

CS1020: DATA STRUCTURES AND ALGORITHMS I

Tutorial 3 – Collection, Array-Based List, Generics

(Week 5, starting 8 February 2016)

1. *Collection*<E> vs *List*<E>

Java defines 2 interfaces: `java.util.Collection` and `java.util.List`.

Collection: <http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html>

List: <http://docs.oracle.com/javase/7/docs/api/java/util/List.html>

Each interface is an abstraction – an idea without any implementation – and can be used as a data type.

(a) What is the relationship between a *Collection* and a *List*?

(b) What do they have in common?

(c) What extra functionality does *List* have?

Tip: Learn to compare, contrast, and draw links between different concepts/topics you've learnt! Hopefully, by the end of the semester, you would have mapped out and understood the relationships between all the data structures covered in this module.

2. *ArrayList*<E> / *Vector*<E>

A very “simple” way to compress a string of characters is to **group adjacent identical characters** into a unit. Each unit contains two pieces of information: the character, and its number of occurrences. In this way, much space is saved if the original string contains many duplicated characters side-by-side.

```
class Pair <T, S> {
    private T _objLeft;
    private S _objRight;

    public Pair(T pObjLeft, S pObjRight) {
        _objLeft = pObjLeft;
        _objRight = pObjRight;
    }
    public T getLeft() { return _objLeft; }
    public S getRight() { return _objRight; }
    public String toString() { // aids debugging
        return "(" + _objLeft + "," + _objRight + ")";
    }
}
```

(a) Complete the method `compressString(String input)`. The method returns an `ArrayList` containing pairs of (character, number of occurrences) in sequence.

For example, if we call

```
result = compressString("AAAAABBBAA");
```

result should be an `ArrayList` that contains three `Pair` references: `{('A', 5), ('B', 3), ('A', 2)}`

```
class Zip {
    public _____ compressString(String input) {
        /* TODO: compress input and return ArrayList of pairs */
    }
}
```

(b) Can we replace all occurrences of `ArrayList<...>` with `Vector<...>`?

3. Generics

Examine the following classes and their relationships:

```
class Animal { ... }
class Flyer extends Animal { ... }
class Glider extends Flyer { ... }

class Writing<T> { ... }
class Book<T> extends Writing<T> { ... }
class Birdopedia extends Book<Flyer> { ... }
```

(a) Regarding generics, where should angled brackets <> appear, and where should they NOT appear?

(b) Are each of these cases, independently, allowed?

	Yes/No	Why or Why Not
Book book = new Book(...);		
Book book = new Book<Animal>(...);		
Book<Animal> book = new Book(...);		
Book<Animal> book = new Book<Animal>(...);		
Book<Flyer> book = new Book<Animal>(...);		
Book<Animal> book = new Book<Flyer>(...);		
Writing<Flyer> writing = new Book<Flyer>(...);		
Book<Flyer> book = new Writing<Flyer>(...);		
Book<Flyer> book = new Birdopedia(...);		

- Learn how to learn 😊 -

Revise a concept
Think of a few cases
Test your understanding
Prepare well for tutorial