Week 9 Lab 7-2

Do an exercise to reverse an array in place, one to return a new array with just the even *ints* in an *int* array, and one to create a flexible *int* array; for extra credit, do Chapter 7 Exercise **8** (create and test an ***Increasing*** class to check if a *double* array is strictly increasing); these exercises are worth 4, 4, and 6 points; the extra credit exercise is also worth 6 points.

A. Copy **Reverser.java** in the Chapter 7 **Source Code** folder, containing the *reverse* method, to **Reverser2.java** and write and test a ***reverse2()*** method:

1. *reverse2* should have the following header:
   public static void reverse2(int[] a)
2. *reverse2* should reverse the array *a* passed into the method in place, that is, it will not return a new array, but instead will replace the contents of the original array with its same items, but in reverse order.
3. ***Hints*: there are at least two possible ways to approach this:**
   - **Simple: Use *reverse()* to create a new array with the *a* array's elements in reverse order, and copy them into *a***
   - **Complex: Use a *for* loop as in *reverse()*, but only go through ½ of the *a* array, and swap elements from start to end (you might want to write a *swap()* method to help with this)**

Show me your program source code and how it works, or submit on Blackboard.

B. Write and test a **CopyEvens.java** program with a *copyEvens()* method that returns a new array containing just the even numbers in its input *int* array (if any):

1. *copyEvens* should have the following header:
   public static int[] copyEvens(int[] a)
2. *copyEvens* should go through the array *a* passed into the method to determine how many even integers are in *a*, then create a new array of that size and copy the even *int*s in *a* into the new array and return it.
   ***Hint*: use a separate counter variable to fill up the new array as you're going through the *a* array in a *for* loop.**
3. In addition, write a helper method *countEvens()* with this header:
   private static int countEvens(int[] a)
   that counts the number of even *int*s in *a* and returns that count. *copyEvens* should call the *countEvens()* method to determine the size of its returned array.

Show me or our TA your program source code and how it works, or submit on Blackboard.

C. Create and test a ***FlexArray*** class that grows an array as necessary:

- The class has one **private int[] array;** instance variable.
- The class has two constructors:
  - **public FlexArray(int size) // creates/sets *array*'s length**
  - **public FlexArray() // calls *this(10);* → sets length to 10**
- There is a method **private void assure(int size)** that makes sure *array* has that many elements by using ***expand()*** or ***Arrays.copyOf(array, size)***.
- There are two methods:
  - **public int get(int index) // returns the value at that index**
  - **public void set(int index, int value) // stores that value**

  Both of these methods must check that *index* is valid, and if not use *assure()* to make *index* valid (so that *array.length* >= *index+1*).
- There is a method that returns *array.length*: **public int length()**.
- Finally, **public void display(int length)** displays *length* elements from *array*, growing it if needed; **public void display()** shows all current ones. These are two overloaded *display()* methods.
- Write tests for these methods in *main* to show that *array* growth works.

Show me your program source code and how it works, or submit on Blackboard.

D. **Extra Credit:** Do Chapter 7 Exercise **8**: create and test an ***Increasing*** class containing this method:

public static boolean isStrictlyIncreasing(double[] in)

- ***isStrictlyIncreasing()*** returns *true* if each value in the given *double* array is greater than the value just before it, or *false* otherwise.
- ***Hint*: run a *for* loop starting at index 1 and compare each element to the element at the current index minus 1 – if the current element is less than or equal to the previous one, return *false*. After the *for* loop, return *true*.**
- Use the Java *array initializer* syntax to create a few *double* arrays in *main* and check that ***isStrictlyIncreasing()*** returns the correct *true*/*false* value for each. **You might want to write a *display()* method that will print *double* arrays.**
  Arrays you might try:
  - **new double[0]** or **{ 3.4 }** – length 0 or 1, should return *true*
  - **{ 1.0, 2.0 }** – should return *true*
  - **{ 2.0, 2.0 }** – should return *false*
  - any other larger *double* array …

Show me your program source code and how it works, or submit on Blackboard.