2018

AP CollegeBoard

AP Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free Response Question 3

- ☑ **Scoring Guideline**
- ☑ **Student Samples**
- ☑ **Scoring Commentary**

AP® COMPUTER SCIENCE A
2018 SCORING GUIDELINES

Apply the question assessment rubric 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.*

AP® COMPUTER SCIENCE A
2018 SCORING GUIDELINES

Question 3: Code Word Checker

Class: `CodeWordChecker`	**9 points**

Intent: *Define implementation of a class to determine if a string meets a set of criteria*

+1 Declares header: `public class CodeWordChecker implements StringChecker`

+1 Declares all appropriate `private` instance variables

+3 Constructors

- **+1** Declares headers: `public CodeWordChecker(int __, int __, String __)` and `public CodeWordChecker(String __)`
- **+1** Uses all parameters to initialize instance variables in 3-parameter constructor
- **+1** Uses parameter and default values to initialize instance variables in 1-parameter constructor

+4 `isValid` method

- **+1** Declares header: `public boolean isValid(String __)`
- **+1** Checks for length between min and max inclusive
- **+1** Checks for unwanted string
- **+1** Returns `true` if length is between min and max and does not contain the unwanted string, `false` otherwise

AP® COMPUTER SCIENCE A
2018 SCORING GUIDELINES

Question 3: Scoring Notes

Class `CodeWordChecker`			**9 points**
Points	Rubric Criteria	Responses earn the point if they…	Responses will not earn the point if they…
+1	Declares header: `public class CodeWordChecker implements StringChecker`	• omit keyword `public`	• declare class `private` • declare class `static`
+1	Declares all appropriate `private` instance variables		• declare variables as `static` • omit keyword `private` • declare variables outside the class
+3	Constructors		
+1	Declares headers: `public CodeWordChecker (int __, int __, String __)` and `public CodeWordChecker (String __)`	• omit keyword `public`	• declare method `static` • declare method `private`
+1	Uses all parameters to initialize instance variables in 3-parameter constructor		• fail to declare instance variables • initialize local variables instead of instance variables • assign variables to parameters
+1	Uses parameter and default values to initialize instance variables in 1-parameter constructor	• initialize instance variables to default values when declared	• fail to declare instance variables • initialize local variables instead of instance variables • assign variables to parameters
+4	`isValid` method		
+1	Declares header: `public boolean isValid (String __)`		• fail to declare method `public` • declare method `static`
+1	Checks for length between min and max inclusive		• fail to use instance variables • fail to declare the method header
+1	Checks for unwanted string		• fail to use instance variables • fail to declare the method header
+1	Returns `true` if length is between min and max and does not contain the unwanted string, `false` otherwise	• have incorrect checks for length and/or containment, but return the correct value based on those checks	• fail to declare the method header • fail to return in all cases • only check one substring location for containment

AP® COMPUTER SCIENCE A
2018 SCORING GUIDELINES

Question 3: Code Word Checker

```
public class CodeWordChecker implements StringChecker
{
   private int minLength;
   private int maxLength;
   private String notAllowed;

   public CodeWordChecker(int minLen, int maxLen, String symbol)
   {
      minLength = minLen;
      maxLength = maxLen;
      notAllowed = symbol;
   }

   public CodeWordChecker(String symbol)
   {
      minLength = 6;
      maxLength = 20;
      notAllowed = symbol;
   }

   public boolean isValid(String str)
   {
      return str.length() >= minLength && str.length() <= maxLength &&
             str.indexOf(notAllowed) == -1;
   }

}
```

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

3A

Write the complete `CodeWordChecker` class. Your implementation must meet all specifications and conform to all examples.

```
public class CodeWordChecker implements StringChecker {

    private int val1;
    private int val2;
    private String word;

    public CodeWordChecker(int a, int b, String c) {
        val1 = a
        val2 = b
        word = c
    }
    public CodeWordChecker(String d) {
        val1 = 6;
        val2 = 20;
        word = d;
    }
    public boolean isValid(String str) {
        if ( str.length() <= val1 || str.length() >= val2) { // inclusive of val1 & val2
            return false;
        }
        if( str.indexOf(word) > 0) {
            return false
        }
        return true;
    }

}
```

3B

Write the complete `CodeWordChecker` class. Your implementation must meet all specifications and conform to all examples.

```
public class CodeWordChecker extends StringChecker ()
{

    private int min = 6;
    private int max = 20;
    private String inv;

    public CodeWord (int min, int max, String inv )
    {
        this.min = min;
        this.max = max;
        this.inv = inv;
    }


    public boolean isValid (String str)
    {
      if ( str.length() <= max && str.length() >= min && str.indexOf(inv) == -1)
        return true;
      else
        return false;
    }

}
```

3C

Write the complete `CodeWordChecker` class. Your implementation must meet all specifications and conform to all examples.

```
public class CodeWordChecker implements StringChecker {int min, int max,
                                                        String not}
{

    public CodeWordChecker {
        min = 6;
        max = 20;
        not = pass;
    }
    public CodeWordChecker (int n1, int n2, String n){
        min = n1;
        max = n2;
        not = n;
    }
    public boolean isValid (String str ) {
        if ( str.length() < min || str.length() > max )
                return false;
        for (int i = 0 ; i < str.length() - not.length(); i++) {
                if (str.substring(i, i + not.length()).equals(not))
                        return false;
        }
        return true;
    }

}
```


GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2018 SCORING COMMENTARY

Question 3

Overview

This question tested the student's ability to:

- Write program code to define a new type by creating a class;
- Write program code to create objects of a class and call methods; and
- Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

Students were asked to design the class `CodeWordChecker` so that it implements the method `isValid` in the given interface, `StringChecker`. In addition to demonstrating an understanding of class constructors 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 define the interface method `isValid` and expose it by declaring the method `public`. Students also had to recognize that two constructors needed to be included in order to conform to the examples given.

This question primarily addressed the fundamental skill of how to create a class so that it adheres to a specific interface. However, there was also a logic component required to make the class operational. The 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 method had to be defined and made publicly available.

Implementing the `isValid` method required students to find the length of a string and check if it was within a specified range and also to determine if one string was a substring of another. Students then needed to return an appropriate Boolean value based on the result of their logic.

Sample: 3A
Score: 8

In this solution point 1 was earned because the header is correctly declared. Point 2 was earned because all appropriate `private` instance variables are declared. Point 3 was earned because the 3-parameter constructor and the 1-parameter constructor headers are declared. Point 4 was earned because each of the parameters is used to initialize the instance variables. Point 5 was earned because the instance variables for minimum and maximum code length are initialized to `6` and `20`, respectively, and the `String` parameter is used to initialize the `String` instance variable. In the solution for the `isValid` method, point 6 was earned because the header is correctly declared. Point 7 was earned because the length of the `String` parameter is correctly compared to the minimum and maximum lengths. Point 8 was not earned because the value returned by the `indexOf` method is checked using `> 0` rather than `>= 0`. Point 9 was earned because the conditional statement returns `false` when given an invalid length, `false` when the instance variable is found in the `String` parameter, and `true` otherwise.

Sample: 3B
Score: 6

In this solution point 1 was not earned because the header uses `extends StringChecker`. Point 2 was earned because all appropriate `private` instance variables are declared. Point 3 was not earned because the response fails to declare the 1-parameter constructor. Point 4 was earned because each of the parameters is used to initialize the instance variables. Point 5 was not earned because the response fails to implement the

Question 3 (continued)

1-parameter constructor. In the solution for the `isValid` method, point 6 was earned because the header is correctly declared. Point 7 was earned because the length of the `String` parameter is determined to be between the minimum and maximum lengths, inclusive. Point 8 was earned because the response checks for the unwanted string by calling the `indexOf` method and comparing it to `-1`. Point 9 was earned because the conditional statement returns `false` when given an invalid length, `false` when the instance variable is found in the `String` parameter, and `true` otherwise.

Sample: 3C
Score: 2

In this solution point 1 was not earned because the header incorrectly includes parameters. Point 2 was not earned because no `private` instance variables are declared. Not declaring instance variables also resulted in the loss of points 4, 5, 7, and 8. Point 3 was not earned because the response fails to declare the 1-parameter constructor. Point 4 could have been earned if instance variables had been declared. In the solution for the `isValid` method, point 6 was earned because the header is correctly declared. Point 7 could have been earned because the length of the `String` parameter is compared to the minimum and maximum lengths. Point 8 was not earned because the instance variables are not declared. In addition, the point would not have been earned because the response, while traversing the `String` parameter one index at a time, fails to test the last occurrence of the substring. Point 9 was earned because the logic for finding an invalid code word is correct.