2025

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free-Response Question 2

- ☑ **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.*

2025 Digital Decision Rules

- Some non-ASCII characters are not printing correctly in ONE, particularly extended punctuation. Certain kinds of double-quotes may display as â□□; a character that displays as ï¼□ may be a parenthesis, comma, or semicolon (or other punctuation). If the badly displayed character makes sense as one of those, evaluate the response accordingly.
- If there are missing closed-double-quotes or closed-parentheses, assume they are at the end of the line where they opened, immediately before the semicolon or curly bracket (if any).
- Assume an open/left curly bracket `{` immediately after any method header or class header that does not already have one.
- Assume an appropriate amount of closing brackets before any method header to close all open brackets from the previous method or constructor.
- If there are missing curly brackets, clear indentation can "convey intent". Evaluate the response accordingly.
- Inside a method with left-justified code, indentation cannot "convey intent", so missing curly brackets cannot be assumed. Without bracketing or indentation, only the first line of a `while` / `if` / `for` is controlled by the statement; with open curly bracket and no indentation cues, the entire remainder of the method is "inside" the statement.

No Penalty

- `:` instead of `;` and vice versa
- `,` instead of `;` and vice versa

Question 2: Class — 9 points

Canonical solution

9 points

```
public class SignedText
{
   private String firstName;
   private String lastName;

   public SignedText(String first, String last)
   {
      firstName = first;
      lastName = last;
   }

   public String getSignature()
   {
      String sig = "";
      if (!firstName.equals(""))
      {
         sig += firstName.substring(0, 1) + "-";
      }
      sig += lastName;
      return sig;
   }

   public String addSignature(String textStr)
   {
      String sig = getSignature();
      int index = textStr.indexOf(sig);

      if (index == -1)
      {
         return textStr + sig;
      }
      else if (index == 0)
      {
         return textStr.substring(sig.length()) + sig;
      }
      else
      {
         return textStr;
      }
   }
}
```

`SignedText`

	Scoring Criteria	Decision Rules	
1	Declares class header: `class SignedText`	Responses **will not** earn the point if they • declare the class as `private` • include extraneous code outside the class • include `()` with or without parameters in the class header	**1 point**
2	Declares all appropriate `private` instance variable(s) and constructor initializes instance variable(s) using appropriate parameters	Responses **can** still earn the point even if they • create and store the signature using only one `String` instance variable Responses **will not** earn the point if they • declare any instance variables `static` • omit `private` in instance variable declaration • declare variables outside the class • declare an instance variable inside the constructor • fail to use either parameter • assign values to instance variables from an unknown source • assign initial value to local variable instead of instance variable, even if the local variables are declared as `private` • assign parameter(s) to variable(s) with incompatible data type • return a value from the constructor • define the constructor inside a method or define a method inside the constructor	**1 point**

3	Declares constructor header:
	`SignedText(String ___,`
	`String ___)`
		Responses **will not** earn the point if they
		• declare the constructor as `private` or `static`
			1 point

4	Declares method headers:
	`public String`
	`getSignature()`
	`public String`
	`addSignature(String ___)`
		Responses **will not** earn the point if they
		• omit `public` in either method header or declare either method as something other than `public`
		• use incorrect method name
		• use incorrect return or parameter type
			1 point

5	Compares first name to the empty string	Responses **can** still earn the point even if they • perform the comparison implicitly (e.g., by comparing the string's length to 0)	**1 point**

6	Determines appropriate signature string in both cases (*algorithm*)	Responses **can** still earn the point even if they • fail to return a value in some cases (*return from* `getSignature` *not assessed*) Responses **will not** earn the point if they • construct the correct string but return something else • define another method inside the signature method or vice versa	**1 point**
7	Calls `String` methods using correct syntax throughout the class	Responses **can** still earn the point even if they • call `substring` only one time • call `indexOf`, `contains`, `charAt`, or other appropriate methods Responses **will not** earn the point if they • only call `length` and/or `equals` without using other `String` methods • fail to call at least one `String` method which accesses elements of a string	**1 point**
8	Identifies the three required cases for `addSignature` using appropriate conditions	Responses **can** still earn the point even if they • return an incorrect string in one or more cases	**1 point**
9	`addSignature` returns appropriate `String` in all three cases (*algorithm*)	Responses **can** still earn the point even if they • identify one or more of the three cases incorrectly, as long as they are unambiguously no-match, match-start, and match-end • implement `getSignature` incorrectly Responses **will not** earn the point if they • call `getSignature` incorrectly, or incorrectly implement its functionality in `addSignature` • fail to identify three distinct cases • build correct strings but assign them to cases incorrectly • print the string instead of or in addition to returning it • define another method inside `addSignature` or vice versa	**1 point**

	Question-specific penalties
	None

Alternate canonical:

```
public class SignedText
{
   private String signature;

   public SignedText(String first, String last)
   {
      signature = last;
      if (first.length() > 0)
      {
         signature = first.substring(0, 1) + "-" + last;
      }
   }

   public String getSignature()
   {
      return signature;
   }

   public String addSignature(String textStr)
   {
      String result = textStr;
      int index = textStr.indexOf(signature);
      if (index < 0)
      {
         result += signature;
      }
      else if (index == 0)
      {
         result = result.substring(signature.length()) + signature;
      }
      return result;
   }
}
```

```
public class SignedText{
   private String firstName;
   private String lastName;

   public signedText(String x, String y){
      firstName = x;
      lastName = y;
   }

   public String getSignature(){
      if(firstName.length() > 0){
         return firstName.substring(0,1) + "-" + lastName;
      }
      return lastName;
   }
   public addSignature(String text){
      int len = text.length();
      int sig = getSignature.length();

      if(len > sig){

         if(text.substring(len - sig,len).equals(getSignature())){
            return text;
         }
         if(text.substring(0,sig).equals(getSignature())){
            return text.substring(len - sig,len) + getSignature();
         }
      }
      return text + getSignature();
   }
}
```

```
public class SignedText{
String firstName;
String lastName;

   public SignedText(theFirstName, theLastName){
      firstName = theFirstName;
      lastName = theLastName;
   }

   public String getSignature(){
      if(firstName.length() == 0){
         return lastName;
      }
      else{
         return firstName.substring(0, 1) + "-" + lastName;
      }
   }

   public String addSignature(String signature){
      int i = signature.indexOf(getSignature());
      //if not there
      if(i == -1){
         return signature + getSignature;
      }
      //if at start
      else if(i == 0){
         int s = getSignature().length();
         return signature.substring(s+1) + getSignature();
      }
      //if at end
      else{
         return signature;
      }
   }
}
```

```
public class SignedText(String firstName, String lastName){

   public String getSignature(){
      if(firstName.length >0){
         return firstName.substring(0) + "-" + lastName
      }
      else{
         return lastName;
      }
   }

   public String addSignature(String text){
   String signature = this.getSignature(firstName, lastName);
      if(text.indexOf(signature) = -1){
         return signature + signature.getSignature();
      }
      else{
         if(text.indexOf(signature) = 0){
            return text.substring(signature.length, text.length);
         }
         else{
            if(text.indexOf(signature) = text.length - signature.length -1){
               return text;
            }
      }
   }
}
```

Question 2

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

Overview

NEW for 2025: The question overviews can be found in the *Chief Reader Report on Student Responses* on AP Central.

Sample: 2A
Score: 7

Point 1 was earned because the class header is correct. The access modifier is `public` and specifies the class `SignedText` without parentheses. Point 2 was earned because all appropriate private instance variables are correctly declared and initialized in the constructor using the appropriate parameters. The parameters are correctly assigned to the instance variables. Point 3 was earned because the constructor header is correct. The incorrect casing of `signedText` is an example of a minor error for which no penalty is assessed, as noted on the *Applying the Scoring Criteria* page of the Scoring Guidelines. Point 4 was not earned because one method header is incorrect. The `addSignature` method is missing a `String` return type. Point 5 was earned because the response checks whether the first name is an empty string. The response performs the comparison implicitly by checking if the string's length is greater than zero `(firstName.length() > 0)`. Point 6 was earned by correctly determining the appropriate signature string based on the first name in both cases. Point 7 was earned because all `String` method calls are syntactically correct. Point 8 was earned because the response identifies the three required cases for `addSignature` using appropriate conditions. Note that if the signature is contained in the parameter, the parameter's length must be longer than the signature's length. The `if(len > sig)` condition has no adverse side effects. Point 9 was not earned because the response returns an incorrect `String` in the match-start case. The response incorrectly uses `text.substring(len - sig, len)` in the second `return` statement. This returns part of the original parameter followed by the signature. Note that the missing parentheses on the `getSignature` method call is an example of a minor error for which no penalty is assessed.

Sample: 2B
Score: 6

Point 1 was earned because the class header is correct. Point 2 was not earned because the instance variables `firstName` and `lastName` do not have the keyword `private`. Point 3 was not earned because the constructor header was declared without `String` types for the parameters. Point 4 was earned because both method headers are correct. Each method is declared `public`, with a `String` return type, and includes parentheses with and without parameters. Point 5 was earned because the response checks whether the first name is an empty string using `firstName.length() == 0`. Point 6 was earned by correctly determining the appropriate signature string based on the first name for both cases. Point 7 was earned because all `String` method calls are syntactically correct. Point 8 was earned because the response identifies the three required cases for `addSignature` using appropriate conditions. Point 9 was not earned because the response sometimes returns the incorrect `String`. In the match-start case, the response incorrectly calls `signature.substring(s + 1)`. This attempts to find the substring of the original parameter that begins after the signature but goes one character too far.

Question 2 (continued)

Sample: 2C
Score: 3

Point 1 was not earned because the class header includes parentheses and parameters. This is similar to the syntax for the relatively new record feature in Java, but unless the response includes the `record` keyword, the response as written (here, a class header) is incorrect. Point 2 was not earned because no `private` instance variables are declared. Point 3 was not earned because no constructor header is declared. Point 4 was earned because both method headers are correct. Each method is declared `public`, with a `String` return type, and includes parentheses with and without parameters. Point 5 was earned because the response checks whether the first name is an empty string. The response performs this comparison implicitly by checking whether the length of the string is greater than zero. Note that the missing parentheses on the `length` method call is a minor error for which no penalty is assessed. Point 6 was not earned because the response incorrectly identifies the first letter of the first name by using `firstName.substring(0)`. The one-parameter `substring` call returns the substring that begins at the specified index and continues to the end of the string, which is not the intended behavior. Point 7 was earned because all `String` method calls are syntactically correct. Point 8 was not earned because the response misidentifies the match-end case for `addSignature`. The match-end case uses `if(text.indexOf(signature) = text.length - signature.length -1)` to check that the signature is at the end of the parameter. However, subtracting `1` in the conditional does not produce the correct result. Note that the use of `=` instead of `==` is a minor error for which no penalty is assessed. Point 9 was not earned for the following reasons, either of which would cause the response to lose point 9. First, the statement `this.getSignature(firstName, lastName)` calls `getSignature` incorrectly and secondly, `return signature + signature.getSignature()` incorrectly returns a signature appended to a signature.