

★ Find and describe my errors! Identify run-time/compiler errors

### Quiz#1: One-Dimensional Arrays in Java

Name: ERROR KEY

1. Declare a one-dimensional array of String data named `fruit`, and store the values "Apple", "Banana", "Orange", and "Pear"

a. Using one statement:

b. Using multiple statements (five total)

c. Use a regular "for" loop to print the contents of `fruit`

```
for(int i = 0; i <= fruit.length ; i++)  
    System.out.println(fruit);
```

*fruit[i]*

d. Use a "for-each" loop to print the contents of `fruit`

```
for (int e: fruit)  
    System.out.println(fruit[e]);
```

2. Write a method that will find and return the length of the shortest String in an array of String data. For example, if `words` contains "Hello", "Hi", "Greetings", and "Hey", the method would return 2, the number of letters in "Hi."

```
public int findShortestWord(String[] words)  
{
```

```
    int minLen = 0; = words[0].length;
```

```
    for(int i = 0; i < words.length; i++)
```

```
        if(minLen > words[i].length)
```

```
            minLen = words[i].length;
```

```
    return minLen;
```

```
}
```

3. Complete the methods below, according to the directions in the comments for each method.

- a. // print the last half of a one-dimensional array. So if nums contains 1,3,5,6,2,3, the method prints 6,2,3  
// **Precondition:** the array has an even number of elements

```
public void printSecondHalf(int[] nums)
{
    for (int i = 0; i < nums.length; i++)
        if (nums.length % 2 == 0)
            System.out.println(nums[i + (nums.length/2)]);
}
```

- b. // print the contents of a one-dimensional array backwards (in reverse order)

```
public void printBackwards(double[] nums)
{
    for (int i = nums.length; i >= 0; i--)
        System.out.println(nums[i]);
}
```

- c. // find and return the minimum value in an array

```
public double findMin(double[] nums)
{
    int min = 99999; Double.MAX_INDEX;
    for(double e: nums)
        if (e < min)
            min = e;
    return min;
}
```

- d. // find and return the index of the maximum value in an array

```
public int findMaxIndex(double[] nums)
{
    double maxIndex = 0;
    for(int i = 1; i < nums.length; i++)
        if(nums[i] > maxIndex) nums[maxIndex]
            maxIndex = i;
    return maxIndex;
}
```

- e. /\* Write a method that will count the number of even integers in an array. So if an array contains  
\* 2, 3, 4, 3, 1, 6, 7, 8 then the method will return 4.  
\*/

```
public int countEvens(int[] nums) {  
    int amount = 0;  
    for(int e: nums)  
        if (e % 2 == 0)  
            amount++;  
    System.out.println(amount);  
}
```

- f. /\* Search the contents for a target String in an array, and return the index of that  
\* target String. For example, if an array names topics contains "Who", "What", "When" and "Where"  
\* and the target value is "When" then searchContents(topics, "When") will return 2.  
\* If the target is not found, return -1  
\*/

```
public int searchContents(String[] words, String target)  
{  
    int index = -1;  
    for(String e: words)  
        if (e == target)  
            index = e; accesses element  
    return index;  
}
```

*e. equals(target)*

- g. // Return the sum of the numbers in an array. Except the number 13 is very unlucky, so it does not count  
// and numbers that come immediately after a 13 also do not count. **Precondition:** the array length > 0

sum13([1, 13, 2, 1]) → 2  
sum13([1, 1, 3, 6]) → 11  
sum13([1, 2, 2, 1, 13]) → 6

```
public int sum13(int[] nums) {  
    int sum = 0;  
    for (int e: nums)  
    {  
        if (e != 13)  
            sum += e;  
        else if (e == 13)  
            return sum;  
    }  
    return sum;  
}
```

}