- symbolic constants...constant and final (name in all CAPS)

Final Exam Topics

While Loop:

```
int i = 1;

while(i <= 1000){
     System.out.println(i):
     i++;
}
```

Final Exam Topics

Switch Statement:

```
switch(collegeName){
     case "bcit":
     case "sfu":
          System.out.println("burnaby");
          break;

     case "ubc":
     case "langara":
          System.out.println("vancouver");
          break;

     default:
          System.out.println("nothing");
          break;
}
```

Final Exam Topics

Overloading:

- method (or constructor) with same name but different parameter order, type, or amount

Composition:

- objects contain objects

Final Exam Topics

Object References:

```
String s1 = "hi";
String s2 = "hi";
System.out.println(s1 == s2); // compares
addresses

s1.equals(s2); // compares string contents
```

Final Exam Topics

Arrays and For Loop

```
private Book[] books = new Book[5];
books[0] = new Book(""); // WATCH FOR NULLS

if(books != null){
     for(int i = 0; i < books.length; i++){
          if(books[i] != null){
               if(books[i].getTitle() != null){
                   System.out.println(
                       books[i].getTitle().toLowerCase());
               }
          }
     }
}
```

Final Exam Topics

ArrayList:

```
import java.util.ArrayList;

private ArrayList<Book> books = new ArrayList<>();
books.add(new Book("Book Title"));
```

Final Exam Topics

ArrayList and Loops (for each and iterator):

```
for(Book book : books){
	System.out.println(book.getTitle());
}

import java.util.Iterator;

Iterator<Book> it = books.iterator();

while(it.hasNext()){
	Book b = it.next();
	System.out.println(b.getTitle());
}
```