# Unit 3 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 StuPreSelectRectanglar) Image processing relies on multi-dimensional arrays. Write a method selectRectanglar that simulates selecting a rectangular area of an image.

// Returns a full 2d int array with data from the rectangular area

// specified by startRow, startCol, endRow, and endCol if those

// four specify a valid area. Otherwise return null.

// Also return null if image is null or has zero rows or zero cols

public static int[][] selectRectanglar(int[][] image,

int startRow, int startCol,

int endRow, int endCol) {

// ADD code

}

Your method should return null on invalid/bad parameters as explained in the given method prolog comment. It should also test for invalid starting/ending row/col values and return null.

Your method should allow different (startRow, startCol) and (endRow, endCol) combinations as long as they form a valid area.

For example, given this 2d image array:

0 1 2 3 4 5 6 7 8

10 11 12 13 14 15 16 17 18

20 21 22 23 24 25 26 27 28

30 31 32 33 34 35 36 37 38

40 41 42 43 44 45 46 47 48

The following start/end combinations all identify the same area. The starting position is identified with grey highlight and the ending position with blue highlight.

(1, 2) ~ (3, 7) // from top left corner to bottom right corner

0 1 2 3 4 5 6 7 8

10 11 12 13 14 15 16 17 18

20 21 22 23 24 25 26 27 28

30 31 32 33 34 35 36 37 38

40 41 42 43 44 45 46 47 48

(1, 7) ~ (3, 2) // from top right corner to bottom left corner

0 1 2 3 4 5 6 7 8

10 11 12 13 14 15 16 17 18

20 21 22 23 24 25 26 27 28

30 31 32 33 34 35 36 37 38

40 41 42 43 44 45 46 47 48

(3, 2) ~ (1, 7) // from bottom left corner to top right corner

0 1 2 3 4 5 6 7 8

10 11 12 13 14 15 16 17 18

20 21 22 23 24 25 26 27 28

30 31 32 33 34 35 36 37 38

40 41 42 43 44 45 46 47 48

(3, 7) ~ (1, 2) // from bottom right corner to top left corner

0 1 2 3 4 5 6 7 8

10 11 12 13 14 15 16 17 18

20 21 22 23 24 25 26 27 28

30 31 32 33 34 35 36 37 38

40 41 42 43 44 45 46 47 48

Be aware that an area formed by the same starting position and ending position, such as (3, 5) ~ (3, 5) covers one single pixel and is still valid. That would lead to a 2d array with 1 single row and 1 single column.

Your main() should test the method. Please at least test four valid calls and two invalid calls. You may follow this example:

int[][] image = new int[][] {

{0, 1, 2, 3, 4, 5, 6, 7, 8},

{10, 11, 12, 13, 14, 15, 16, 17, 18},

{20, 21, 22, 23, 24, 25, 26, 27, 28},

{30, 31, 32, 33, 34, 35, 36, 37, 38},

{40, 41, 42, 43, 44, 45, 46, 47, 48}

};

System.out.println(Arrays.deepToString(image)); // original

int startRow = 1, startCol = 2, endRow = 3, endCol = 7;

int[][] select; // store selected area

// (valid 1) from top left corner to bottom right corner:

// (startRow, startCol) ~ (endRow, endCol)

select = selectRectanglar(image, startRow, startCol, endRow, endCol);

System.out.printf("(%d, %d) - (%d, %d): %s\n", startRow, startCol, endRow, endCol, Arrays.deepToString(select));

// reuse variables to add additional testing cases

...

The above segment of code will produce this output:

[[0, 1, 2, 3, 4, 5, 6, 7, 8], [10, 11, 12, 13, 14, 15, 16, 17, 18], [20, 21, 22, 23, 24, 25, 26, 27, 28], [30, 31, 32, 33, 34, 35, 36, 37, 38], [40, 41, 42, 43, 44, 45, 46, 47, 48]]

(1, 2) - (3, 7): [[12, 13, 14, 15, 16, 17], [22, 23, 24, 25, 26, 27], [32, 33, 34, 35, 36, 37]]

Arrays.deepToString() is similar to Arrays.toString() but works with 2D arrays (and higher dimension arrays). You need: import java.util.Arrays;

A downside of Arrays.deepToString() is that it always returns the content in one long string. Feel free to replace it with a method of your own to print a 2d array in a nicer format.

1. (Name this program StuPreTestNumFileArr) Revise the TestNumFile program from lecture Unit3 File Input and save your work in a new program TestNumFileArr. We will practice sorting and insertion following those steps:

Step 0. First set up the given code in your new program and make sure it works properly. Take a screenshot of its execution and include it in your assignment report.

Step 1. Add a private helper method:

// return a partially filled int array with all elements from ArrayList

// param in the same sequence and 5 extra spots at the end

private static int[] toIntArray(ArrayList<Integer> list) {

// ADD code

}

We want to copy the data from the ArrayList to an array. To allow space for additional operations, we also want this array to be a partially filled array. We accomplish this by creating an array whose capacity is 5 + list.size().

Step 2. Add the following code to the end of main():

int[] numArr = null; // to hold data

int size = 0; // track actual # of elements.

numArr = toIntArray(numList);

size = numList.size();

System.out.println("Now in arr: " + Arrays.toString(numArr));

// need import

This segment of code should print the same data plus 5 zeros at the end:

Now in arr: [15, 6, 74, 28, 90, 34, 0, 0, 0, 0, 0]

Step 3. Add a private helper method to sort this partially filled array following the given method prolog comment. You must write your own sorting code and use selection sort or insertion sort.

// sort a partially filled array ([0 ~ numOfElements-1]) into

// ascending order

// Will return and not modify the arr if arr is null or

// numOfElements is invalid

private static void sort(int[] arr, int numOfElements) {

// ADD code

}

numOfElements is invalid if it’s negative or larger than arr.length.

Step 4. Add the following code to main() to test the sorting method (continue from step 2 work):

sort(numArr, size);

System.out.println("After sorting: " + Arrays.toString(numArr));

This segment of code should print the data after sorting. Be aware that the 5 zeros at the end should not be touched by the sort() method.

After sorting: [6, 15, 28, 34, 74, 90, 0, 0, 0, 0, 0]

Step 5. Add a private helper method to insert a new item into the partially filled array. Do not add the new item to the end of the filled area and then sort. Insert and maintain the sorted order.

// insert into a partially filled array ([0~numOfElements-1]) and

// maintain sorted order (ascending)

// This method returns number of stored elements at the end.

// If insert failed (such as arr is null, numOfElements is invalid,

// or arr is already full), do not modify arr content and return

// numOfElements

private static int insert(int[] arr, int numOfElements, int newItem) {

// ADD code

}

Step 6. Add the following code to main() to test the insert method (continue from step 4 work):

Scanner stdIn = new Scanner(System.in);

int num;

for (int i=0; i<2; i++) {

System.out.print("Enter a number: ");

num = stdIn.nextInt();

size = insert(numArr, size, num);

System.out.printf("After inserting %d: %s\n", num, Arrays.toString(numArr));

}

stdIn.close(); // Eclipse requires closing a Scanner object

This segment of code will generate this output with two inputs -5 and 60:

Enter a number: *-5*

After inserting -5: [-5, 6, 15, 28, 34, 74, 90, 0, 0, 0, 0]

Enter a number: *60*

After inserting 60: [-5, 6, 15, 28, 34, 60, 74, 90, 0, 0, 0]

Now take a screenshot of your final work and include it in your assignment report.

1. Reflection: answer those questions AFTER you’ve completed this assignment:
2. What’s the hardest part of this assignment for you? Please explain.
3. Can you implement at least one search algorithm and one sort algorithm on your own without using any references? What may help if you’re yet to accomplish that?

## SUBMISSION

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

* Exercise 1: provide the java file and a screenshot of the execution result.
* Exercise 2: provide the java file and two required screenshots of execution.
* Exercise 3: Assignment reflection
* Check the completeness of your work against the rubric before turning it in.

## Rubric: Unit 3 Assignment

| **Criteria** | **Ratings** | | | **Pts** |
| --- | --- | --- | --- | --- |
| **Exercise 1 (SelectRectangular)** | 7 pts. Correct method meeting all requirements. | 6 ~ 1 pts. One or more steps incorrect:  (1 pts) handle invalid image parameter: null or zero rows/cols.  (1 pts) handle invalid start/end values  (2 pts) handle different valid start/end values  (2 pts) return a 2D array containing the required data given valid start/end values  (1 pts) at least five additional testing cases in main() as required. | 0 pts. All steps are incorrect or no submission. | 7 |
| **Exercise 2 (TestFileInputArr)** | 10 pts. Correct method meeting all requirements. | 9 ~ 1 pts. At least one requirement is incorrect or missing:  [3 pts] Step 1. toIntArray(): (1 pts) local int[] with required size; (1 pts) data copied to front of local int; (1 pts) return.  [3 pts] Step 3. sort(): (1 pts) arr not modified on invalid params; (1 pts) proper selection/insertion sort code; (1 pts) selection/insertion sort only involves the specified elements.  [3 pts] Step 5. insert():(1 pts) arr not modified and proper value returned on invalid params; (1 pts) proper insert code (doesn’t earn this pts if sorting); (1 pts) insertion only involves the specified elements and returns a proper value (doesn’t earn this point if sorting).    [1 pts] Step 2, 4, 6: required main() code and the program works as described. | 0 pts. Incorrect or missed all requirements; no submission | 10 |
| **Exercise 3. Reflection** | 1 pts. Answered both questions. | 0.5 pts. Answered only one question. | 0 pts. Didn’t answer the questions or 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 |