2022

AP CollegeBoard

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free-Response Question 3

- ☑ **Scoring Guidelines**
- ☑ **Student Samples**
- ☑ **Scoring Commentary**

Applying the Scoring Criteria

Apply the question scoring criteria first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

v) Array/collection access confusion (`[] get`)
w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
x) Local variables used but none declared
y) Destruction of persistent data (e.g., changing value referenced by parameter)
z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ ≤ ≥ <> ≠)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `( )`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `( )` on parameter-less method or constructor invocations
- Missing `( )` around `if` or `while` conditions

Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be **unambiguously inferred from context, for example, "ArayList" instead of "ArrayList". As a counterexample, note that if the code declares `"int G=99, g=0;"`, then uses `"while (G < 10)"` instead of `"while (g < 10)"`, the context does **not** allow for the reader to assume the use of the lower case variable.*

Question 3: Array / ArrayList — 9 points

Canonical solution

(a) — **3 points**

```
public double getAverageRating()
{
   int sum = 0;

   for (Review r : allReviews)
   {
      sum += r.getRating();
   }

   return (double) sum / allReviews.length;
}
```

(b) — **6 points**

```
public ArrayList<String> collectComments()
{
   ArrayList<String> commentList = new ArrayList<String>();

   for (int i = 0; i < allReviews.length; i++)
   {
      String comment = allReviews[i].getComment();
      if (comment.indexOf("!") >= 0)
      {
         String last =
            comment.substring(comment.length() - 1);
         if (!last.equals("!") && !last.equals("."))
         {
            comment += ".";
         }
         commentList.add(i + "-" + comment);
      }
   }
   return commentList;
}
```

(a) `getAverageRating`

	Scoring Criteria	Decision Rules	
1	Initializes and accumulates sum	Response **can** still earn the point even if they • fail to use a loop to accumulate • fail to call `getRating` or call `getRating` incorrectly	**1 point**
2	Accesses every element of `allReviews` (*no bounds errors*)	Responses **will not** earn the point if they • access the elements of `allReviews` incorrectly	**1 point**
3	Computes and returns `double` average rating based on `getRating` return values (*algorithm*)	Response **can** still earn the point even if they • fail to initialize the accumulator for the sum Responses **will not** earn the point if they • fail to accumulate the sum of all ratings • use integer division to compute average • include parameters on call to `getRating` • fail to call `getRating` on all elements of `allReviews`	**1 point**
		Total for part (a)	**3 points**

(b) `collectComments`

	Scoring Criteria	Decision Rules	
4	Instantiates an `ArrayList` capable of holding `String` objects		**1 point**
5	Accesses every element of `allReviews` (*no bounds errors*)	Responses **can** still earn the point even if they • fail to keep track of the index Responses **will not** earn the point if they • access the elements of `allReviews` incorrectly	**1 point**
6	Calls `getComment` on an element of `allReviews`, calls at least one `String` method appropriately on the `getComment` return value, and all `String` method calls are syntactically valid	Responses **can** still earn the point even if they • call some of the `String` methods on objects other than `getComment` return values Responses **will not** earn the point if they • include a parameter when calling `getComment` • call any `String` methods incorrectly • call any `String` methods on objects other than `String` values	**1 point**
7	Compares the final character of the comment to both a period and an exclamation point	Responses **can** still earn the point even if they • use incorrect logic in the comparison • call `String` methods incorrectly Responses **will not** earn the point if they • use `==` instead of `equals` when comparing `String` objects	**1 point**
8	Assembles string appropriately based on result of comparison of last character with period and exclamation point (*algorithm*)	Responses **can** still earn the point even if they • call `String` methods incorrectly • use `==` instead of `equals` Responses **will not** earn the point if they • fail to keep track of the element index • use incorrect logic in the comparison	**1 point**
9	Adds all and only appropriate constructed strings to the `ArrayList` (*algorithm*)	Responses **can** still earn the point even if they • initialize the `ArrayList` incorrectly • fail to return the constructed `ArrayList` (*return is not assessed*) • assemble the review string incorrectly • access the elements of `allReviews` incorrectly	**1 point**

Total for part (b)	**6 points**

Question-specific penalties
None

Total for question 3	**9 points**

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ○ Question 2 ○ Question 3 ● Question 4 ○

Begin your response to each question at the top of a new page.

a)

```
Public double GetAverageRating()
{
    int sum = 0;
    for (int i = 0; i < allReviews.length; i++)
    {
        sum += allReviews[i].GetRating();
    }
    return (double)(sum) / (allReviews.length);
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ○ Question 2 ○ Question 3 ● Question 4 ○

Begin your response to each question at the top of a new page.

b)

```
public ArrayList<String> collectComments()
{
    ArrayList<String> comments = new ArrayList<String>();
    for (int i = 0; i < allReviews.length; i++)
    {
        if( allReviews[i].indexOf("!") >= 0 )
        {
            String review = allReviews[i].getComment();
            if(!(review.substring(review.length -1).equals(".")))
            &&!(review.substring(review.length -1).equals("!")))
            {
                review += ".";
            }
            comments.add(i + "-" + review);
        }
    }
    return comments;
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ○ Question 2 ○ Question 3 ● Question 4 ○

Begin your response to each question at the top of a new page.

```
public double getAverageRating() {
int total = 0
for(int i = 0; i < allReviews.length(); i++)
 total += allReviews[i];
double avg = (double) total / allReviews.length();
return avg;
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ○ Question 2 ○ Question 3 ● Question 4 ○

Begin your response to each question at the top of a new page.

```
public ArrayList<String> collectComments(){
ArrayList<> exM = new ArrayList<String>;
int K = 0;
for (int i = 0; i < allReviews.length(); i++){
   String com = allReviews[i].getComment;
   if(com.indexOf("!") >-1)
   exM.add(i + " " + com);
   if(com.lastIndexOf("!") != com.length()-1 || com.lastIndex
     of(".") != com.length()-1)
   exM[K] = i+" - "+com+".";

   k++;
  }
  return exM;
```


Use a pencil only. Do NOT write your name. Do NOT write outside the box.

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ○ Question 2 ○ Question 3 ● Question 4 ○

Begin your response to each question at the top of a new page.

a.

```
public getAverageRating()
{
    double total=0
    double average=0
    for(int i=0; i<allReviews[].length; i++)
    {
        total = total + allReviews[i].getRating();
    }
    average = total / allReviews[].length;
    return average;
}
```

Part a

```
Public ArrayList<String> collectComments()
{
    String[] exited=new String[];
    for(int i=0; i<allReviews[].length; i++)
    {
        for(int i=0; i<allReviews[i].length(); i++)
        {
            if(allReviews[i].substring(i, i+1)=="!")
            {
moreline→   if(allreviews[i].substring(allReviews[i].length-1)!="."||
                allreviews[].substring(allReviews[i].length-1)!="!")
                {
                    exited.add(i+"-"+allreviews[i]+".";
                }
                else
                {
                    exited.add(i+"-"+allReviews[i];
                }
            }
        }
    return exited;
}
```

Part b

Question 3

Note: Student samples are quoted verbatim and may contain spelling and grammatical errors.

Overview

This question tested the student's ability to:

- Write program code to create objects of a class and call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- Write program code to create, traverse, and manipulate elements in 1D array and `ArrayList` objects.

This question involved the traversal and manipulation of a one-dimensional (1D) array containing `Review` objects and the instantiation and building of an `ArrayList` containing `String` objects. Students were expected to write two methods in the `ReviewAnalysis` class, using its 1D array instance variable, and use two methods from the `Review` class when accessing `Review` objects. Students were also expected to use methods of the `String` class and construct a `String` object.

In part (a) students were expected to write a loop that accessed each element of the array instance variable `allReviews` and returned the calculated average of all the return values of the method `getRating`. Students had to declare and initialize a variable to hold the sum of all ratings. Inside the loop, students were expected to call the method `getRating` on all `Review` elements in `allReviews`, accumulating a total of the ratings. Outside the loop, students were expected to calculate and return the average rating as a `double` value.

In part (b) students were asked to develop an algorithm to: (1) Identify all `Review` elements of `allReviews` that have comments containing an exclamation point, using the `getComment` method; (2) build an `ArrayList` of `String` objects, each of which would be a string based on an identified review and formatted according to the specification:

> "<index number of `Review` object><hyphen><comment>"

When building the formatted `String` to be added to the `ArrayList`, the specification required that each identified formatted comment end in a period if the original comment did not already end with a period or an exclamation point. In identifying the comments meeting the specification, students were to use methods of the `String` class, such as `indexOf`, `substring`, and `equals`, as well as Boolean operators.

Question 3 (continued)

Sample: 3A
Score: 8

In part (a) point 1 was earned because the integer `sum` is initialized and then accumulated in the body of the `for` loop. Point 2 was earned because all elements of the array `allReviews` are accessed correctly, and there are no bounds errors. Point 3 was earned because the average is correctly calculated and returned. The average is calculated by accumulating the sum of ratings from all elements of `allReviews`. Each rating is accessed by a call to `getRating` on an element of `allReviews`. The average is then calculated by dividing the sum, cast to a `double`, by the length of the array, which represents the number of ratings.

In part (b) point 4 was earned because an `ArrayList` is properly instantiated to hold `String` objects. Point 5 was earned because all elements of `allReviews` are accessed appropriately with an indexed `for` loop, and there are no bounds errors. Point 6 was not earned because the call to the `indexOf` method in the first `if` statement is not correct. The expression `allReviews[i].indexOf("!")` is incorrect because the value referred to by `allReviews[i]` is not a `String`. The correct expression would be `allReviews[i].getComment().indexOf("!")`. Alternatively, the `String` returned by the method call `allReviews[i].getComment()` could be stored in a variable and used to call `indexOf`. If the `indexOf` call had been correct, point 6 would have been earned, even though parentheses are missing on the call `review.length - 1`. The missing parentheses is one of the minor errors for which no penalty is assessed on this exam. (See the "No Penalty" category on page 1 of the Scoring Guidelines for a complete list of such errors.) Point 7 was earned because the final character of the comment is compared to both a period and an exclamation point using the `equals` method of the `String` class. A correct response could also correctly extract the final character using the `charAt` method of the `String` class, in which case comparing the final character to both a period and an exclamation point would require `char` data types and the use of the `==` or `!=` operators. Point 8 was earned because the logic of examining the final character is correct, and an appropriate `String` is assembled using the correct index from the `for` loop. The logic is correct in the `if` statement because the response checks that the final character of the comment is not a period and is also not an exclamation point. A period is concatenated only to comments that end with neither a period nor an exclamation point. Point 9 was earned by adding to the `ArrayList` only the appropriate constructed strings, containing the required exclamation point, even though the call to `indexOf` is not syntactically valid. The incorrect `indexOf` method call was assessed in Point 6.

Sample: 3B
Score: 6

In part (a) point 1 was earned because the integer `total` is initialized and accumulated. The omitted call to `getRating` is not assessed in this point. Point 2 was earned because all elements of the array `allReviews` are accessed correctly, and there are no bounds errors. Point 3 was not earned because the average is not calculated using accumulated `getRating` return values. The omitted `getRating` method call is assessed in this point.

Question 3 (continued)

In part (b) point 4 was not earned because an `ArrayList` is not properly instantiated to hold `String` objects. The angle brackets (`<` and `>`) on the left-hand side of the assignment statement cannot be empty. The type of object to be stored in the `ArrayList` must be specified within the angle brackets. When writing a method that returns an `ArrayList<String>`, any of the following `ArrayList` declarations and instantiations would receive credit.

```
ArrayList list1 = new ArrayList();
ArrayList list2 = new ArrayList<>();
ArrayList list3 = new ArrayList<String>();
ArrayList<String> list4 = new ArrayList();
ArrayList<String> list5 = new ArrayList<>();
ArrayList<String> list6 = new ArrayList<String>();
ArrayList ArrayList = new ArrayList();
```

Point 5 was earned because all elements of `allReviews` are accessed correctly using an indexed `for` loop, and there are no bounds errors. Point 6 was earned because the `getComment` method retrieves a `String` from each review and stores its value in a `String` variable named `com`. All `String` method calls in the response are syntactically correct. Point 7 was earned because the final character of the `String` is correctly compared to both a period and an exclamation point. The `lastIndexOf` method is a `String` method that returns the index of the last occurrence of its parameter. Methods that are not part of the AP Java subset can be used in a response, as long as they are used correctly. Point 8 was not earned for multiple reasons. First, the logic used to examine the final character is incorrect; the `&&` operator should be used instead of the `||` operator. As a result of the incorrect use of `||`, a period is appended to comments that end in either a period or an exclamation point. Second, the update of an element in the `ArrayList` is incorrect. The statement `exM[k] = i + "-" + com + "."` does not modify the value of the `String` in the `ArrayList` and therefore no correctly constructed `String` exists. Either of these errors is sufficient to deem the point unearned. Point 9 was earned because all and only appropriate constructed strings are added to the `ArrayList`. Although the constructed values are not entirely correct, that error was assessed in point 8. Point 9 can still be earned even if a response assembles the review `String` incorrectly.

Sample: 3C
Score: 3

In part (a) point 1 was earned because the integer `total` is initialized and is accumulated in the body of the `for` loop. Point 2 was earned because all elements of the array `allReviews` are accessed correctly, and there are no bounds errors. The extraneous `[]` used when referencing the entire `allReviews` array is a minor "No Penalty" error. Point 3 was earned because the average is correctly calculated and returned. Because the accumulated sum is stored in the `double` variable `total`, the division in the calculation of the average produces the correct `double` value.

In part (b) point 4 was not earned because an `ArrayList` is not properly instantiated to hold `String` objects. The response instead attempts to declare a `String` array. Point 5 was not earned because the inner loop reuses the variable `i`. The updated value of `i` at the end of

Question 3 (continued)

the inner loop is used in the outer loop. This may result in some elements of `allReviews` being skipped. All elements of `allReviews` must be accessed in order to earn this point. Point 6 was not earned because the `getComment` method is never called to get a `String` object. All the `String` methods are called on `Review` objects rather than `String` objects. Point 7 was not earned because the comparisons of the final character with a period and an exclamation point use `!=` instead of `equals`. Even though the `String` methods are called incorrectly, the point could have been earned if `equals` had been used correctly. Point 8 was not earned because the logic used to examine the final character is incorrect; the `&&` operator should be used instead of the `||` operator to check that neither a period nor an exclamation point appears at the end of the `String`. The invalid use of `!=` is not the reason the point was not earned, as that was assessed in point 7. Point 9 was not earned for multiple reasons. First, the use of nested loops could allow duplicate copies of the same comment if multiple exclamation points occur in the comment. Second, the search for the exclamation point is syntactically incorrect.