# Unit 2 Assignment

## GENERAL

1. Save the work of each exercise in a separate Java source code file and name it *StuPrefixClassName.java* and follow the coding style as explained in Unit 1 Assignment.

2. Overall comment your program appropriately (file prolog comment, method prolog comment, and comments on the code). Pay attention to the standard stuff like coding style, meaningful identifier names, indention, and locations of braces.

3. When you’re done with an exercise, include a screenshot of the execution of the program in your assignment report document. This is to remind you to verify the execution result.

## EXERCISES

1. (Name this program StuPreE1ShiftRight) Write a method shiftRight that takes a parameter of int array and that returns an array that is “right shifted” by one position. The rightmost element should be “wrapped around” and place at index 0. You may modify and return the given array, or return a new array. The main() should unit test the method using assertion. Please at least test all four testing cases listed here:

Example Input | Expected Output

{1, 2, 3, 4, 5} | {5, 1, 2, 3, 4}

{1, 2} | {2, 1}  
{20} | {20}

{} // empty arr | {}

You may follow this example to unit test:

int[] arr; // original arr

int[] expected; // expected result

int[] result; // actual result returned from method

arr = new int[]{1, 2, 3, 4, 5};

expected = new int[]{5, 1, 2, 3, 4};

System.out.println(Arrays.toString(arr)); // print original before calling method, in case it's modified in method

result = shiftRight(arr);

if (! Arrays.equals(result, expected) )

System.out.println( "Rightshift failed. Got " + Arrays.toString(result) );

// reuse arr, expected, result in additional testing cases

...

The above code uses two methods from a utility class Arrays. Arrays.equals() compares the content of two arrays. Arrays.toString() returns a string representation of an array’s content (used in Figure 6.14.2 of ch6.14 Perfect size arrays). You will need import java.util.Arrays;

1. (Name this program StuPreE2OddEven) Write a method isOddEven that takes a parameter of int array and that returns true if all odd numbers in the array come before all the even numbers. The main() should unit test the method. Please at least test all testing cases listed here.

Example Input | Expected Output

{7, 3, 6, 6} | true

{1, 2, -3} | false  
{-4} | true

{} // empty arr | true // no violation

1. (Name this program StuPreE3Max) Write a method maxToEnd that takes an ArrayList of String as a parameter and that moves the largest string in the list to the tail, otherwise preserving the relative order of the elements. Assume there is at least one element in the list. If there are multiple copies of the maximum value, move only the first copy.

Strings should be compared lexicographically with compareTo() of String class. If needed, review ch3.13 String comparisons.

// move the largest string (lexicographical order) to the tail,

// otherwise preserve the order of elements

// assume at least one element

public static void maxToEnd(ArrayList<String> list) {

// ADD code

}

Example Input | Expected Output (list updated)

{"the", "best", "day", "ever"} | {"best", "day", "ever", "the"}

{"one", "Value"} | {"Value", "one"}   
{"three", "Two", "three", "four"}| {"Two", "three", "four", "three"}

The main() should unit test the method using assertion. Please follow the example code below and at least test all testing cases listed here.

String[] strArr; // will be used to convert a list of values to arraylist

ArrayList<String> strList;

ArrayList<String> expected;

strArr = new String[] {"the", "best", "day", "ever"};

strList = new ArrayList<>(Arrays.asList(strArr));

System.out.println(strList); // print original

maxToEnd(strList);

expected = new ArrayList<>(Arrays.asList(new String[] {"best", "day", "ever", "the"}));

if ( !strList.equals(expected) )

System.out.println("failed. Result: " + strList);

// reuse the variables to add additional testing cases

...

Arrays.asList(*arr*) returns an ArrayList object containing the values stored in the array parameter. You need: import java.util.Arrays;

1. Reflection: answer those questions AFTER you’ve completed this assignment:
2. What do you like (or dislike) about arrays? What about ArrayList?
3. What’s the hardest part of this assignment for you? Please explain.

## SUBMISSION

Submit three .java files + one word/PDF document. Please put all screenshots and answers to the reflection question into your word/PDF document.

* Exercise 1 ~ 3: provide the java file and a screenshot of the execution result for each exercise
* Exercise 4: Assignment reflection
* Check the completeness of your work against the rubric before turning it in.

## Rubric: Unit 2 Assignment

| **Criteria** | **Ratings** | | | **Pts** |
| --- | --- | --- | --- | --- |
| **Exercise 1 (ShiftRight)** | 5.5 pts. Correct method meeting requirements. | 5 pts ~ 0.5 pts. At least one requirement is incorrect or missing:  (1 pts) The last item saved in [0] of a proper array;  (2 pts) items [1~2nd to the last spot] properly shifted and saved in a proper array;  (1 pts) Method takes one single array param and returns an array with the shifted result;  (1.5 pts) at least three additional testing cases in main() as required. | 0 pts. Incorrect or missed all requirements; no submission. | 5.5 |
| **Exercise 2 (OddEven)** | 6 pts. Correct method meeting requirements. | 5.5 ~ 0.5 pts. At least one requirement is incorrect or missing:  (2 pts) Loop through the array to identify invalid situations;  (1 pts) return true if and only if there is no violation of the rules;  (1 pts) return false if and only if the array violates the rules;  (2 pts) at least four required testing cases in main(). | 0 pts. Incorrect or missed all requirements; no submission | 6 |
| **Exercise 3 (Max)** | 5.5 pts. Correct method meeting requirements. | 5 ~ 0.5 pts. At least one requirement is incorrect or missing:  (2 pts) Loop through ArrayList to identify the 1st copy of a max string;  (1 pts) identified max string moved to the tail of the ArrayList;  (1 pts) the relative order of other elements unchanged;  (1.5 pts) at least three additional testing cases in main() as required. | 0 pts. Incorrect or missed all requirements; no submission | 5.5 |
| **Exercise 4. Reflection** | 1 pts. Answered both questions. | 0.5 pts. Answered only one question. | 0 pts. Didn’t answer the questions; no submission. | 1 |
| **Required program name (StuPre part) + Style Points (Proper comments; meaning identifier names; consistent indentation)** | 2 pts. Correctly named program and proper style in all three areas of style points. | 1 pts. Problems in one of the following areas: program name, three areas of style points. | 0 pts. Problems in two or more of the following areas: program name, three areas of style points. | 2 |
|  |  |  | *Total Points* | 20 |