COMP1102/8702 – Practical Class 7

Aims and Objectives

This laboratory has been designed to help you

1. allow you to exercise your array declaration, initialisation and manipulation skills,
2. give you practice in defining classes which use an array as the basis for an implementation of a a sequential data structure, such as a list.

Task 1

The *main* method in the class `ArrayTask` declares and initialises an array named `intList` (or more precisely, the variable `intList` contains a reference to an array object) to contain the five `int` values `5`, `20`, `32`, `7` and `9`.

```
public class ArrayTask {

   public static void main(String[] args) {
      int[] intList = {5,20,32,7,9};
      for (int i = 0; i < intList.length; i++) {
         System.out.println("intList[" + i + "]: " + intList[i]);
      }
   }
}
```

The *for* loop prints out each element of the array, producing the following output:

```
intList[0]: 5
intList[1]: 20
intList[2]: 32
intList[3]: 7
intList[4]: 9
```

1. Start *IntelliJ* and open the project named "Practical07" (download if from FLO).
2. Open the file named `ArrayTask.java`.
3. Modify the *for* loop in the method main so that the values in the array `intList` are printed out in reverse order. When run, the modified program should produce the following output:

```
intList[4]: 9
intList[3]: 7
intList[2]: 32
intList[1]: 20
intList[0]: 5
```

Checkpoint 32

Have the program source code and output marked by a demonstrator

Task 2

For this task you will add code to the program from Task 1 so that it also calculates and prints the sum of all the elements in the array `intList`.

1. Add statements to the method *main* as described below.
   i Declare an `int` variable called `sum` and initialise it to 0.
   ii Include an additional statement in the *for* loop which adds the ith element of the array to the variable `sum`.
   iii Include a statement to print the value of `sum` (see example below).
2. Compile and run the program. The output produced should be:

```
intList[4]: 9
intList[3]: 7
intList[2]: 32
intList[1]: 20
intList[0]: 5
Sum = 73
```

Checkpoint 33

Have the program source code and output marked by a demonstrator

Task 3

For this task you will add code to the program from Task 2 so that it makes a copy of the array and increases each element by 1.

1. Add statements to the method *main* as described below.
   i Each value in `intList` should be increased by 1 (***this must be done by changing the values in the array and not by adding 1 when the value is printed***). The resulting output should now be:

```
intList[4]: 10
intList[3]: 8
intList[2]: 33
intList[1]: 21
intList[0]: 6
Sum = 78
```

   ii Declare another array called `copy` with the same type as `intList`. That is, it is also an array of 5 `ints`.
   iii Immediately after the declaration of `copy`, include another *for* loop which assigns to each element of `copy` the respective element of `intList`. After doing this both arrays should contain the same values. Include statements to calculate and then print the sum of the elements of `copy`.

2. Compile and run the program. The output produced should be:

```
intList[4]: 10
intList[3]: 8
intList[2]: 33
intList[1]: 21
intList[0]: 6
Sum of intList = 78
Sum of copy = 73
```

Checkpoint 34

Have the program source code and output marked by a demonstrator

Task 4

For this task you will define and add another class

1. Change the method *main* to contain the following code:

```
public class ArrayTask {

   public static void main(String[] args) {
      IntList list1 = new IntList();
      list1.printList();
      list1.insertAtEndofList(42);
      list1.printList();
      list1.insertAtEndofList(2);
      list1.printList();
      System.out.println(list1.getElementAt(3));
      System.out.println(list1.getElementAt(-1));
      System.out.println(list1.getElementAt(1));
      list1.insertAtEndofList(7);
      list1.insertAtEndofList(9);
      list1.printList();
      list1.insertAtEndofList(100);
   }
}
```

2. Create a new class called `IntList`. The aim of the class is to represent a list. It is initially empty but can have elements inserted at the end of it. Since arrays are not extensible, one way to represent the list is to create an array which is long enough for all foreseeable circumstances and to keep track of how much of the array is being used with another variable. Initially this variable will be zero and each time a value is inserted it will become one larger. The class should include the following:

   i An instance variable which refers to an array containing 4 `ints`.

   ii An instance variable called `listLength` to keep track of how many elements of the array are being used. It should be initialised to 0.

   iii A method called `printList` which prints out all of the elements in the list. If the list does not contain any elements then `"List is empty"` should be printed (see example output below).

   iv A method called `insertAtEndofList` which inserts a value (provided as a formal

parameter) at the end of the list. To do this it should use `listLength` to index the array and assign the value to that element of the array. `listLength` should then be increased by 1 and a message produced (see example below).

Prior to performing the insertion the method should check that there are unused elements in the array (make use of the `length` attribute of the array and `listLength`). If there are not, the insertion should not be performed and an appropriate message should be output (see example below).

v A method called `getElementAt` which returns the value at a given index (similar to array indexing). It should check that the index passed as a parameter is that of an element which is being used. For example, after adding two elements to a list (calling `insertAtEndofList` twice), the only valid indexes will be 0 and 1. If the index is not valid, a message should be printed and 0 returned (see example below).

A good strategy to adopt in developing this class is to write and test each method one at a time. To do this you will need to comment out those lines in the method *main* which call methods which have not been written yet. You may also want to set up the contents of the list manually so, for example, you could write and test `printList`. e.g.

```
private int[] theList = {42,2,0,0};
private int listLength = 2;
```

Once you are convinced `printList` is performing correctly you will need to change the above declarations to represent an empty list with a maximum length 4 (instead of explicitly initialising the array).

Note: The method *main* should not be changed from that given above.

The final output of your program should be:

```
List is empty
42 inserted
Printing list...
Element 0 = 42
2 inserted
Printing list...
Element 0 = 42
Element 1 = 2
Invalid index: 3
0
Invalid index: -1
0
2
7 inserted
9 inserted
Printing list...
Element 0 = 42
Element 1 = 2
Element 2 = 7
Element 3 = 9
Insertion of 100 failed
```

Checkpoint 35

Have the program source code and output marked by a demonstrator

Task 5 (Extension Practice)

1. Change the method *main* to contain the following code:

```
public class ArrayTask {

   public static void main(String[] args) {
      IntList list1 = new IntList();
      list1.insertAtEndofList(42);
      list1.insertAtEndofList(2);
      list1.insertAtEndofList(7);
      list1.printList();
      if (list1.contains(2)) {
         System.out.println("Contains 2");
      }
      if (list1.contains(54)) {
         System.out.println("Contains 54");
      }
      list1.delete(99);
      list1.delete(2);
      list1.printList();
      list1.insertAtEndofList(2);
      list1.printList();
   }
}
```

2. Extend the class `IntList` to include the following methods:

   i A method called `contains` which returns `true` if the value, passed as a parameter, is contained in the list. Otherwise `false` is returned (see example below).

   ii A method called `delete` which deletes the first occurrence of the value, passed as a parameter, from the list. As a result the list will become shorter by one and all the elements after the deleted value will be shifted to the left by one position to remove the "*hole*" (see example below)

   The final output of your program should be:

```
42 inserted
2 inserted
7 inserted
Printing list...
Element 0 = 42
Element 1 = 2
Element 2 = 7
Contains 2
Deletion of 99 failed
2 deleted
Printing list...
Element 0 = 42
Element 1 = 7
2 inserted
Printing list...
Element 0 = 42
Element 1 = 7
Element 2 = 2
```