Hyperion Development

# Task: Lists

www.hyperiondev.com

# Introduction

**Collecting information**

In this task you are going to learn about the collections framework in java. This is essential component which gives you access to multiple libraries and data structures to enhance the functionality of your program. Within this framework, we'll focus on learning about the structure and implementations of LinkedLists and ArrayLists, whilst analysing what makes them similar as well as unique. This kind of analysis is key to have the most optimised program with the best code implementation. Let's get started!

-The Hyperion Team


**Start collecting**

We'll start by looking at the java.util package. This important package contains a large assortment of classes and interfaces that support a broad range of functionality. For example, java.util has classes that generate pseudorandom numbers, manage date and time, observe events, manipulate sets of bits, tokenize strings, and handle formatted data. The java.util package also contains one of Java's most powerful subsystems: the Collections Framework. The Collections Framework is a sophisticated hierarchy of interfaces and classes that provide state-of-the-art technology for managing groups of objects. It merits close attention by all programmers.

Now that you are familiar with the collection interface, you are ready to examine the standard classes that implement them. Some of the classes provide full implementations that can be used as-is. Others are abstract, providing skeletal implementations that are used as starting points for creating concrete collections.

**ArrayLists**
Standard Java arrays are of a fixed length, which means that after they are created, they cannot expand or shrink. On the other hand, ArrayLists are created with an initial size, but when this size is exceeded, the collection is automatically enlarged. When objects are removed, the ArrayList may shrink in size. Note that the ArrayList class is in the java.util package, so it's necessary to import it before using it.

The process of creating an ArrayList is the same as you would any for any object:

```
import java.util.ArrayList;
//...
ArrayList colors = new ArrayList();
```

You can optionally specify a capacity and type of objects the ArrayList will hold:

```
ArrayList<String> colors = new ArrayList<String>(10);
```

The code above defines an ArrayList of Strings with 10 as its initial size.

ArrayLists store objects. Thus, the type specified must be a class type. You cannot pass, for example, int as the object's' type. Instead, use the special class types that correspond to the desired value type, such as Integer for int, Double for double, and so forth.

The ArrayList class provides a number of useful methods for manipulating its objects. The add() method adds new objects to the ArrayList. Conversely, the remove() methods remove objects from the ArrayList.

Other useful methods include the following:
- **contains**(): Returns true if the list contains the specified element
- **get**(int index): Returns the element at the specified position in the list
- **size**(): Returns the number of elements in the list
- **clear**(): Removes all of the elements from the list

***Note:*** *As with arrays, the indexing starts with 0.*

**LinkedLists**

The LinkedList is very similar in syntax to the ArrayList. You can easily change an ArrayList to a LinkedList by changing the object type. The most notable difference between the LinkedList and the ArrayList is in the way they store objects:

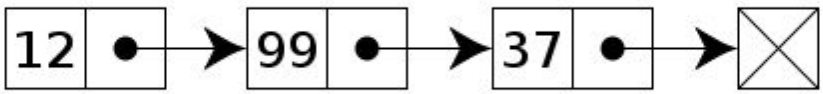
- The ArrayList is better for storing and accessing data, as it is very similar to a normal array.
- The LinkedList is better for manipulating data, such as making numerous inserts and deletes.

The process of creating an ArrayList is the same as you would any for any object:

```
LinkedList<String> c = new LinkedList<String>();
```

***Note:*** *You cannot specify an initial capacity for the LinkedList.*

In addition to storing the object, the LinkedList stores the memory address (or link) of the element that follows it. It's called a LinkedList because each element contains a link to the neighboring element.




Since the LinkedList is an array, you can use the enhanced for loop to iterate over its elements.

**Summary**

- Use an ArrayList when you need rapid access to your data.
- Use a LinkedList when you need to manipulate your data.

## Instructions

First read example.java, open it using jGRASP (Right click the file and select 'Open with jGRASP").

- In this folder there is a file called example.java
- Open this folder using the JGRASP program. You can either do this by right clicking on the example.java file, going to the 'Open with' menu, and selecting JGrasp.exe. Alternatively, you can run the JGRASP program on your computer, go to the top left corner of the program, select File->Open and navigate to example.java on your hard drive and double click on it to open it.
- Once example.java is open in JGRASP please read all of its content very carefully. This will help you understand the basic structure of a Java program.
- There is a compulsory task at the end of the example.java document. The instructions of what to do to complete the task can be found here. You must complete this exercise to continue onto the next task of this course.

# Compulsory Task 1

**Follow these steps:**

- Create a new java file called QuizScores.java
- Import the java.util.ArrayList package.
- Create a public class called 'QuizScores'.'
- Within this class, create a main method.
- Within this method, create an integer ArrayList called 'Group C'.
- Add ten elements to the ArrayList using the add() function to represent scores of learners from Group C.
- Print out the ArrayList
- Print out the highest score..
- Remove the lowest score, and print out the next lowest score.
- Print out the average score.
- Compile, save and run your file.

# Compulsory Task 2

### Follow these steps:

- Create a new java file called Speed.java
- Import the java.util.LinkedList package.
- Create a public class called 'Speed'.
- Within this class, create a main method.
- Within this method, create an String LinkedList called 'Cars'.
- Add ten elements to the LinkedList to represent names of cars.
- Insert a car at a unique index in the LinkedList.
- Remove the car that used to be at that index.
- Print out the LinkedList (it may be useful to use a for loop).
- Print out the size of the LinkedList.
- Print out any cars that contain a vowel in their name.
- Compile, save and run your file.

### Things to look out for

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **jGRASP or Java** may not be installed correctly.
2. If you are not using Windows, please ask your tutor for alternative instructions.

**Still need help?**

Just write your queries in your comments.txt file and your tutor will respond.

# Task Statistics

Last update to task: 03/02/2016.
Authors: Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: [Hyperion Development Feedback.]