Unit 06 Array FRQ – SOLUTION

1. An array of `String` objects, `words`, has been properly declared and initialized. Each element of `words` contains a `String` consisting of at least 3 lowercase letters (a–z).

(a) Write a code segment that uses an enhanced for loop to print all elements of words that end with `"ing"`. As an example, if `words` contains: `{"ten", "fading", "post", "card", "thunder", "hinge", "trailing", "batting"}`, then the following output should be produced by the code segment.

```
fading
trailing
batting
```

Write the code segment as described above. The code segment must use an ***enhanced for loop*** to earn full credit.

```
for (String str : words) {
    if (str.substring(str.length() - 3).equals("ing")) {
        System.out.println(str);
    }
}

// Total: 3
//   +1 [Skill 3.D] Traverses all elements of words using an enhanced for loop
//   +1 [Skill 3.A] Identifies a String that ends in "ing"
//   +1 [Skill 3.C] Prints all and only String objects that end in "ing"
//   -1 (w) Extraneous code that causes side-effect
 (e.g., printing to output, incorrect precondition check)
//   -1 (x) Local variables used but none declared
```

2. Employees at a store are paid daily wages according to the following rules:
   Each employee is paid the same fixed amount per day.
   Each employee is paid an additional amount for each item they sold on that day.
   Daily Bonus: If the number of items sold that day by an employee is greater than a computed threshold, then the employee also receives a bonus equal to *10 percent* of the employee's daily wages.

You will write two methods in the `Payroll` class below.

```
public class Payroll {
    private int[] itemsSold; // number of items sold by each employee
    private double[] wages; // wages to be computed in part (b)

    /** Returns the bonus threshold as described in part (a).
     */
    public double computeBonusThreshold() {
        /* To be implemented in part (a) */
    }

    /** Computes employee wages as described in part (b)
     * and stores them in wages.
     * The parameter fixedWage represents the fixed amount
     * each employee is paid per day.
     * The parameter perItemWage represents the amount each
     * employee is paid per item sold.
     */
    public void computeWages(double fixedWage, double perItemWage) {
        /* To be implemented in part (b) */
    }

    // Other instance variables, constructors, and methods not shown.
}
```

The bonus threshold is calculated based on the number of items sold by all employees on a given day. The employee with the greatest number of sales and the employee with the least number of sales on that day are ignored in the calculation. The average number of items sold by the remaining employees on that day is computed, and this value is used as the bonus threshold.

For a given day, the number of items sold by each employee is stored in the array `itemsSold`. The example below shows the contents of `itemsSold` for a day in which there were ten employees. Each array index represents an individual employee. For example, `itemsSold[3]` represents the number of items sold by employee 3.

	0	1	2	3	4	5	6	7	8	9
`itemsSold`	48	50	37	62	38	70	55	37	64	60

Based on the information in the table, the bonus threshold is calculated as follows.

$$\frac{(48+50+37+62+38+70+55+37+64+60)-37-70}{8} = 51.75$$

Unit 06 Array FRQ – SOLUTION

(a) Complete the method `computeBonusThreshold` below, which is intended to return the bonus threshold based on the contents of the `itemsSold` array. Assume that `itemsSold` has been filled appropriately, and that the array contains at least three employees.

```
/** Returns the bonus threshold as described in part (a).
  */
public double computeBonusThreshold()

   int total = itemsSold[0];
   int min = itemsSold[0];
   int max = itemsSold[0];
   for(int i=1; i<itemsSold.length; i++) {
       total += itemSold[i];
       if(itemsSold[i] < min)
           min = itemSold[i];
       if(itemSold[i] > max)
           max = itemSold[i];
   }
   return (total – min – max)/
       double(itemSold.length-2);
}
```

// **Total: 5**

+1 Accesses all elements in itemsSold
+1 Determines minimum value in itemsSold
+1 Determines maximum value in itemsSold
+1 Computes the threshold from elements in itemsSold
+1 Returns the correct threshold as a double
-1 (v) Array/collection access confusion ([] get)
-1 (w) Extraneous code that causes side-effect
(e.g., printing to output, incorrect precondition check)
-1 (x) Local variables used but none declared
-1 (y) Destruction of persistent data (e.g., changing value referenced by parameter)

Unit 06 Array FRQ – SOLUTION

The `computeWages` method is intended to calculate the wages for each employee and to assign them to the appropriate element of the array `wages`. For example, `wages[3]` should be assigned the wages for employee

An employee's wages consist of their daily wages plus a possible bonus and are calculated as follows.

- Each employee's wages are equal to the fixed wage plus the number of items sold times the amount paid per item sold.
- If the employee sold more items than the bonus threshold, the employee also receives a *10 percent* bonus added to their wages.

As described in part (a), computeBonusThreshold() returns 51.75 for the example array below.

	0	1	2	3	4	5	6	7	8	9
`itemsSold`	48	50	37	62	38	70	55	37	64	60

Suppose that `fixedWage` is `10.0` and `perItemWage` is `1.5`.
Employee `0` did not sell more items than the bonus threshold, so employee `0`'s wages are equal to `10.0 + 1.5 * 48`, which evaluates to `82.0`. This value will be assigned to `wages[0]`.
Employee `9` sold more items than the bonus threshold, so employee `9` receives a *10 percent* bonus.
Employee `9`'s wages are equal to `(10.0 + 1.5 * 60) * 1.1`, or `110.0`. This value will be assigned to `wages[9]`.

(b) Write the method computeWages. Assume that itemsSold has been filled appropriately, and there are at least three employees in the array. Assume also that the wages array and the itemsSold array have the same length. Your solution must call `computeBonusThreshold` appropriately to receive full credit.

```
/** Computes employee wages as described in part (b)
  * and stores them in wages.
  * The parameter fixedWage represents the fixed amount each employee
  * is paid per day.
  * The parameter perItemWage represents the amount each employee
  * is paid per item sold.
  */
public void computeWages(double fixedWage, double perItemWage)

  double threshold = computeBonusThreshold();
  for (int i=0; i < itemsSold.length; i++)
  {
      double baseWage = fixedWage
         + perItemWage * itemsSold[i];
      if (itemsSold[i] > threshold) {
          wages[i] = baseWage * 1.1;
      } else {
          wages[i] = baseWage;
      }
  }
}
```

// **Total: 4**
+1 Correct call to computeBonusThreshold
+1 Differentiate computation of wage based on bonus threshold
+1 Assign to an element with a correctly computed wage
+1 Computes all wages and assigns them to wages
-1 (v) Array/collection access confusion ([] get)
-1 (w) Extraneous code that causes side-effect
(e.g., printing to output, incorrect precondition check)
-1 (x) Local variables used but none declared
-1 (y) Destruction of persistent data (e.g., changing value referenced by parameter)
-1 (z) Void method or constructor that returns a value