Quiz 3 Results

- 20 Points possible
- Max:
- Min:
- Mean:
- Median:

Chapter 5: Arrays

- We have seen how to declare array:
  ```java
  int[] quizScore;  // declares an array of ints
  double[] temperature;  // declares an array of doubles
  String[] lastName;  // declares an array of Strings
  ```
  – This does not create the array elements
  – You must allocate the array elements before using the array.

Declaring the array reference

```java
int[] quizScore;
```

Allocating Array Elements

- We also seen how to allocate the array elements
  ```java
  quizScore = new int[65];
  temperature = new double[count];
  ```
  – This allocates the array elements, initializes their values, and assigns the array reference
  – The size of the array is set when it is allocated – you can’t change it after it has been set.
  - But, you can destroy and create a new array using the same name
Creating the array

int[] quizScore;

quizScore = new int[65];

0
1
2
3
...
64

Single Statement

- You can create the array reference and the array in one statement:

  ```java
  int[] quizScore = new int[65];
  double[] temperature = new double[count];
  boolean[] isFemale = new boolean[100];
  ```

Accessing Array Elements

- Individual array elements are accessed by subscripting or indexing.
- The subscript value ranges from 0 to the array length – 1
- The subscript is an integer expression

Array Elements and Assignment

- Use subscripting to assign a value to an array element:

  ```java
  quizScore[10] = 45;
  temperature[13] = 12.6;
  int[] example = new int[10];
  for (int i = 0; i < 10; i++) {
      example[i] = i * 2;
  }
  ```
Accessing Array Elements

• Use subscripting to access the value of an array element:

```java
System.out.print("quizScore[10] = " + quizScore[10]);
if (temperature[13] <= 32.0) {
    // Do something here
}
for (int i = 0; i < 10; i++) {
    System.out.println(i + " " + example[i]);
}
```

Initializing Arrays

• You can create the array reference, allocate the array, and initialize it in one statement

```java
double[] data = {5.1, 3.7, 6.2, 10.3, 4.5};
int[] numDaysPerMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Array Initialization

double[] data = {5.1, 3.7, 6.2, 10.3, 4.5};

---

Initializing Arrays: Example

```java
// ArraySum.java - sum the elements in an array and compute their average
class ArraySum {
    public static void main(String[] args) {
        int[] data = {11, 12, 13, 14, 15, 16, 17};
        int sum = 0;
        double average;
        for (int i = 0; i < 7; i++) {
            sum = sum + data[i];
            System.out.print(data[i] + " ");
        }
        average = sum / 7.0;
        System.out.println("sum = " + sum + " average = " + average);
    }
}
```
**Array index**

- The value of the array index expression must be an integer
- The value of the array index must be in the range 0 to array length – 1

**Array Index: Example**

// ReverseArray.java: One-dimensional array
import tio.*;

class ReverseArray {
    public static void main(String[] args) {
        int[] simpleArray = new int[5];
        System.out.println("Enter 5 integers.");
        for(int i = 0; i < 5; i++)
            simpleArray[i] = Console.in.readInt();
        System.out.println("In reverse they are:");
        for(int i = 0; i < 5; i++)
            System.out.println(simpleArray[4-i]);
    }
}

**Array length**

- The length of an array is important
- This information is stored with the array
  - access with <arrayName>.length

  quizScore.length // 65
  temperature.length // count
  isFemale.length // 100

**Array length**

- An example:

  int[] example = new int[10];
  for (int i = 0; i < example.length; i++) {
      System.out.println(i + " " + example[i]);
  }

- This style is preferred over one with literals
  - Much easier to change your program if you decide
    you need to change the array size
ReverseArray.java revisited

```java
import tio.*;

class ReverseArray2 {
    public static void main(String[] args) {
        int[] simpleArray = new int[5];
        System.out.println("Enter " + simpleArray.length + " integers.");
        for(int i = 0; i < simpleArray.length; i++)
            simpleArray[i] = Console.in.readInt();
        System.out.println("In reverse they are:");
        for(int i = 0; i < simpleArray.length; i++)
            System.out.println(simpleArray[(simpleArray.length-1) - i]);
    }
}
```

An Exercise

- What if we wanted to have the user enter the size of the array?
  - Modify ReverseArray2.java to prompt the user for the array size

A Common Error

- Indexing from 1 to length
  - If you try to access outside this range, Java generates an IndexOutOfBoundsException

```java
int[] example = new int[10];
for ( int i = 1; i < example.length; i++ ) {
    System.out.println(i + " * " + example[i]);
}
```

Another Common Error

- Forgetting to allocate the array

```java
int[] example;
...
for ( int i = 0; i < example.length; i++ ) {
    System.out.println(i + " * " + example[i]);
}
```
Passing Arrays to Methods

- Works exactly the same way as with other types

```java
int[] example;
...
someMethod(example);
...
static void someMethod(int[] e) {
  ...
}
```

What is passed to the method?

- Recall that the array definition is really a reference to the array
  - For example,
    ```java
    int[] example;
    ```
    declares a reference to an array of int.
  - So when you call a method
    ```java
    someMethod(example)
    ```
    you are really passing an array reference to the method.

Passing Arrays to Methods

```java
// ArraySum2.java - sum the elements in an array
// using a method
class ArraySum2 {
  public static void main(String[] args) {
    int[] data1 = {1, 2, 3, 4, 5, 6, 7};
    int[] data2 = {16, 18, 77};
    System.out.println("data1: " + sum(data1));
    System.out.println("data2: " + sum(data2));
  }
  // sum the elements in an array
  static int sum(int[] a) {
    int sum = 0;
    for (int i = 0; i < a.length; i++)
      sum = sum + a[i];
    return sum;
  }
}
```

What is passed to the method?

- So, the array reference is copied from the actual parameter to the formal parameter
- BUT,
  - The array contents are not copied.
- Why?
  - Arrays can be very large
  - Copying large arrays is time consuming and inefficient
Passing Arrays to Methods

double[] data =
{5.1, 3.7, 6.2, 10.3, 4.5};

someMethod(data);