Chief Reader Report on Student Responses:

2017 AP® Computer Science A Free-Response Questions

- Number of Students Scored 60,519
- Number of Readers 308
- Score Distribution

Exam Score	N	%At
5	14,623	24.2
4	12,650	20.9
3	13,271	21.9
2	6,970	11.5
1	13,005	21.5

- Global Mean 3.15

The following comments on the 2017 free-response questions for AP® Computer Science A were written by the Chief Reader, John Cigas of Park University. They give an overview of each free-response question and of how students performed on the question, including typical student errors. General comments regarding the skills and content that students frequently have the most problems with are included. Some suggestions for improving student preparation in these areas are also provided. Teachers are encouraged to attend a College Board workshop to learn strategies for improving student performance in specific areas.

Question #1

Task: ArrayList

Max. Points: 9 **Mean Score:** 4.10

What were responses expected to demonstrate in their response to this question?

This question involves identifying and processing the digits of a nonnegative integer. The `Digits` class is used to store these digits in an `ArrayList` private instance variable. Students were asked to write the constructor of the `Digits` class and a method to determine if the stored digits of the number are in strictly increasing order.

In part (a) students were asked to write a constructor that initializes the instance variable and fills it with all of the digits from the parameter `num`. Students must understand that the `ArrayList` instance variable must be instantiated before elements can be added. Students also needed to identify and isolate each digit of an integer value and add objects to an `ArrayList` in the correct order without going out of bounds.

In part (b) students were asked to complete a `boolean` method that returns `true` if the stored digits in `digitList` are in strictly increasing order; otherwise, it returns `false`. Students needed to demonstrate understanding of writing and evaluating `boolean` expressions and returning correct `boolean` values. Additionally, they needed to compare consecutive values in a list without going out of bounds.

How well did the response address the course content related to this question? How well did the responses integrate the skills required on this question?

Although most students understood the concept of an `ArrayList` and could add items to an `ArrayList`, very few knew they had to instantiate an instance variable in the constructor, and of those that tried, many did the instantiation incorrectly.

Students used two common approaches to extract digits from an integer: the mathematical approach using remainder and the `String` approach using either the `charAt` or `substring` method. Students who used the mathematical approach understood that finding the remainder when dividing by 10 would extract the rightmost digit of the number, but sometimes had difficulty deciding which type of loop to use and/or which terminating condition to use to exit the loop. This approach stayed within the AP Computer Science A Java subset. Alternatively, some students chose to convert the `int` parameter into a `String` and extract digits from that string using various `String` methods, but many did not do this correctly. Students also had difficulty converting the `char` or `String` representation of their identified digit back to an `Integer` or `int` before adding to the `ArrayList` of `Integer` objects. The use of data types `Character` and `char`, as well as converting from `String` to `Integer`, are not in the course subset.

Most students wrote a correct comparison statement to determine if two consecutive `ArrayList` elements were in strictly increasing order. Students who were unsuccessful typically either failed to access an element of the `ArrayList` correctly using the `get` method or they used an incorrect `boolean` operator to compare consecutive elements.

Some students failed to recognize that for an `ArrayList` with n elements, there are n - 1 pairs and, therefore, the loop must execute $n - 1$ times, not n times.

Most students were able to logically determine whether or not the list elements were in increasing order and return the correct `boolean` value. Typically, students who did not return the correct value only compared and returned the result of one pair of consecutive elements or returned `true` when the first pair in increasing order was found.

What common student misconceptions or gaps in knowledge were seen in the responses to this question?

Common Misconceptions/Knowledge Gaps	*Responses that Demonstrate Understanding*
Most students failed to instantiate the private instance variable `digitList`.	`digitList = new ArrayList<Integer>();`
Some students failed to identify a single digit of the parameter `num` because they used division instead of remainder. `int adigit = num / 10;`	`int adigit = num % 10;`

Students tried to identify a length one substring/character of the `String` representation of `num` by treating `num` itself as a `String`.
`num.charAt(i)`
`num.substring(k, k + 1)`
`String s = num;`
`String s =(String)num;`
`String s = num.toString();`
	`String s = "" + num;`
	`s.charAt(i)`
	or
	`String s = ((Integer)num).toString();`
	`s.substring(k, k + 1))`

Students failed to convert a one-character substring or single character to an `int` or `Integer` when adding to a list of `Integer` objects.
`digitList.add(s.substring(k, k + 1));`
`digitList.add(s.charAt(i));`
`digitList.add((int)(s.charAt(i)));`
	`digitList.add(Integer.valueOf(s.substring(k, k+1)));`
	`digitList.add(Integer.parseInt(s.substring(k, k+1)));`
	`digitList.add(s.charAt(i) – '0');`
	Note, these solutions are outside the AP Java Subset.

Students failed to add digits to `digitList` in the correct order.
`while (num / 10 > 0)`
`{`
`  digitList.add(num % 10);`
`  num /= 10;`
`}`
`digitList.add (num % 10);`
	`while (num / 10 > 0)`
	`{`
	`  digitList.add(0, num % 10);`
	`  num /= 10;`
	`}`
	`digitList.add (0, num % 10);`

Students failed to add the digit zero when `num` is zero.
`while (num > 0)`
`{`
`  digitList.add(0, num % 10);`
`  num /= 10;`
`}`
	`if (num == 0)`
	`{`
	`  digitList.add(0);`
	`}`
	`while (num > 0)`
	`{`
	`  digitList.add(0, num % 10);`
	`  num /= 10;`
	`}`
	or
	`while (num > 9)`
	`{`
	`  digitList.add(0, num % 10);`
	`  num /= 10;`
	`}`
	`digitList.add(0, num);`

Students accessed `digitList` as an array. `digitList[i] >= digitList[i + 1]`	`digitList.get(i) >= digitList.get(i + 1)`

When trying to determine if a consecutive pair of `digitList` elements is not in strictly increasing order, students failed to consider the case where the two elements are equal.
`if (digitList.get(x) > digitList.get(x + 1))`
`{`
`  return false;`
`}`
	`if (digitList.get(x) >= digitList.get(x + 1))`
	`{`
	`  return false;`
	`}`

When trying to determine if all consecutive pairs of `digitList` are in strictly increasing order, students failed to check <u>all</u> pairs.

```
for (int x = 0; x < digitList.size() - 1; x++)
{
   if (digitList.get(x) >= digitList.get(x + 1))
   {
      return false;
   }
   else
   {
      return true;
   }
}
```

```
for (int x = 0; x < digitList.size() - 1; x++)
{
   if (digitList.get(x) >= digitList.get(x + 1))
   {
      return false;
   }
}
return true;
```

When trying to determine if all consecutive pairs of `digitList` are in strictly increasing order, students incorrectly stored the result of every comparison and replaced the previous result.

```
boolean increasing = true;
for (int x = 0; x < digitList.size() - 1; x++)
{
   if (digitList.get(x) >= digitList.get(x + 1))
   {
      increasing = false;
   }
   else
   {
      increasing = true;
   }
}
return increasing;
```

```
boolean increasing = true;
for (int x = 0; x < digitList.size() - 1; x++)
{
   if (digitList.get(x) >= digitList.get(x + 1))
   {
      increasing = false;
   }
}
return increasing;
```

Students made loop boundary errors when comparing consecutive pairs of elements.

```
for (int x = 0; x < digitList.size(); x++)
{
   if (digitList.get(x) >= digitList.get(x + 1))
   {
      return false;
   }
}
return true;
```

or

```
for (int x = 0; x < digitList.size(); x++)
{
   if (digitList.get(x - 1) >= digitList.get(x))
   {
      return false;
   }
}
return true;
```

```
for (int x = 0; x < digitList.size() - 1; x++)
{
   if (digitList.get(x) >= digitList.get(x + 1))
   {
      return false;
   }
}
return true;
```

or

```
for (int x = 1; x < digitList.size(); x++)
{
   if (digitList.get(x - 1) >= digitList.get(x))
   {
      return false;
   }
}
return true;
```

Based on your experience at the AP® Reading with student responses, what advice would you offer to teachers to help them improve the student performance on the exam?

Students need practice with programs that reinforce the following concepts:

- When an instance variable is a list or an array, it needs to be instantiated in a constructor
- Variables of a primitive type (e.g., `int`, `double`, `char`, `boolean`) do not have methods
- When a list has n elements it has $n - 1$ consecutive pairs of elements, which requires $n - 1$ iterations through a loop
- Some algorithms, like sequential search or checking if a list is in increasing order, have two stopping conditions. One condition (item found, items not in increasing order) can be detected within the loop, but the other can only be detected after all items have been examined and the loop has completed.

What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?

Suggested resources include:

- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the List and ArrayList section would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html
- The 2007 free-response question on self-divisors also requires students to extract individual digits from a number. The Runestone Interactive Java Review offers an interactive environment for students to practice solving this question and a video, which will walk students through the response. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/LoopBasics/selfDivisorA.html

Question #2

Task: Class Design

Max. Points: 9 **Mean Score:** 5.86

What were responses expected to demonstrate in their response to this question?

This question focused on the mechanics of creating a class to implement a specific interface. Students were asked to design the class `MultPractice` so that it implements two methods in the given `StudyPractice` interface. Students had to decide on an internal representation of the object that would allow them to implement two required methods so they are consistent with the given examples. In addition to demonstrating an understanding of class, constructor, and method header syntax, students had to correctly declare, initialize, access, and update instance variables. Students were expected to properly encapsulate their data members by declaring them as `private` and to properly expose the interface methods by declaring them to be `public`. To obtain a correct implementation, students had to demonstrate an understanding of the difference between returning a value and producing a side effect and how to convert from an integer to a string.

How well did the response address the course content related to this question? How well did the responses integrate the skills required on this question?

The problem includes a design component where students are free to choose any internal representation that allows them to implement the methods. This separation of interface and implementation is a central theme in computer science. Additionally, students had to demonstrate an understanding of how to control the visibility of class elements so that the language would enforce the separation. The internal representation had to be hidden and inaccessible from the client, and the interface methods had to be publicly available.

This question addresses the fundamental skill of how to create a class so that it obeys a specific interface. The actual implementation and program logic required to make the class operational is minimal and of secondary importance.

Overall, students who understood the concept of making a class match the specifications scored highly on this question.

What common student misconceptions or gaps in knowledge were seen in the responses to this question?

Common Misconceptions/Knowledge Gaps	*Responses that Demonstrate Understanding*

Students omitted the keyword `class` or the entire `implements` clause.
`public MultPractice implements StudyPractice`
`public class MultPractice`
Students used `extends` instead of `implements`.
`class MultPractice extends StudyPractice`
Students also included parameters in the class header. This frequently coincided with a missing or incorrect constructor header.
`class MultPractice(int x, int y)`
`implements StudyPractice`
	`public class MultPractice implements StudyPractice`
	The `public` access specifier was not required.
	`class MultPractice implements StudyPractice`

Failure to declare *any* instance variables meant that students could not earn points for the declarations, the initialization in the constructor, building the problem string in `getProblem`, and updating the problem in `nextProblem`.
Students declared instance variables incorrectly by placing them above the class header, or by omitting the `private` access specifier, or by using a different specifier. Students sometimes forgot to include the type in the declaration or incorrectly added `static`.
`public int first, second;`
`protected int first, second;`
`private first, second;`
`int first, second;`
`private static int first, second;`
	`private int first, second;`
	A `final` modifier on the first operand is acceptable.
	`private final int first;`
	`private int second;`
	Students sometimes made use of an additional instance variable to hold the problem string.
	`private String problem;`
	Students sometimes made use of an additional instance variable to hold a count of the number of times `nextProblem` is called.
	`private int count = 0;`

For the constructor, students inserted `void` as a return type or omitted the parameter types. Sometimes, the assignments were backwards. If students used the same names for parameters and instance variables, they failed to distinguish them with `this`.

```
void MultPractice(int num1, int num2)
{
   num1 = first;
   num2 = second;
}
```

```
public MultPractice(first, second)
{
   first = first;
   second = second;
}
```

Students frequently tried to create the instance variables inside the constructor.

```
MultPractice(int num1, int num2)
{
   private int first = num1;
   private int second = num2;
}
```

```
public MultPractice(int num1, int num2)
{
   first = num1;
   second = num2;
}
```

The `public` modifier on the constructor header is not strictly required. Also, a call to the default constructor as the first statement is harmless.

```
MultPractice(int first, int second)
{
   super();
   this.first = first;
   this.second = second;
}
```

If the `problem` instance variable is used:

```
MultPractice(int num1, int num2)
{
   first = num1;
   second = num2;
   problem = num1 + " TIMES " + num2;
}
```

Students omitted the `public` modifier on the `getProblem` header. Occasionally, students made the method `private` and/or `static`.

```
String getProblem()
private String getProblem()
public static String getProblem()
```

Students padded the method header with additional parameters.

```
public String getProblem(int first, int second)
{
   return first + " TIMES " + second;
}
```

When building the string, students erroneously referenced the names of constructor parameters, or quoted the variable names, or made an incorrect conversion of an `int` value to a string.

```
"first TIMES second"
(String) first + " TIMES "
first.intValue() + " TIMES "
first.toString() + " TIMES "
```

Students inserted a call to `nextProblem`, incorrectly updating the second value.

```
nextProblem();
```

```
public String getProblem()
{
   return first + " TIMES " + second;
}
```

If the `count` instance variable is used:

```
public String getProblem()
{
   return first + " TIMES " + (second + count);
}
```

If the instance variable `problem` is used:

```
public String getProblem()
{
   return problem;
}
```

Occasionally, students explicitly converted the instance variable to a string before concatenating.

```
String.valueOf(first) + " TIMES "
Integer.toString(first) + " TIMES "
```

Students omitted the `public` modifier on the `nextProblem` header or they added a parameter, possibly shadowing an instance variable.

```
void nextProblem()
```

```
public void nextProblem(int second)
{
   second++;
}
```

There were a variety of incorrect attempts at incrementing.

```
second + 1;
second = second++;
num2++;
int second++;
```

Students called the constructor to accomplish the update.

```
MultPractice(first, second + 1);
```

Students returned a value from this `void` method or printed the value instead.

```
return second + 1;
System.out.print(second + 1);
```

```
public void nextProblem()
{
   second++;
}
```

If the `count` instance variable is used:

```
public void nextProblem()
{
   count++;
}
```

If the `problem` instance variable is used:

```
public void nextProblem()
{
   second++;
   problem = first + " TIMES " + second;
}
```

Based on your experience at the AP® Reading with student responses, what advice would you offer to teachers to help them improve the student performance on the exam?

- Teachers should have students practice writing classes without the aid of a computer. Students become dependent on IDEs and compilers and do not learn the proper structure and syntax for class and method headers.
- Teachers should stress the importance of encapsulation and visibility. All instance variables should be declared as private. Students should pay close attention to the visibility and signature of methods, especially when inherited or implemented.
- Students need to know how to write constructors. Constructors do not return values and have no return type.
- Printing is not returning. Rarely are students asked to print. Unsolicited print statements usually receive a deduction in the scoring rubric.
- Declaring static instance variables does not preserve state in individual objects.
- Students should learn how to properly construct a string. Casting an integer to a string is not allowed.

What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?

Suggested resources include:

- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the Java Basics: Classes and Objects section and the Object-Oriented Concepts section would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html

Question #3

Task: String Processing

Max. Points: 9 **Mean Score:** 3.84

What were responses expected to demonstrate in their response to this question?

This question tested the students' ability to use `String` methods from the AP Java subset to perform processing of strings using various parameters and instance variables. The problem required students to use a provided helper method in their solutions.

In part (a) students were asked to write a method to examine and potentially modify the instance variable `currentPhrase`. Students were required to use the already-implemented `findNthOccurrence` helper method to replace the `nth` occurrence of a string in `currentPhrase` if it was present the number of times specified by the parameter. The new string was created by identifying and extracting the substrings of `currentPhrase` to retain, concatenating these strings with the replacement string parameter `repl` in the correct order, and assigning this value to `currentPhrase`.

In part (b) students were asked to write a method to find the index of the last occurrence of a specified string in `currentPhrase` using the already-implemented `findNthOccurrence` helper method. In finding the last occurrence, student solutions need to be capable of examining `currentPhrase` multiple times using either a call to `findNthOccurrence` or by reimplementing this functionality and examining various substrings while advancing through `currentPhrase`.

Students who used `findNthOccurrence` to examine `currentPhrase` were required to demonstrate an understanding of iteration: setting the loop lower bound ensuring the precondition $n > 0$ is not violated in the `findNthOccurrence` call and setting the loop upper bound to allow `findNthOccurrence` to be called with an argument `n` equal to `currentPhrase.length`.

Students who reimplemented the `findNthOccurrence` functionality were also required to demonstrate an understanding of iteration. The loop bounds needed to be set to ensure that no bounds errors occurred in the matching of substrings to the string parameter.

Regardless of the algorithm used to find the *n*th occurrence, students were required to return the correct index of the last occurrence, or return -1 if no occurrence was found.

How well did the response address the course content related to this question? How well did the responses integrate the skills required on this question?

Student responses showed an understanding of the following concepts:

- Using abstraction by calling the provided method `findNthOccurrence`
- Using a conditional to preserve `currentPhrase` if the `nth` occurrence did not exist
- Searching through `currentPhrase`, either by calling `findNthOccurrence` or by creating their own search algorithm
- Maintaining state in an object through the instance variable `currentPhrase`

Student responses exhibited the following skills:

- Calling methods with parameters
- Finding the location of `str` in `currentPhrase`
- Extracting substrings from `currentPhrase`
- Creating a replacement string using the identified components of `currentPhrase` and `repl`
- Returning the index of the last occurrence of a substring

What common student misconceptions or gaps in knowledge were seen in the responses to this question?

Common Misconceptions/Knowledge Gaps	*Responses that Demonstrate Understanding*
Students failed to call the `Phrase` instance method `findNthOccurrence` from within the `Phrase` class correctly. `int index = currentPhrase.findNthOccurrence(str, n);`	`int index = findNthOccurrence(str, n);`

Students miscalculated the appropriate indices for calls to the `substring` method, both numerically and by using `repl` instead of `str`.
`String temp1, temp2;`
`temp1 = currentPhrase.substring(0, index - 1);`
`temp2 = currentPhrase.substring(index + repl.length());`
	Calling `str.substring(start, end)` returns the substring beginning at `start` and ending at `end-1`. Index `end` is not included in the string that is returned.
	`String temp1, temp2;`
	`temp1 = currentPhrase.substring(0, index);`
	`temp2 = currentPhrase.substring(index + str.length());`

Students failed to reassign the value of a string, often by using some perceived mutator method, such as `replace`. `currentPhrase.replace(temp1 + repl + temp2);`	`currentPhrase = temp1 + repl + temp2;`

<table>
<tr>
<td>Students failed to structure loop bounds correctly when processing a string in nontraditional ways:
• When the loop bounds are not [0, length() - 1]
• When comparing substrings of length > 1</td>
<td>

```
int n = 1;
int index = -1;
while (findNthOccurrence(str, n) != -1)
{
   index = findNthOccurrence(str, n);
   n++;
}
return index;
```

or

```
int index = -1;
for (int n = 0; n < currentPhrase.length() -
     str.length(); n++)
{
   if (currentPhrase.substring(n,
       n + str.length()).equals(str))
   {
       index = n;
   }
}
return n;
```

</td>
</tr>
<tr>
<td>Students made method calls with parameters that violated the method's preconditions.

```
int n = 0;
while (findNthOccurrence(str, n) != -1)
```

</td>
<td>

```
int n = 1;
while (findNthOccurrence(str, n) != -1)
```

</td>
</tr>
</table>

Based on your experience at the AP® Reading with student responses, what advice would you offer to teachers to help them improve the student performance on the exam?

Students should practice invoking a variety of methods.

- Develop solutions requiring students to use methods that they do not implement, both instance methods and static methods
- Emphasize that methods invoked within the same class are not called on an instance of the class
- Highlight the use of pre and post conditions

Students should practice using methods to manipulate strings.

- Develop a lesson emphasizing the proper ways to modify `String` objects
- Emphasize the correct indices to use when extracting substrings

Students should practice determining the proper lower bound and upper bound for loops.

- Assign problems with loops requiring different lower (do not start at 0) and/or different upper bounds (different from `< .length()`)

What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?

Suggested resources include:

- Codingbat.com provides students with practice writing the body of a method based on a given specification. This website provides three different levels of questions that involve manipulating `String` objects.
- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the Strings section would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html

Question #4

Task: 2D Array Processing

Max. Points: 9 **Mean Score:** 4.75

What were responses expected to demonstrate in their response to this question?

This question involved reasoning about two-dimensional (2D) arrays of integers and objects. The students were expected to write two static methods of an enclosing `Successors` class. A provided `Position` class was used to represent an integer's location (row and column) in a 2D array.

In part (a) students were asked to implement a static method with two parameters, an integer value, and a 2D array of integers. They were expected to search the given array for the given value. If found, students were expected to create and return a new `Position` object representing the value's location in the array. Otherwise they were expected to return `null`.

In part (b) students were asked to implement a static method with a 2D array of integers parameter. They were expected to create a 2D array of `Position` references with the same dimensions as the given array. Then they were to use the method they implemented in part (a) to find the position of the successor (integer one greater) for each integer in the given array. They then assigned this successor position to the corresponding element in the new array. Finally, they returned the new array.

In writing the required methods, correct responses demonstrated the ability to search a 2D array, create new `Position` and 2D array objects, return objects and `null`, use parameters and local variables, implement and invoke `static` methods, and demonstrate the principle of code reuse by utilizing a previously implemented method.

How well did the response address the course content related to this question? How well did the responses integrate the skills required on this question?

Searching an array or list is a common algorithm. Searching a 2D array is slightly more involved but most students did this correctly. They were less consistent about creating a new `Position` object and returning it or `null` as necessary.

The idea of a successor array is a somewhat abstract concept. However, student responses demonstrated that a significant share of students understood this concept. Most of these students utilized the part (a) method and wrote solid solutions. Solutions that attempted to reimplement the part (a) method generally had significant problems.

What common student misconceptions or gaps in knowledge were seen in the responses to this question?

Common Misconceptions/Knowledge Gaps	*Responses that Demonstrate Understanding*

Students omitted the keyword `new` when creating an object of a class or creating an array.
`Position pos = Position(row, col);`
`Position[][] newArr =`
`  Position[intArr.length][intArr[0].length];`
	`Position pos = new Position(row, col);`
	`Position[][] newArr =`
	`  new Position[intArr.length][intArr[0].length];`

Students incorrectly created a new 2D array in several ways.
By not specifying the correct number of columns:
`Position[][] newArr =`
`  new Position[intArr.length][intArr[].length];`
By using an incorrect type:
`int[][] newArr = …`
`String[][] newArr = …`
By not creating a new array at all:
`Position[][] newArr;`
	`Position[][] newArr =`
	`  new Position[intArr.length][intArr[0].length];`

Students failed to implement a 2D sequential search correctly by using an if-else statement inside the loop.

```
for (int r = 0; r < intArr.length; r++)
{
   for (int c = 0; c < intArr[0].length; c++)
   {
      if (intArr[r][c] == num)
      {
         return new Position(r, c);
      }
      else
      {
         return null;
      }
   }
}
```

or

```
Position p = null;
for (int r = 0; r < intArr.length; r++)
{
   for (int c = 0; c < intArr[0].length; c++)
   {
      if (intArr[r][c] == num)
      {
         p = new Position(r, c);
      }
      else
      {
         p = null;
      }

   }
}
return p;
```

```
for (int r = 0; r < intArr.length; r++)
{
   for (int c = 0; c < intArr[0].length; c++)
   {
      if (intArr[r][c] == num)
      {
         return new Position(r, c);
      }
   }
}
return null;
```

or

```
Position p = null;
for (int r = 0; r < intArr.length; r++)
{
   for (int c = 0; c < intArr[0].length; c++)
   {
      if (intArr[r][c] == num)
      {
         p = new Position(r, c);
      }
   }
}
return p;
```

Students attempted to access variables that were out of scope.
`boolean found = false;`
`for (int r = 0; r < intArr.length; r++)`
`{`
`  for (int c = 0; c < intArr[0].length; c++)`
`  {`
`  if (intArr[r][c] == num)`
`  {`
`  **Position p = new Position(r, c);**`
`  found = true;`
`  }`
`  }`
`}`
`if (found)`
`{`
`  **return p;**`
`}`
`else`
`{`
`  return null;`
`}`
	Position p = null;
	`for (int r = 0; r < intArr.length; r++)`
	`{`
	`  for (int c = 0; c < intArr[0].length; c++)`
	`  {`
	`  if (intArr[r][c] == num)`
	`  {`
	`  **p = new Position(r, c);**`
	`  }`
	`  }`
	`}`
	return p;

Students used enhanced for loop variables as row and column indices.
`for (int r : intArr)`
`{`
`  for (int c : intArr)`
`  {`
`  if (intArr[r][c] == num)`
`  {`
`  return new Position(r, c);`
`  }`
`  }`
`}`
`return null;`
	`int r = 0;`
	`int c = 0;`
	`for (int[] row : intArr)`
	`{`
	`  for (int val : row)`
	`  {`
	`  if (val == num)`
	`  {`
	`  return new Position(r, c);`
	`  }`
	`  c++;`
	`  }`
	`  r++;`
	`  c = 0;`
	`}`
	`return null;`

Students used `this.` to access static methods inside of a static method.
`if (this.findPosition(…))`
	`if (Successors.findPosition(…))`
	or
	`if (findPosition(…))`

Students misunderstood the encapsulation by attempting to use non-existent or inaccessible instance variables or methods of a `Position` object.
`Position pos = new Position(3,4)… ;`
`int r = pos.row;`
`int r = pos.getRow();`
	The `Position` class as specified has no known public instance variables or accessor methods.
	`int r = 3;`
	`int c = 4;`
	`Position pos = new Position(r, c);`

Based on your experience at the AP® Reading with student responses, what advice would you offer to teachers to help them improve the student performance on the exam?

Students should practice writing free-response problems that include the following skills:

- Constructing objects of unfamiliar problem-specific classes
- Constructing 2D arrays
- Performing searches of 2D arrays
- Reusing methods instead of reimplementing them
- Implementing and invoking static methods

What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?

Suggested resources include:

- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the Two-Dimensional Arrays section would be most helpful for this question. This section includes practice free-response questions that use 2D arrays. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html