The University of Jordan
Faculty of Engineering and Technology
Department of Computer Engineering

Object-Oriented Problem Solving: CPE 342

Write a program that includes in its class the following methods (all methods in this assignment must be defined in the same class):

1. A method called *isThere* that takes an array of integer values and an integer value as parameters and returns true if the integer value exists in the array and false otherwise.

2. A method called *createArrayFromUser* that does not take parameters, it returns an array of integer values. The array is created by reading the array size, then values of array elements from the user.
   a. The method must prompt the user to enter array size and values with appropriate messages.
   b. The method must make sure that no entered value is repeated more than once. Invoke the method *isThere* to check whether an entered value already appears in the array. If so, prompt the user to enter another value with the following message "*You already entered this value, please enter another value!*".

3. A method called *createArrayRandom* that does not take parameters, it returns an array of integer values. The array is created by reading the array size, then two integer values *a* and *b* from the user. The array is formed by generating random integer numbers between the two numbers including the numbers themselves.
   a. The method must check which of the two values is greater than the other in order to generate the random numbers correctly.
   b. The method must make sure that no value is repeated more than once in the resulting array. Invoke the method *isThere* to check whether the selected value already appears in the array.

4. A method called *addNumbersIf* that takes an array of integer values, and a variable length argument list of integer values as parameters and returns an array of integer values. The returned array results from adding values from the variable length argument list to the passed array. However, values must be added only if they do not already appear in the passed array. Invoke the method *isThere* to check whether the value already appears in the array.

   For example: if the array 66, 25, 53, 17, 83 is passed to the first parameter, and the numbers 12 25 34 53 are passed to the variable length argument list, the method must return the array 61, 25, 53, 17, 83, 12, 34.

5. Write a method called *create2D* that does not take parameters and returns a two-dimensional array of integers. The method must prompt the user to enter the number of rows. Then for each row, it must prompt the user whether he wants to create the row by entering its values and thus invoke the method *createArrayFromUser* or create the row from random values and thus invoke the method *createArrayRandom*.

6. Write a method called *returnRandomRow* that takes a two-dimensional array and returns one of its rows randomly.

7. Write a method called *returnMaxAverageRow* that takes a two-dimensional array and returns the row who has the maximum average among the array rows.

8. Write a method called *shuffleRows* that takes a two-dimensional array and shuffle its rows. Note that this method must not return a new array, it must shuffle the passed array rows in its place.

9. In your *main* method:

Note: you must print the affected array after each step.
For that:

1. **Define one method that prints contents of a one-dimensional array to the console on one line space separated.**
2. **Define one method that prints contents of a two-dimensional array to the console each row on a line with its elements space separated.**

a. Create a two-dimensional array named *list2D* by invoking the method *create2D*.
b. Define a one-dimensional array named *list1D* with the integer values {10, 20, 30, 40}.
c. Invoke the method *addNumbersIf* to add three integers read from the user to the array *list1D*.
d. Add the array *list1D* to be the last row of the array *list2D*.
e. Print a row that is selected randomly from the array *list2D* by invoking the *returnRandomRow* method.
f. Print the row with the maximum average in the array *list2D* by invoking the *returnMaxAverageRow* method.
g. Shuffle rows of the array list2D by invoking the *shuffleRows* method.

Supplementary Material

- Study the following two implementations of the copy method which takes a two-dimensional array and returns a copy of it in two different ways, and understand the difference between the two implementations.

First implementation: rows of new array are the same as rows of the source array.

```
public static int [][] copy2D(int [][] source){
  int [][] dst = new int[source.length][];
  for(int i=0; i<source.length; i++)
     dst[i] = source[i];
  return dst;
}
```

Second implementation: rows of new array are copies of rows of the source array.

```
public static int [][] copy2D(int [][] source){
  int [][] dst = new int[source.length][];
  for(int i=0; i<source.length; i++){
     dst[i] = new int[source[i].length];
     for(int j=0; j<source[i].length; j++)
        dst[i][j] = source[i][j];
  }
  return dst;
}
```

Whenever you need to construct a 2-dimensional array from other 1-dimensional arrays or 2-dimensional arrays you can use one of these two approaches, depending on what your application requires. A third extreme approach does not create a copy of the source 2-dimensional array, but assigns the address of the source array to the reference variable of the destination array (dst = source).