Quiz 4 Results

- 20 Points possible
- Min: <8
- Max: 20
- Mean: 14.1
- Median: 15

Quiz Portion of Class Grade

- Remember
  - You need to get 50% of the total quiz score to pass this class
  - You won’t necessarily pass, but this is a minimum standard
  - If you currently have less than 40 points total on your quizzes, I strongly recommend that you come to see me to discuss your situation

Question 1

- Is the following code fragment legal? If so, what does it print?

```java
int[] a1 = new int[5]; // int[3]
int[] a2 = new int[7]; // int[9]
a1 = a2;
System.out.println(a1.length);
```

- Yes, it’s legal
  - 7 or 9
Question 2 or 3

• Write a single Java statement that declares $x$ to be an array of double, allocates an array with 10 elements of type double, and assigns $x$ to refer to the array.
  ```java
double[] x = new double[10];
```

• Write a single Java statement that declares $b$ to be an array of boolean, allocates an array with 20 elements of type boolean, and assigns $b$ to refer to the array.
  ```java
boolean[] b = new boolean[20];
```

Question 3 or 2

• True or false: In general, an algorithm that runs in $O(n)$ time is faster than one that runs in $O(\log(n))$ time.
  Answer: False

Question 4 or 6

• What is printed by the following program?
  ```java
class Quiz4P4 {
    public static void main( String[] args ) {
      String[][] greek = {"alpha","beta","gamma"},
                        {"delta","epsilon","zeta"},
                        {"eta","theta"},{"iota"};
      System.out.println( greek.length );
      System.out.println( greek[1].length ); //or [2]
      System.out.println( greek[2][1] ); //or [1][2]
    }
  }
```
  **Answer:**
  - greek.length: 4
  - greek[1][2]: zeta
  - greek[1].length: 3
  - greek[2][1]: theta
  - greek[2].length: 2

Question 5 or 4

• What is printed by the following program?
  ```java
class Quiz4P5 {
    public static void main( String[] args ) {
      int array[] = {1,2,3,4}; // or {5,6,7,8}
      mystery( array );
    }
    static void mystery( int[] a ) {
      for (int i = a.length-1; i >= 0; i-- )
        System.out.print(a[i] + " ");
      System.out.println();
    }
  }
```
  **Answer:**
  4 3 2 1 or 8 7 6 5
class Quiz4P6 {
    public static void main( String[] args ) {
        int[] x = {3, 5, 7};
        int[] y = {10, 20, 30};
        mystery(x, y);
    }
    static void mystery( int[] x, int[] y ) {
        x = y;
    }
}

• Answer:
  x[1] = 20
  y[1] = 20
  x[1] = 5
  y[1] = 20

  • Or:
    x[2] = 30
    y[2] = 30
    x[2] = 7
    y[2] = 30

Question 7

• Write a method named **findMin** (**findMax**) that takes an array of double as a parameter and returns the index of the element that contains the smallest (largest) value in the array.

```java
static int findMin( double[] array ) {
    int minIndex = 0;
    for (int i = 0; i < array.length; i++ ) {
        if ( array[i] < array[minIndex] ) {
            minIndex = i;
        }
    }
    return minIndex;
}
```

```java
static int findMax( double[] array ) {
    int maxIndex = 0;
    for (int i = 0; i < array.length; i++ ) {
        if ( array[i] > array[maxIndex] ) {
            maxIndex = i;
        }
    }
    return maxIndex;
}
```

Question 6 or 5

```java
class Quiz4P6 {
    public static void main( String[] args ) {
        int[] x = {3, 5, 7};
        int[] y = {10, 20, 30};
        mystery(x, y);
    }
        static void mystery( int[] x, int[] y ) {
            x = y;
        }
}
```

• Answer:
  x[1] = 20
  y[1] = 20
  x[1] = 5
  y[1] = 20

  • Or:
    x[2] = 30
    y[2] = 30
    x[2] = 7
    y[2] = 30

Question 7

• Write a method named **findMin** (**findMax**) that takes an array of double as a parameter and returns the index of the element that contains the smallest (largest) value in the array.

```java
static int findMin( double[] array ) {
    int minIndex = 0;
    for (int i = 0; i < array.length; i++ ) {
        if ( array[i] < array[minIndex] ) {
            minIndex = i;
        }
    }
    return minIndex;
}
```

```java
static int findMax( double[] array ) {
    int maxIndex = 0;
    for (int i = 0; i < array.length; i++ ) {
        if ( array[i] > array[maxIndex] ) {
            maxIndex = i;
        }
    }
    return maxIndex;
}
```

Homework 8

• Any Questions?
Homework 8
• Scoring
  ~ Triples
    • must be scored on a single roll
    • 3 1’s scores 1000
    • 3 2’s scores 200
    • 3 3’s scores 300
    • 3 4’s scores 400
    • 3 5’s scores 500
    • 3 6’s scores 600
  ~ Singletons:
    • each 5 scores 50
    • each 1 scores 100

Homework 8
• You need a method to determine the score of the dice selected by the player
• For example
  - static int score( int[] selectedDice )
• How are you going to test this method?
  —What are the cases that you need to check?

Example: Counter
• We often want to count things, why not create an abstraction for doing it?
  — Advantage: you can reuse it in different places in the program, or even in other programs
• Data:
  —Current value of the counter (initially zero)
• Operations:
  —Reset, Increment, Decrement, Get the current value

CounterTest.java
```java
class CounterTest {
    public static void main( String[] args ) {
        Counter countThis = new Counter();
        Counter countThat = new Counter();
        countThis.increment();
        countThis.increment();
        countThat.increment();
        System.out.println("countThis: " + countThis.get() );
        System.out.println("countThat: " + countThat.get() );
    }
}
```
Counter.java

class Counter {
  int value;
  void reset() { value = 0; }
  void increment() { value++;
  }
  void decrement() { value--; }
  int get() { return value; }
}

• No main() method
• value is an instance variable
• Instance methods
  – no static keyword

Objects in Memory

Important Details

• Each Counter object has its own copy of the member variables
  – In this case, the integer variable called value
• When the methods are called, the call is of the form <objectname>.<methodname>()
• The object itself is an implicit parameter to the method, so that any references to the data access that object’s copy of the instance variables

Abstract Data Type

• Counter is an example of an Abstract Data Type (ADT) – an abstraction representing a particular type of data
• Classes allow us to implement ADTs
  – The data and methods combine to implement the functionality we desire or expect for this type of data
  – The implementation details are hidden from the user
  – The implementation is all in one place
  – The type can be used in many different places in the program or in many programs
Package

• In general, each class is in a separate file
  – The name of the file should match the class name
    (with .java at the end)
• All classes in the same directory are part of the
  same package
• Whether or not a method is in the same class or
  package as the data or method it is accessing
  affects what it can see and do.

Data Hiding

• It is desirable to hide the inner details of a class
  (ADT) from the users of the class
• We want to be able to determine the correctness
  of our class without having to examine the entire
  program that it is used in
• For example, with our Counter class, we want to
  ensure that the value doesn’t change by more
  than 1.

Data Hiding

• Accessing instance variables from outside the
  class violates the data hiding principle

```
class CounterTest2 {
    public static void main( String[] args ) {
        Counter c = new Counter();
        c.value = 150;
        System.out.println( "c.value = " + c.get() );
    }
}
```

A Better Counter Class

• Use the `private` and `public` access specifiers to
  control access to instance variables and methods

```
class Counter {
    private int value;
    public void reset() { value = 0; }
    public void increment() { value++; }
    public void decrement() { value--; }
    public int get() { return value; }
}
```
Public/Private/Default

• private methods/fields cannot be accessed from outside of the class
• public methods/fields can be accessed from anywhere
• default (no specifier) methods/fields have package access – they can be accessed from other classes in the same package
  – if you don't specify a package (see section 12.11), all classes in the same directory are part of the same default, un-named package