INFO1113 Week 7 Tutorial

Generics, Data Structures and Iterators

Generics

We explored collections in an earlier in the semester. We will be re-examining collections in conjuction with generics. Generics gives us the ability to parameterise types with our classes. You may have used an `ArrayList` earlier in the semester and had to attach type data to ensure the compiler could check your operations.

To define a type parameter within your class, your class will need to contain pair of angle brackets with a identifier.

```
public class Example<T>
```

In the above example, the identifier is `T` and is enclosed within < and > symbols.

Question 1: Generic Container

Write a simple class that can store any element. Any type argument can be passed to the container.

```
public class Container<T> {
    private T element;
    //Add the rest of the methods
}
```

The container must support the following operations:

- `Container(T element)`, constructs a new container and set its internal element to the one passed.
- `boolean isNull()`, Checks if the element in the container is null.
- `T set(T element)`, sets an element in the container.
- `T get()`, returns the stored element.

Question 2: Linked List

We visited earlier in the semester how to construct a linked list with integers. You will now rewrite the `LinkedList` data structure to use generics. Each node will contain a value and a link to the next node.

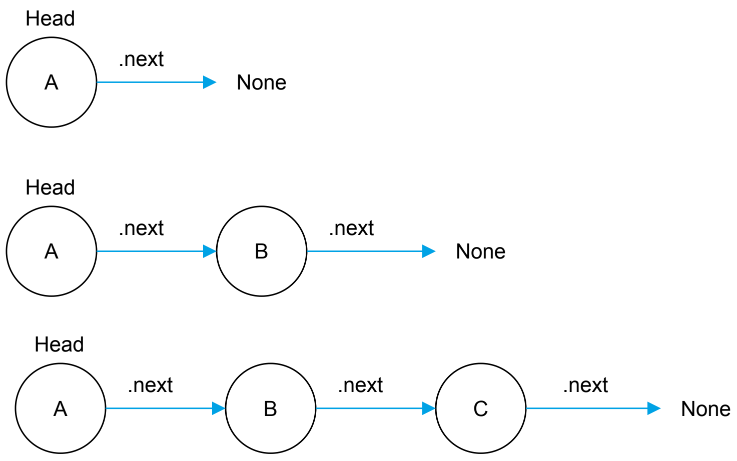

Figure 1: Linked List Diagram

You can use the following definition for `Node` and `LinkedList` and extend it where necessary.

```
class LinkedList<T> {
    //Your linked list class
}

class Node<E> {
    private E element;
    private Node<E> next;
    //Your node class
}
```

You will need to implement the following methods for the class.

- `long size()`, returns the current size of the linked list
- `T add(T element)`, Adds an element to the end of the list
- `T set(int index, T element)`, sets an element at a specific index in the list.
- `T get(int index)`, returns an element at a specific index, should throw IndexOutOfBoundsError if index is < 0 or $>= size()$.
- `T remove(int index)`, removes the element at a specific index and returns it. When a node is removed, it will need to stitch the previous node to the removed node's neighbour.

Please ensure that when your code compiles that it does not contain any warnings related to **unchecked operations**. By ensuring your usage of type parameters can be checked by the compiler.

Collection interfaces

Java includes collection interfaces that will allow your objects to be `iterable` and usable within a `for`-each loop. The two interfaces your collection must utilise are `Iterable` and `Iterator`. Your collection will need to implement `Iterable` in the collection class that will return an implemented `Iterator` class.

You can refer the java documentation for Iterable and Iterator.

Question 3: Making your collections iterable

With one of the collections you have implemented over the semester, implement the `Iterable` and `Iterator` interfaces to allow the collection to used in a `for`-each loop. The `Iterable` interface requires implementing the following methods for the collection type.

For iterable:

- `Iterator<T> iterator()`, returns an iterator for the collection.

You will need to create your own class that will implement the `Iterator` class.

For iterator:

- `boolean hasNext()`, returns true if another element can be return, false if the iterator has finished.
- `E next()`, returns the next element and moves the cursor to the next element.

Question 4: Generic Array

You are to implement a generic array that will allow any only data of a specific type added to the array.

```
public class GenericArray<T> {
    private T[] array;

    public GenericArray() {
        //initialise array here
    }

    //Define your remaining methods
}
```

The array must support the following operations:

- `long size()`, returns the capacity of the array.
- `T set(int index, T element)`, sets an element in the array.
- `T get(int index)`, returns an element at a specific index, should throw IndexOutOfBoundsError if index is < 0 or >= size().
- `long resize(long newSize)`, returns the old size and resizes the array.

Extension

Question 6: Binary Search Tree

Once you finished the previous questions, attempt to implement a Binary Search Tree. A binary search tree can be implemented as a linked data structure where the a node will contain two links, a left ad right child.

You may use the following scaffold for `BinaryTree`.

```java
public class BinarySearchTree<K extends Comparable, V>
    implements Iterable<Node<K, V>> {

    private Node<K, V> root;
    public BinaryTree() {
        root = null;
    }
}
```

When adding an element to the tree, your element should be added to the root if it is null (or the same key), if the root is not null, it should check to see if the it is less than the root (keys must use `compareTo()` method) and attempt to add to the left child, else add to the right child.

- `long size()`, returns the number of elements added to the tree.
- `void put(K key, V value)`, puts a value in the
- `V get(K key)`, Using the key, it will return the value mapped to the key.
- `V remove(K key)`, removes the `Node` in the tree and returns the value

Question 7: In-order traversal

Once you have completed your binary tree implementation. You will need to implement an iterator that will traverse the binary tree with an in-order traversal.

Pseudo-code for an in-order traversal

```
inorder(root):
  if left_child of root is not null:
    inorder(left_child)

  visit(root)

  if right_child of root is not null:
    inorder(right_child)
```

It is best that you attempt implement the traversal without using an iterator first so you are familiar with the method, then to transform it to an iterator object where it will maintain state.

Question 8: Assessed Task: Assignment 1

Remember you are required to complete Assignment 1 within the due date. Go to EdStem for this unit and click on Assessment to find out the task and the due date. This is a marked task.