https://runestone.academy/runestone/books/published/apcsareview/ArrayBasics/toctree.html

Answers are in ***Bold***

8.1 Arrays in Java

I've never messed much with arrays in Java so this chapter is likely gonna be a new experience for me. Arrays in Java are similar to lists in Python, at least at one dimension. To declare an array in Java you declare a variable like you normally would, but this time you place square brackets next to the variable type, like so: int[ ] highScores = null; **Then you need to create the array, this is done by using the new keyword and it initializes the size of the array.** highScores = new int[5]; **When you go to write to an array you just assign each index to a variable like** highScores[5] = 99; **If it's an int or float array its starting value will be zero, if it's a string array it will be null, and if it's boolean it will be false. To set array values upon creation you don't need to set a size and instead do this:** int[] highScores = {99, 98, 98, 88, 68};

Run Original - 1 of 1

```java
public class Test1
{
   public static void main(String[] args)
   {
     // declare arrays
     int[ ] highScores = null;
     String[ ] names = null;

     // create the arrays
     highScores = new int[5];
     names = new String[5];

     // print the initial values at index 0
     System.out.println(highScores[0]);
     System.out.println(names[0]);

     // set the values in the highScores array
     highScores[0] =  99;
     highScores[1] =  98;
     highScores[2] =  98;
     highScores[3] =  88;
     highScores[4] =  68;
     System.out.println(highScores[0]);

     // set the values in the names array
```

```
0
null
99
Jamal
```

Activity: 4 -- ActiveCode (array-set)

You can also get the length of an array using the .length **method. There are no parentheses on this one, unlike the string** .length() **method,**

Run Original - 1 of 1

```java
public class Test2
{
   public static void main(String[] args)
   {
     int[ ] highScores = {99,98,98,88,68};
     System.out.println(highScores.length);
   }
}
```

```
5
```

Activity: 5 -- ActiveCode (lcab2)

8.1 Check your Understanding

7-1-3: Click on the values at index 1 and 3 in the following array.

3	2	1	-3

Check Me

You are Correct!

Activity: 7 -- Clickable (arrayClick1)

7-1-4: What index is the first element in an array at?

- (selected) A. 0
- B. 1

Check Me Compare me

✓ The index is really telling the computer how far the item is from the front of the array. So the first element in an array is at index 0.

Activity: 8 -- Multiple Choice (qab_1)

7-1-5: Click on the values at index 0 and 2 in the following array.

4	-2	8	7

Check Me

You are Correct!

Activity: 9 -- Clickable (arrayClick2)

7-1-6: Which index is the last element in an array called `highScores` at?

- A. `highScores.length`
- (selected) B. `highScores.length - 1`

Check Me Compare me

✓ Since the first element in an array is at index 0 the last element is the length minus 1.

Activity: 10 -- Multiple Choice (qab_2)

8.2 Looping with a For-Each Loop

A for each loop can only be used on a collection of items like an array. It can only do this because it will look for the length of the collection, and as it does this it assigns an initialization value to whatever item is in the dataset at that index. Syntax for this is like this: for(initialization : datasetBeingUsed) { }

Run

Original - 1 of 1

```java
public class Test1
{
   public static double getAvg(int[] values)
   {
     double total = 0;
     for (int val : values)
     {
       total  = total + val;
     }
     return total / values.length;
   }

   public static void main(String[] args)
   {
     int[ ] values = {2, 6, 7, 12, 5};
     System.out.println(getAvg(values));
   }
}
```

```
6.4
```

Activity: 1 -- ActiveCode (lcaf1)

7-2-1: What are some of the reasons you would use a for-each loop instead of a for loop?

```
I: If you wish to access every element of an array.
II: If you wish to modify elements of the array.
III: If you wish to refer to elements through a variable name instead of an arr
ay index.
```

- A. Only I.
- B. I and III only.
- C. II and III only.
- D. All of the Above.

Check Me Compare me

✓ Correct! For-each loops access all elements and enable users to use a variable name to refer to array elements, but do not allow users to modify elements directly.

Activity: 2 -- Multiple Choice (qab_6A)

8.2 Check your Understanding

7-2-2: The following method has the correct code to return the largest value in an integer array called *vals* (a field of the current object), but the code is mixed up. Drag the blocks from the left into the correct order on the right and indent them correctly as well. You will be told if any of the blocks are in the wrong order or not indented correctly.

Drag from here

Drop blocks here

```
4 public int getLargest()
  {
6   int largest = vals[0];
5   for (int item : vals)
    {
3     if (item > largest)
      {
8       largest = item;
2     }  // end if
7   } // end for
    return largest;
1 } // end method
```

Check Me Reset Help Me

Perfect! It took you only one try to solve this. Great job!

Activity: 4 -- Parsons (pab_2)

7-2-3: Given that `a` is an array of integers and `val` is an integer value, which of the following best describes the conditions under which the following code segment will return true?

```
boolean temp = false;
for ( int i = 0; i < a.length; i++)
{
  temp = ( a[i] == val );
}
return temp;
```

- A. Whenever the first element in *a* is equal to *val*.
- B. Whenever *a* contains any element which equals *val*.
- C. Whenever the last element in *a* is equal to *val*.
- D. Whenever only 1 element in *a* is equal to *val*.

Check Me Compare me

✓ The variable *temp* is assigned to the result of checking if the current element in the array is equal to *val*. The last time through the loop it will check if the last element is equal to *val*.

Activity: 5 -- Multiple Choice (qab_3)

7-2-4: Given the following field and method, which of the following best describes the contents of `myStuff` after (`int m = mystery(n);`) has been executed?

```
// private field in the class
private int[ ] myStuff;

//precondition: myStuff contains
//  integers in no particular order
public int mystery(int num)
{
   for (int k = myStuff.length - 1; k >= 0; k--)
   {
       if (myStuff[k] < num)
       {
          return k;
       }
   }

   return -1;
 }
```

- A. All values in positions *m+1* through *myStuff.length-1* are greater than or equal to *n*.
- B. All values in position 0 through *m* are less than *n*.
- C. All values in position *m+1* through *myStuff.length-1* are less than *n*.
- D. The smallest value is at position *m*.

Check Me Compare me

✓ Mystery steps backwards through the array until the first value less than the passed num (*n*) is found and then it returns the index where this value is found.

Activity: 6 -- Multiple Choice (qab_4)

7-2-5: Given the following code segment, which of the following will cause an infinite loop? Assume that `temp` is an int variable initialized to be greater than zero and that `a` is an array of integers.

```
for ( int k = 0; k < a.length; k++ )
{
   while ( a[ k ] < temp )
   {
      a[ k ] *= 2;
   }
}
```

- A. The values don't matter this will always cause an infinite loop.
- B. Whenever *a* includes a value that is less than or equal to zero.
- C. Whenever *a* has values larger then *temp*.
- D. When all values in *a* are larger than *temp*.
- E. Whenever *a* includes a value equal to *temp*.

Check Me Compare me

✓ When *a* contains a value that is less than or equal to zero then multiplying that value by 2 will never make the result larger than the *temp* value (which was set to some value > 0), so an infinite loop will occur.

Activity: 7 -- Multiple Choice (qab_5)

8.3 Using the For Loop to Loop Through an Array

A standard For Loop can be used to iterate through an array as well by starting at 0 and continuing while the initialization is less than the length of the array.

Run

Original - 1 of 1

```java
public class ArrayWorker
{
   private int[ ] values;

   public ArrayWorker(int[] theValues)
   {
      values = theValues;
   }

   public void multAll(int amt)
   {
     for (int i = 0; i < values.length; i++)
     {
       values[i] = values[i] * amt;
     } // end for loop
   } // end method

   public void printValues()
   {
     for (int val : values )
     {
       System.out.println(val);
     }
   }

   public static void main(String[] args)
```

```
4
12
14
24
10
```

7-3-1: The following method has the correct code to subtract amt from all the values in the array **values** (a field of the current object), but the code is mixed up. Drag the blocks from the left into the correct order on the right and indent them correctly. You will be told if any of the blocks are in the wrong order or not indented correctly.

Drag from here

Drop blocks here

```
4  public void subAll(int amt)
   {
3    for (int i = 0;
             i < values.length;
             i++)
     {
5      values[i] = values[i] - amt;
1    } // end for loop
2  } // end method
```

Check Me Reset Help Me

Perfect! It took you only one try to solve this. Great job!

Activity: 2 -- Parsons (pab_1r)

8.4 Looping from Back to Front

You can loop from the back to the front of an array by starting at the length of the array -1 and then iterating while it is greater than or equal to 0. for(int index = values.length - 1; index >= 0; index - -)

Run Original - 1 of 1

```
  }

  public void printValues()
  {
    for (int val : values )
    {
      System.out.print(val + ", ");
    }
    System.out.println();
  }

  public static void main (String[] args)
  {
     int[] theArray = {-30, -5, 8, 23, 46};
     ArrayWorker worker = new ArrayWorker(theArray);
     System.out.println(worker.getIndexLastSmaller(50));
     System.out.println(worker.getIndexLastSmaller(30));
     System.out.println(worker.getIndexLastSmaller(10));
     System.out.println(worker.getIndexLastSmaller(0));
     System.out.println(worker.getIndexLastSmaller(-20));
     System.out.println(worker.getIndexLastSmaller(-30));
  }
}
```

```
4
3
2
1
0
-1
```

Activity: 1 -- ActiveCode (lcbf1)

7-4-1: Given the following code segment what will be returned when you execute: getIndexLastSmaller(-13);

```
private int[ ] values = {-20, -15, 2, 8, 16, 33};

public int getIndexLastSmaller(int compare)
{
   for (int i = values.length - 1; i >=0; i--)
   {
      if (values[i] < compare) return i;
   }
   return -1; // to show none found
}
```

- A. -1
- B. -15
- C. 1
- D. You will get an out of bounds error.

Check Me Compare me

✓ Since the method loops from the back towards the front -15 is the last value in the array that is less than -13 and it is at index 1.

Activity: 2 -- Multiple Choice (qab_6)

7-4-2: Given the following code segment what will be returned when you execute: getIndexLastSmaller(7);

```
private int[ ] values = {-20, -15, 2, 8, 16, 33};

public int getIndexLastSmaller(int compare)
{
   for (int i = values.length; i >=0; i--)
   {
      if (values[i] < compare) return i;
   }
   return -1; // to show none found
}
```

- A. -1
- B. 1
- C. 2
- D. You will get an out of bounds error.

Check Me Compare me

✓ You can not start the index at the length of the array. You must start at the length of the array minus one. This is a common mistake.

Activity: 3 -- Multiple Choice (qab_7)

8.5 Looping through Part of an Array

You can loop through part of an array using complex conditionals such as I = 0; I < values.length && I < 5; i++ **which will loop through the array from index 0 to index 5.**

Run Original - 1 of 1

```
  public void doubleFirstFive()
  {
    for (int i = 0; i < values.length && i < 5; i++)
    {
      values[i] = values[i] * 2;
    }
  }

  public void printArray()
  {
     for (int val: values)
     {
        System.out.println(val);
     }
  }

  public static void main(String[] args)
  {
    int[] numArray = {3, 8, -3, 2, 20, 5, 33, 1};
    ArrayWorker worker = new ArrayWorker(numArray);
    worker.doubleFirstFive();
    worker.printArray();
  }
}
```

```
6
16
-6
4
40
5
33
1
```

Activity: 1 -- ActiveCode (lclp1)

7-5-1: Given the following values of a and the method doubleLast what will the values of a be after you execute: doubleLast()?

```
private int[ ] a = {-20, -15, 2, 8, 16, 33};

public void doubleLast()
{

   for (int i = a.length / 2; i < a.length; i++)
   {
      a[i] = a[i] * 2;
   }
}
```

- A. {-40, -30, 4, 16, 32, 66}
- B. {-40, -30, 4, 8, 16, 32}
- C. {-20, -15, 2, 16, 32, 66}
- D. {-20, -15, 2, 8, 16, 33}

Check Me Compare me

✓ It loops from the middle to the end doubling each value. Since there are 6 elements it will start at index 3.

Activity: 3 -- Multiple Choice (qab_8)

7-5-2: Given the following values of a and the method mystery what will the values of a be after you execute: mystery()?

```
private int[ ] a = {-20, -15, 2, 8, 16, 33};

public void mystery()
{

   for (int i = 0; i < a.length/2; i+=2)
   {
      a[i] = a[i] * 2;
   }
}
```

- A. {-40, -30, 4, 16, 32, 66}
- B. {-40, -30, 4, 8, 16, 33}
- C. {-20, -15, 2, 16, 32, 66}
- D. {-40, -15, 4, 8, 16, 33}
- E. {-40, -15, 4, 8, 32, 33}

Check Me Compare me

✓ This loops from the beginning to the middle and doubles every other element (i+=2 is the same as i = i + 2).

Activity: 4 -- Multiple Choice (qab_9)

8.6 Things to Watch for when Looping Through an Array

- **Make sure to start at index 0 and remember that the largest index is array.length - 1**
- **Remember that the method header type should be the same type as the return value you use such as an int method should return an int type**

8.7 Common Mistakes

- **Forgetting to create the array - only declaring it** int[ ] nums;
- **Using 1 as the first index, not 0**
- **Using** array.length **as the last valid index in an array, not** array.length - 1
- **Using** array.length( ) **instead of** array.length **(Not penalized in the free response on exam)**
- **Using** array.get(0) **instead of** array[0] **(Not penalized in the free response on exam)**
- **Going out of bounds hen looping through an array**
- **Jumping out of the loop too early by using return before the loop has finished**