# **Assign Roles**

Make sure everyone in your group has a role. If you have 3 people, combine Facilitator & Process Analyst.

|  |  |
| --- | --- |
| **Team Roles** | **Team Member** |
| **Facilitator**: reads the questions aloud, keeps track of time and makes sure everyone contributes appropriately. |  |
| **Spokesperson**: talks to the instructor and other teams. Compiles and runs programs when applicable. |  |
| **Quality Control**: records all answers & questions, and provides team reflection to team & instructor. |  |
| **Process Analyst**: Considers how the team could work and learn more effectively. |  |

## *Objectives:*

- Explain what the next() and hasNext() methods in an iterator provide.

- Write a code sample that uses an iterator.**Context - (2 minutes)**

1. Explain why it is useful to be able to traverse through a collection of elements that are stored in a data structure such as an array or list.

## Model 1. (5 min) Traversing a List

The following data represents five different objects stored in an ArrayList named foods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| “Apple” | “Banana” | “Cherry” | “Date” | “Eggplant” |
| 0 | 1 | 2 | 3 | 4 |

List<String> foods = new ArrayList<String>();

**Critical Thinking Questions**

1. What is the data type stored in the foods ArrayList?
2. Using the Java code shown below

|  |
| --- |
| String temp; int i = 0;  while(i < foods.size()) {  temp = foods.get(i);  System.out.println(temp);  i++;  } |

describe what the following methods do:  
  
- size()  
  
- get()

1. What would the above code example output?
2. Explain the purpose of the index variable i.
3. Rewrite the above code example using a for loop. (Be sure to make the call to get() inside of System.out.println() so to avoid creating the temporary variable temp.)

|  |
| --- |
| // your code goes below |

## 

## Model 2. (15 min) Iterators

The following model provides an **iterator** for the ArrayList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| “Apple” | “Banana” | “Cherry” | “Date” | “Eggplant” |
| 0 | 1 | 2 | 3 | 4 |
| ↑ |  |  |  |  |

where the up arrow (↑ ) points to the first element in the iterator.   
  
The Java API for an iterator provides two[[1]](#footnote-0) methods: next() and hasNext().

The behavior of these two methods is described below:

|  |  |  |
| --- | --- | --- |
| Method | Return Value | Effects |
| next() | The next element in the iterator. | Advances the logical arrow so that it now points to the subsequent element in the iterator. |
| hasNext() | A Boolean value that returns true if there are more elements provided in the iterator. | Once the last element has been provided by the iterator, this method returns false. |

The following table highlights what the next() and hasNext() methods return after each method is invoked two times:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Invocation | Method | Return value | Method | Return value | Position of ↑after next() is called. |
| 1 | hasNext() | true | next() | Apple | Banana |
| 2 | hasNext() | true | next() | Banana | Cherry |

After these methods have been invoked, the status of the iterator now appears as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| “Apple” | “Banana” | “Cherry” | “Date” | “Eggplant” |
| 0 | 1 | 2 | 3 | 4 |
|  |  | ↑ |  |  |

**Critical Thinking Questions**

1. Explain the role of the ↑ for the iterator.
2. Complete the remainder of the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Invocation | Method | Return value | Method | Return value | Position of ↑after next() is called. |
| 3 | hasNext() |  | next() |  |  |
| 4 | hasNext() |  | next() |  |  |
| 5 | hasNext() |  | next() |  | *<ignore>* |

After the next() method returns the last element in the iteration (i.e. “Eggplant”), the iteration is considered complete and hasNext() returns false.

2. Using the following iterator for a collection of animals:

|  |  |  |
| --- | --- | --- |
| “Lions” | “Tigers” | “Bears” |
| ↑ |  |  |

Fill in the table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Invocation | Method | Return value | Method | Return value | Position of ↑after next() is called. |
| 1 | hasNext() |  | next() |  |  |
| 2 | hasNext() |  | next() |  |  |
| 3 | hasNext() |  | next() |  | *<ignore>* |
| 4 | hasNext() |  | *<ignore>* | *<ignore>* | *<ignore>* |

Notice how the iterator for CTQ 2 does not have a numeric position for each of the animal objects. This demonstrates how the usage of iterators is independent from the data structure (i.e. array, list, set, etc.) that the iteration is being performed on.

The typical usage for using an iterator is with a while loop. The following pseudo-code highlights this usage:

|  |
| --- |
| while (hasNext()) {  next(); } |

Many Java data structures in the standard API provide an iterator. Look through the API for the [java.util.List](https://docs.oracle.com/javase/7/docs/api/java/util/List.html) class, and identify the name of the method that provides an iterator.

3. What is the name of this method?

The Java code shown below is a modification of the Java code for CTQ 1 in Model 1, but this time using an iterator:

|  |
| --- |
| java.util.Iterator<String> itr = foods.iterator();  while(itr.hasNext()) {  temp = foods.next();  System.out.println(temp);  } |

4. Contrast this code example with the Java code for CTQ 1 in Model 1. Do iterators require an index variable?

The Set interface provided in the Java API provides an iterator. Examine the API for the [java.util.Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html) interface.  
  
5. What is the name of the method that provides an iterator for a Set?

Assume the following variable have been declared:

java.util.Set<Integer> mySet;

and several items have been added to mySet.

6. What is the data type of objects in this Set?

7. Provide the Java code that obtains an iterator for mySet, and outputs each element in the set using the iterator you have created.

|  |
| --- |
| java.util.Iterator |

8. In your words, what is the main benefit of using an iterator?

**Group Reflection**

(Submitted by the Process Analyst)

**Key Concepts**:

Identify the most important concepts you learned today.

**Further Clarification Needed**:

What, if anything, is your group still confused about?

**Group Assessment**:

What was a strength of your group’s performance today? An area for improvement?

1. Actually three methods are provided, but we will only use these two. [↑](#footnote-ref-0)