CSE 11 **START EARLY!**
Winter 2016
Program Assignment #2 (100 points)

Due: 30 January 2016 at 1159pm (2359, Pacific Standard Time)

PROGRAM #2: IntArray11

READ THE ENTIRE ASSIGNMENT BEFORE STARTING

In lecture, we described several common operations that can be performed on arrays. In this assignment you will implement a class, called `IntArray11`, that defines and implements these operations on integer arrays.

The goals of this assignment are:

1) to become comfortable with defining a class and using it
2) become more comfortable with reading javadoc-created documentation and using it to guide implementation
3) developing your own test code.

This is a more complicated program than #1, so it will require you to start to develop some of your own debugging skills. It is roughly double the length of your previous assignment. You will also code a program called `IntArray11Test` so that you can use your code.

TASK 1 : The `IntArray11` class

You are being provided a skeleton of the IntArray11 class that has been commented using Javadoc-style comments.

1. Download IntArray11.java
2. Generate the documentation using the javadoc command-line tool, to see the details of each constructor and public method. See **Week 3 lecture slide 48** for an example of how to run the javadoc command.
3. Go through the signature of all the methods defined in the Javadoc and carefully read method description to understand its operation on array.
4. Implement all methods correctly.
5. Put your name in the file after the existing Javadoc @author tag (replace YOUR NAME HERE)

When you Implement IntArray11, **You may NOT:**

a) Change the signature of any `public` method
b) Add any new `public` methods,
c) Add any `public` instance or class variables. All class/instance variables should declared `private`
d) Use any other pre-defined classes (e.g., Array, ArrayList,…) that provide similar functionality.

You are allowed to add any number of your own **`private`** methods and variables.

As you look at the `IntArray11` definition, it should be clear that:

a. **An instance of the class** must internally store an **array of `int`s**. This instance variable makes up a key component of the state of an `IntArray11` object.
b. The class must also **define methods that manipulate an instance's internally-stored array**.
c. You should also note, that in the skeleton, `boolean` methods **always return `false`.** That is **clearly incorrect** in the full implementation. Those return values in the skeleton are present so that the initial code will compile.

For clarity, we will list the constructors and methods defined in IntArray11 with a brief description.In grading your assignment, we will write our test programs to utilize only your implemented `IntArray11` class to test its functionality.

The following table illustrates a portion of the Javadoc that was generated after Step 2:

Method Summary

Methods

Modifier and Type	Method and Description
boolean	**append**(int element) Append element at the end of the array
boolean	**delete**(int index) Delete an element at index.
int[]	**getArray**() retrieve a copy of the stored Array
int	**getElement**(int index) get the Element at index
static int	**getNArrays**() Determine how many IntArray11 Objects have been constructed
int	**getNelem**() get the number of elements stored in the array
boolean	**insert**(int element) Insert an element at the beginning of the array
boolean	**insert**(int index, int element) Insert an element at index in the array
void	**reverse**() reverse the order of the elements in the array
boolean	**reverse**(int start, int end) reverse the order of the elements in the array from start to to end index
boolean	**setElement**(int index, int element) set the value of an element in the stored arra
boolean	**swap**(int index1, int index2) swap two elements in the array
java.lang.String	**toString**() Pretty Print -- Empty String "[]" else "[e1, e2, ..., en]"

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

From the table we can see that **methods that return `boolean`** should return `true` if method was successful, `false` if an error would occur. For example, if you attempted to swap two indices and either index is out of bounds, you should return `false`**.**

Note : Your `intArray11` implementation should never generate a runtime error. It must not print any error messages to the screen (though you will find such error messages useful during debugging)

Some hints for solving Task 1:

1. You may find an internal helper method that copies the contents of one array into another to be useful.
2. If the `insert` method is invoked, and your internal array is full, you should create a new internal array that is large enough to hold all the elements and the newly inserted element.

TASK 2 : The IntArray11Test Program

This is a java program and you must therefore define a `main()` method. You are to create a test program to utilize your `IntArray11` class. The program itself is "freeform" and will be graded by inspection by the readers. Your IntArray11Test program is intended to help you write your IntArray11 class.

The **only hard requirement of your IntArray11Test program is that it must perform at least one meaningful test (invocation) of each public IntArray11 method**. See the sample IntArray11Test program provided. It tests several methods:

1. Constructors
2. `getArray`() method
3. `getNArrays()` method

You should add to this program, using the comments as a guide. Notice how a logical test has been coded into a method. Good practice would be construct a logical test inside a method and then invoke the method from within main(). Feel free to simply add new tests to main() using the supplied program as a guide.

Don't forget to put your name in the file after the existing Javadoc @author tag (replace YOUR NAME HERE)

The goal of this part of the assignment is for you to practice writing code that helps you in debugging. A good test should have a known outcome that you can verify. For example in the test program supplied, the `testGetArray()` method uses the constructor to create an array with elements [1..100], it then gets a copy of the array and validates that each element was set as expected. You could use almost the identical code to write a test for `getElement()`.

Some hints for solving Task 2:

1. Build your IntArray11 class in stages. Define a method, then define a test for the

method in IntArray11Test.java. Once you are satisfied that the method you have just coded is correct, move onto the next one.

2. Utilize private methods as you need them. An internal (helper) method to copy the contents of one array to another might be useful
3. Do NOT assume that arrays of only a certain size will be constructed.
4. Properly indent your test code, but do not be overly concerned with comments. A line or two commenting what a test is trying to do is helpful.
5. If you write a test and it works the first time, change your IntArray11 code in some small way to so that it should NOT pass the test. The idea is to validate that your test is doing what you think it does. Don't forget to return your IntArray11 code to the proper state! For example, in the test, you could have the constructor generate [0 .. 99] (which is incorrect) and verify that the test fails.

Make Copies of your Program Files as you go along, If you make a big mistake you can go back to the previously working code

TASK 3 : Turning in your Program

YOU MUST SUBMIT FROM THE LAB MACHINES FOR THIS TO WORK. PLEASE VERIFY WELL BEFORE THE DEADLINE THAT YOU CAN TURNIN FILES

You will be using the "bundlePR2" program that will turn in the file

```
IntArray11Test.java IntArray11.java
```

No other files will be turned in and they **must be named exactly as above.** BundlePR2 uses the department's standard turnin program underneath.

To turn-in your program, you must be in the directory that has your source code and then you execute the following

```
$ /home/linux/ieng6/cs11wa/public/bin/bundlePR2
```

The output of the turnin should be similar to what you saw in your first programming assignment.

You can turn in your program multiple times. The turnin program will ask you if you want to overwrite a previously-turned in project. **ONLY THE LAST TURNIN IS USED!**

Don't forget to turn in your best version of the assignment.

Frequently asked questions

Q. Can I add extra output? No. We attempt to autograde your programs to the extent possible. Having extra output can cause you to lose points.

Q. What if my programs don't compile? Can I get partial credit? No. The bundle program will not allow you to turn in a program that does not compile.

Q. I know about ArrayList, can I use it? Absolutely NOT! That is a key part of the assignment.

Q. I know about other classes that easily provide similar functionality, can I use them? No. one of the purposes of this assignment is write and debug some common array methods.

Q. Can my IntArray11 class extend an existing class (using the extends keyword)? No.

Q. Will you grade program style? Yes. In particular, indentation should be proper, variable names should be sensible. We will also look for code clarity, too. Overly long or complex codes are frowned upon.

Q. I don't understand how you will grade IntArray11Test? The readers will inspect your code to see if looks like it does meaningful tests. We'll make sure that your code passes your tests. In other words, your testing code has to run properly.

Q. I don't understand how you will grade IntArray11? We will run our test program against your IntArray11 class (autograde) and the readers will inspect your code for clarity

START EARLY! ASK QUESTIONS