Iterating Over Things and the for-each Loop

Due date: by the end of the recitation session

Introduction

You might have used a for-each loop in some of your programs. You might have wondered how it works. If you did neither, you still need to complete this activity because iterating over collections of items is something that we will do not only for lists and linear structures, but for all other data structures as well.

Let's start with the following program:

```
import java.util.ArrayList;

public class ForEach {
  public static void main (String [] args ) {
    ArrayList<String> list = new ArrayList<String>();

    list.add("Warsaw");
    list.add("Venice");
    list.add("Atlanta");
    list.add("New York");
    list.add("Rome");

    for ( String city : list )
      System.out.println(city);

    //add duplicates
    for (String city : list )
      list.add(city);

    //remove cities starting with "W"
    for ( String city : list )
      if ( city.startsWith("W") )
        list.remove (city) ;
  }
}
```

What do you think it does? Try to figure it out before attempting to compile it and run it.

The above code uses the for-each loop. It is a handy tool, but (as you might have discovered when trying to run the above code) it does not always work.
This loop is based on iterators and you need to understand a bit more about iterators before you understand why only one of the above for loops works without causing a runtime error.

Worksheet: `https://goo.gl/TyuoZD`

Part 1

For this part you will need to study the documentation and the implementation of your the class that should be familiar by now: the `ArrayList`. The documentation can be found at:
https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Hopefully you still have the source code for OpenJDK that you downloaded for the first recitation activity.
If not, you can download it from the course website at
http://cs.nyu.edu/~joannakl/cs102_s17/source_code/jdk1.8.zip (this is a fairly large file to download!).
The `ArrayList` class is located in the `util` package of `java` directory.

Questions (answer them in the worksheet):

1. Look at the documentation page for `ArrayList` class. List all interfaces that this class implements.
2. Look at the source code for `ArrayList` class. What are the interfaces listed in the definition of the class?
3. Why do you think the lists in questions 1 and 2 are not the same? Is the documentation wrong?
4. Look at the documentation of the `Iterable<E>` interface. List all the methods that are required by that interface. Indicate the ones that are provided by the `ArrayList` class directly (not the ones that may be inherited from other classes).

Part 2 `Iterable<E>` and `Iterator<E>`

One of the methods required by the `Iterable<E>` interface is called `iterator()` - it returns an instance of `Iterator<E>`. Look at the documentation for `Iterator<E>`,
https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html and answer the following questions.

Questions (answer them in the worksheet):

1. Is `Iterator<E>` a class or an interface?
2. List all the methods from `Iterator<E>`.
3. Some of the methods have the `default` keyword listed next to their name. Research what that keyword does/means and briefly explain it.
4. What is the purpose of the `next()` method? In what situation might it throw an exception?
5. What is the purpose of the `hasNext()` method?

Part 3 Implementing the `Iterable<E>` Interface

Look at the source code for `ArrayList` class. Since it is implementing the `Iterable<E>` interface, it should have a method called `iterator()`. That method should be returning an instance of an `Iterator<E>`. Look for the class associated with that instance. Answer the following questions based on the source code that you are looking at.

Questions (answer them in the worksheet):

1. What is the name of the class that represents the object returned by the `iterator()` method of the `ArrayList<E>` class?
2. Where is that class implemented (specify the file name and the line number)? What is the relationship between that class and the `ArrayList<E>` class?
3. Look at the cursor data field in that class. In your own words, describe its purpose.

4. Look at the `next()` method of that class. Ignoring the `checkForComodification()` function call, figure out what the function does and describe the steps.
5. One of the lines in the `next()` method is

   ```
   Object[] elementData = ArrayList.this.elementData;
   ```

   What does this line do? What do you think `ArrayList.this` refers to? Could we replace it with `this`?

Part 4 Iterators and For-Each Loops in a Wild

Go back to the code example from the first page. Try to figure out why it does not work.
Mifos, http://mifos.org/ , is an open source platform for delivering a range of financial services. One of their GitHub repositories is located at https://github.com/apache/incubator-fineract
You will need to locate several files in the Mifos GitHub repository (the one linked above). To simplify the process just go to the main page of the repository, hit 't' (as in the key labeled 't' on your keyboard) and then start typing the name of the file that you are after.
You are going to look at the following files:

- `LoanRepositoryWrapper.java`, function `findLoanByClientId`, line approx. 145
- `ShareAccountWritePlatformServiceJpaRepositoryImpl.java`, function `undoApproveShareAccount`, line approx. 338
- `CalendarUtils.java`, function `convertToLocalDateList`, line approx. 175-179

Use the above source code to answer the questions.

Questions (answer them in the worksheet):

1. The first two files that you looked at demonstrate the use of a for-each loop. What is the type of the collection over which they are iterating? Verify that these collections implement `Iterable<E>` interface.
2. The third file, `CalendarUtils.java`, has a strange looking for loop on lines 175-179. It is not a for-each loop, but it does use an iterator. What is the loop 'counter'? What is the loop continuation condition? How is the 'counter' incremented?