

Goals:

By the end of this activity you should be able to do the following:

- Understand the basics of instantiating arrays and assigning / accessing array elements.
- How to iterate through arrays using loops.

Description:

In this activity, you will create a class called Scores that will hold an array of numerical values and provide methods that allow users to interact with the Scores class.

Directions:

Don't forget to add your Javadoc comments for your classes, constructor, and methods in this activity.

Part 1: Scores – instance variable, constructor and method stubs

- Create a class called Scores.
 - Add an instance variable with the name `numbers` to your class that is an array of `int` values:

```
private int[] numbers;
```
 - Add a constructor that has a parameter declared as an array of `int` values.

```
public Scores(int[] numbersIn) {  
    }  
}
```
- Add method stubs for the following methods. **The first one is given; do the rest on your own.**
 - `findEvens`: no parameter, returns an array of `int` (all of the even-valued scores)

```
public int[] findEvens() {  
    return null;  
}
```

An array is an object, so `null` is a placeholder return.
 - `findOdds`: no parameter, returns an array of `ints` (all of the odd-valued scores)
 - `calculateAverage`: no parameters; returns a `double` (the average of all scores)
 - `toString`: no parameters; returns a `String` containing all scores
 - `toStringInReverse`: no parameters; returns a `String` containing all scores in reverse order


Compile Scores and run the following in interactions. **Do not continue until your program compiles and the following code runs without error in interactions.**

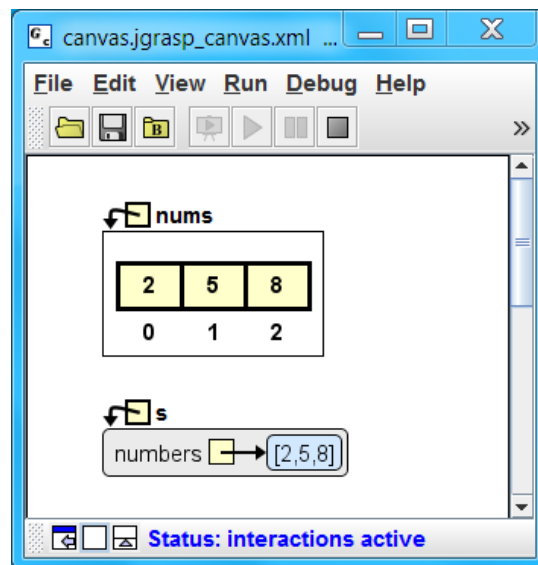
```
▶ Scores s = new Scores(null);  
▶ int[] evens = s.findEvens();  
▶ int[] odds = s.findOdds();  
▶ double avg = s.calculateAverage();
```

Part 2: Scores – completing the constructor

- In your constructor, add code that will set the value of `numbersIn` to `numbers`. **You access the entire array object using its variable name with no brackets.**
`numbers = _____;`
- Compile Scores. In the interactions pane set up an array of int values using an initializer list and send it to the constructor of scores:

```
▶ int [] nums = {2, 5, 8};  
▶ Scores s = new Scores(nums);
```

On the Workbench tab, click the Open New Viewer Canvas button , then drag the array `nums` and the Scores object `s` from the Workbench tab onto the canvas window. You should now be able to see the values of each of these items. To change the viewer for a variable on the canvas, select the viewer for the variable, then click the viewer menu button ▼ (at upper right of viewer frame), select “Viewer”, then select the desired viewer from the list. You should experiment with different viewers. Below, the “Presentation – Structure Identifier” viewer has been selected for `nums` and the “Basic” viewer has been selected for variable `s`. You should leave the canvas window open for the remainder of the activity since we will be adding other variables.



Canvas with int array `nums` and Scores `s`

Change the values in the int array `nums` using assignment statements in interactions as indicated below and you should see the viewer on the canvas updated as well. It should be obvious why the values in `nums` changed, but why did the int array field in the Scores object `s` also change?

```
▶ nums[0] = 9;  
▶ nums[1] = 8;  
▶ nums[2] = 7;
```

Part 3: Scores – toString and toStringInReverse methods

- The `toString` method will create a local `String` and then concatenate all of the values of numbers to the `String`.

```
public String toString() {  
    String result = "";  
    for (int i = 0; i < numbers.length; i++) {  
        L  
    }  
    return result;  
}
```

- The loop above iterates as the variable `i` ranges from 0 through the length of `numbers` - 1 and then terminates (or exits) when `i` is no longer less than the length of the array `numbers`. Within the for loop above, add the number at each index to the result:

```
result += numbers[i] + "\t";
```

- Check the `toString` return in interactions:

```
int[] nums = {2, 5, 8};  
Scores s = new Scores(nums);  
s  
2 5 8
```

- The `toStringInReverse` method will be exactly the same as `toString`, but will iterate from the length of `numbers` - 1 to 0.

```
public String toStringInReverse() {  
    String result = "";  
    for (int i = numbers.length - 1; _____; _____) {  
        result += numbers[i] + "\t";  
    }  
    return result;  
}
```

Compile `Scores` and run the following code in the interactions pane. **Do not continue until the following code runs without error in interactions.**

```
int[] nums = {2, 5, 8};  
Scores s = new Scores(nums);  
s.toStringInReverse()  
8 5 2
```

Part 4: Scores – findEvens and findOdds methods

- There are two parts to the `findEvens` method. First, count the number of evens in the array:

```
int numberEvens = 0;
for (int i = 0; i < numbers.length; i++) {
    if (numbers[ ] % 2 == 0) {
        numberEvens++;
    }
}
```

- You will then need to create an array with the appropriate length to store the number of even numbers.

```
int[] evens = new int[numberEvens];
```

- Add the even numbers to the `evens` array. In the following loop, `i` represents the current index of `numbers` and `count` is the current index of `evens`.

```
int count = 0;
for (int i = 0; i < numbers.length; i++) {
    if (numbers[ ] % 2 == 0) {
        evens[ ] = numbers[ ];
        count++;
    }
}
return evens;
```

- Compile `Scores` and test the return of `findEvens` by assigning it to an `int[] evens`. After the array `evens` has been assigned and it appears on the workbench, drag it onto your canvas window. The array does not have a `toString` method that includes the value at each index, so you will use a method from the `java.util.Arrays` class to display the array in interactions.

```
import java.util.Arrays;
int[] nums = {2, 5, 8, 1, 10};
Scores s = new Scores(nums);
int[] evens = s.findEvens();
evens // toString output of an array object (will vary)
[D@5abb7465
Arrays.toString(evens)
[2, 8, 10]
```

- Create the `findOdds` method on your own. It will perform the exact same function as `findEvens`, but it will find all odd numbers in the array (i.e., the numbers that are not divisible by 2). *Hint: the remainder when dividing by 2 is 1 rather than 0.*

- Test `findOdds` in the interactions pane. After the `int` array `odds` is created, be sure to drag it from the workbench to the canvas. **Do not continue until your output is correct.**

```

▶ import java.util.Arrays;
▶ int[] nums = {1, 5, 8, 3, 10};
▶ Scores s = new Scores(nums);
▶ int[] odds = s.findOdds();
  Arrays.toString(odds)
  [1, 5, 3]

```

Part 5: Scores - `calculateAverage` method

- First, find the sum of all values in the numbers array.

```

int sum = 0;

for (int i = 0; i < numbers.length; i++) {
    sum += numbers[i];
}

```

- Return the sum divided by the number of elements in the array. Remember that `sum` and `numbers.length` are both type `int` so you need to do a cast before you divide. For this activity, you can assume that `number.length` is not equal to zero.

```
return _____ / _____;
```

Compile `Scores` and run the following code in the interactions pane. Be sure to drag `avg` from the workbench onto the canvas

```

▶ int[] nums = {2, 5, 8, 7, 19};
▶ Scores s = new Scores(nums);
▶ double avg = s.calculateAverage();

```

Now invoke the `findEvens` and `findOdds` method, assigning the results to `int[] evens` and `int[] odds` respectively as shown below.

```

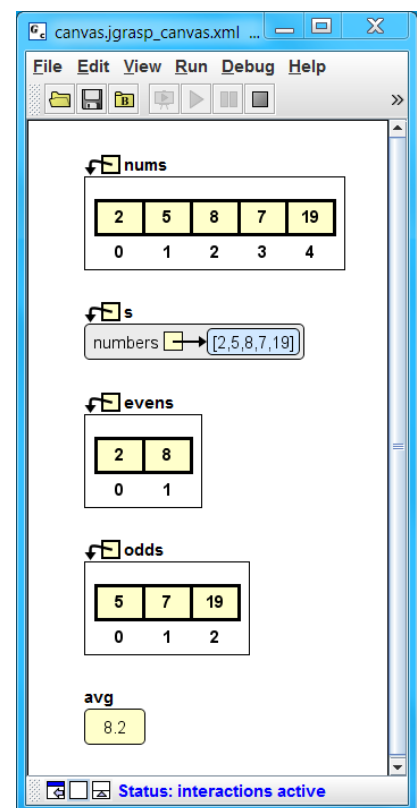
▶ int[] evens = s.findEvens();
▶ int[] odds = s.findOdds();

```

Your canvas should show all of the current results as indicated in the figure at right.

You should test all methods in the interactions pane with appropriate variables in the canvas window. You should ensure that your methods work with a different set of values for the `int` array `nums` than shown above.

Finally, submit your `Scores.java` file to Web-CAT.



The canvas after dragging variables onto canvas from the Workbench and invoking the methods `findEvens`, `findOdds`, and `calculateAverage` on the `Scores` object `s`.