2024

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.*

Question 2: Class — 9 points

Canonical solution

9 points

```
public class Scoreboard
{
   private String team1Name, team2Name;
   private int whoseTurn;
   private int score1, score2;

   public Scoreboard(String team1, String team2)
   {
      team1Name = team1;
      team2Name = team2;
      whoseTurn = 1;
      score1 = 0;
      score2 = 0;
   }

   public void recordPlay(int points)
   {
      if (points == 0)
      {
         if (whoseTurn == 1)
         {
            whoseTurn = 2;
         }
         else
         {
            whoseTurn = 1;
         }
      }
      else
      {
         if (whoseTurn == 1)
         {
            score1 += points;
         }
         else
         {
            score2 += points;
         }
      }
   }

   public String getScore()
   {
      String result = score1 + "-" + score2 + "-";
      if (whoseTurn == 1)
      {
         result += team1Name;
      }
      else
      {
         result += team2Name;
      }
      return result;
   }
}
```

`Scoreboard`

	Scoring Criteria	Decision Rules	
1	Declares class header: `class Scoreboard`	Responses **will not** earn the point if they • declare the class as something other than `public`	**1 point**
2	Declares at least one `private String` instance variable and one `private int` instance variable	Responses **will not** earn the point if they • declare any instance variable `static` • declare a variable outside the class	**1 point**
3	Declares constructor header: `Scoreboard(String ___, String ___)` and constructor initializes both team name instance variables using parameters	Responses **can** still earn the point even if they • declare instance variable(s) outside the class, or in the class within a method or constructor Responses **will not** earn the point if they • fail to declare or initialize instance variables for both team names • declare the constructor as something other than `public`	**1 point**

4	Declares method headers:
	`public void recordPlay(int ___)`
	`public String getScore()`
		Responses **will not** earn the point if they
		• use incorrect method names
		• omit or declare incorrectly either method header
		• omit `public` in either method header or declare either method as something other than `public`
			1 point

5	Recording method checks for parameter value of zero	Responses **can** still earn the point even if they • use a method name inconsistent with the examples, as long as it is recognizably equivalent	**1 point**
6	Recording method increases at least one declared instance variable representing one team's score	Responses **can** still earn the point even if they • declare any instance variable incorrectly, outside the class, or in the class within a method or constructor • use something other than the parameter to update the instance variable • use a method name inconsistent with the examples, as long as it is recognizably equivalent	**1 point**

7	Recording method switches active team	Responses **can** still earn the point even if they • perform the switch in a method other than the recording method • store the switched active team in a local variable, as long as the switch occurs in both active team cases • use a method name inconsistent with the examples, as long as it is recognizably equivalent • perform the switch when the parameter is not zero	**1 point**
8	Recording method adds correct number of points to the active team's score (*algorithm*)	Responses **can** still earn the point even if they • fail to switch active team correctly • declare an instance variable that holds a team's score outside the class, or in the class within a method or constructor • use a method name inconsistent with the examples, as long as it is recognizably equivalent Responses **will not** earn the point if they • switch teams when the parameter is positive • fail to declare an instance variable to track the active team, initialize it incorrectly, or never change its value • add correct number of points for only one team • increase score by something other than the parameter • fail to declare instance variables to hold both teams' scores	**1 point**
9	Accessor method builds and returns specified string (*algorithm*)	Responses **can** still earn the point even if they • fail to declare instance variables and use variables from constructor or methods within the class • use a method name inconsistent with the examples, as long as it is recognizably equivalent Responses **will not** earn the point if they • omit the literal hyphens in the constructed string	**1 point**
			Total for question 2 **9 points**

Alternate canonical:

```
public class Scoreboard
{
   private String team1Name, team2Name;
   private boolean isTeam1Active;
   private int score1, score2;

   public Scoreboard(String team1, String team2)
   {
      team1Name = team1;
      team2Name = team2;
      isTeam1Active = true;
      score1 = 0;
      score2 = 0;
   }

   public void recordPlay(int score)
   {
      if (score == 0)
      {
         isTeam1Active = !isTeam1Active;
      }
      else if (isTeam1Active)
      {
         score1 += score;
      }
      else
      {
         score2 += score;
      }
   }

   public String getScore()
   {
      String result = score1 + "-" + score2 + "-";
      if (isTeam1Active)
      {
         result += team1Name;
      }
      else
      {
         result += team2Name;
      }
      return result;
   }
}
```

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 class Scoreboard {
  private int tOne = 0;
  private int tTwo = 0;
  private String tOneName = "";
  private String tTwoName = "";
  private boolean whosActive = true;
  public void Scoreboard (String nameOne, String nameTwo) {
     tOneName = nameOne;
     tTwoName = nameTwo;
  }
  public void recordPlay (int score) {
     if (whosActive == true ) {
        if (score == 0) {
           whosActive == false;
        }
        else {
           tOne += score;
        }
     }else {
        if (score == 0) {
           whosActive == true;
        }
        else {
           tTwo += score;
        }
     }
  }
```


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.

```
public String getScore() {
    if (whosActive == true) {
        System.out.print(tOne + "-" + tTwo + "-" + tOneName)
    }
    else {
        System.out.print(tOne + "-" + tTwo + "-" + tTwoName)
    }
}
}
```

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 class Scoreboard
{
  Private String team1;
  Private String team2;
  Private int n;
  Private int n1;
  Public void Scoreboard(String a, String b){

  team1 = a;
  team2 = b;
 }

 Public void recordPlay(int c){
   IFC c != 0)
     n+=c;

   else
     n1 += c;
 }

 Public void getScore(){
   if (n >= n1)
     System.out.println(n+"_"+n1+team1);
 else
   System.out.println(n+"_"+n1+ team2);
}
```


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.

```
Public class Scoreboard {
Private String team = "";
private String team2 = "";

Public Scoreboard (String team, String p){
this.team = team;
team2 = p;
}
Public int recordPlay(int x){
if(team1 == true && x=0){
team1 == false;
}
if(team1 == true){
String team t=x;

}
if(team1 == false && x=0{
  team1 = true;
}  if(team1 == false{
     String p = x;

Public boolean team1=
true;

public String getScore(){
return team t+"-"+
p++;
}
```


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

Question 2

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 define a new type by creating a class (Skill 3.B).
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements (Skill 3.C).

This question assessed the ability to design an entire class, with constructor, instance variables, and methods that access and update them; to maintain and update the state of the object according to a specified algorithm; and to build a specified string based on the state of the object.

Students were asked to design a class, `Scoreboard`, to hold and modify data for a game with two teams. This class must contain instance variables to store the names of each team, the scores of each team, and an indicator of which team is current. It also requires a constructor and two methods, each matching a provided specification. In designing the `Scoreboard` class, students were expected to understand and demonstrate how to construct appropriate headers for the class itself and for its instance methods. Students were also expected to understand and correctly use instance variables, including declaration of the variables, initialization of the variables, and use of the variables inside the constructor and methods.

The prompt specifications and example code of the class required students to implement two methods. The first method specified, `recordPlay`, is passed a single parameter representing points scored by the active team. If the point count is positive, then the active team's score variable is increased. If the point count is zero, then the active team is switched to the other team. The method must determine which team is the active team and increase the score for that team only and must successfully update the indicator of the current team, correctly in either direction and only when zero points are scored. The second method, `getScore`, constructs and returns a specified `String` that contains team scores, string literals (hyphens), and the name of the active team (which may already be stored in a variable or may be chosen in a conditional statement).

Sample: 2A
Score: 7

Point 1 was earned because the class header is correct. To earn this point, the header must contain `class Scoreboard` and may optionally contain the `public` visibility specifier. Point 2 was earned because the response declares at least one `private String` instance variable (`tOneName` and `tTwoName`) and at least one `private int` instance variable (`tOne` and `tTwo`). Note that the instance data is initialized at declaration, which is valid. This point would not be earned if the declaration of any instance variable contained the keyword `static`. Point 3 was not earned because the constructor has a `void` return type. The focus of this point is a correct constructor header and initialization of the team names, `tOneName` and `tTwoName`, using the parameters. This point can still be earned if the instance variables are improperly declared inside the constructor, in another method, or outside of the class. Point 4 was earned because both the `recordPlay` and `getScore`

Question 2 (continued)

method headers are correct. The focus of this point is syntactically correct method headers for both methods. Point 5 was earned because the recording method, `recordPlay`, checks for a parameter value of zero (in this case, `score == 0`). Point 6 was earned because the recording method increases both `tOne` and `tTwo`, although increasing only one of the two instance variables would have been enough to earn this point. Point 7 was earned because the recording method switches the active team using a `boolean` instance variable, `whosActive`. Note that the use of `==` instead of `=` (or vice versa) is one of the minor errors for which no penalty is assessed on this exam. (See the "No Penalty" section of the Scoring Guidelines for a complete list.) The focus of this point is the logic of switching the active team back and forth between the two teams, with at most one switch per method call. Point 8 was earned because the recording method adds the correct number of points to the active team's score (either `tOne` or `tTwo`) and because the variable to track the active team, `whosActive`, was properly initialized during declaration and is switched when the parameter is zero. The focus of this point is a correct algorithm. The algorithm updates the active team's score when the parameter is greater than zero and switches the active team when the parameter is zero. This point can still be earned even if the response loses point 7 by failing to switch the active team correctly, as long as the response does not switch teams on a positive parameter. Point 9 was not earned because the response prints the specified string to output rather than returning it, which is an algorithm error. The focus of this point is to build and return the specified string, including both string literal hyphens. Note that penalty point (w) from the "1-Point Penalty" list in the Scoring Guidelines is not applied.

Sample: 2B
Score: 4

Point 1 was earned because the class header is correct. The access for the class header is `public` and the response specifies a class name of `Scoreboard`. Point 2 was earned because at least one `private String` instance variable (`team1` and `team2`) and at least one `private int` instance variable (`n` and `n1`) was declared. Point 3 was not earned because the constructor header has a `void` return type. The constructor header has the correct access, `public`, and the correct name, `Scoreboard`. Point 4 was not earned because although the `recordPlay` method header is correct, the `getScore` method has an incorrect `void` return type. Point 5 was earned because the recording method, `recordPlay`, checks for a parameter value of zero. Note that even though the score update is incorrect in the `if` statement, the response still earns this point. Point 6 was earned because the recording method increases at least one instance variable representing one team's score, even though the update is incorrect because the response did not check for the active team. Point 7 was not earned because there is no active team designation and no team switch occurred. Point 8 was not earned because the response fails to declare an instance variable to track the active team, fails to update the active team, and updates only one team's score. Point 9 was not earned because the response prints the string to output instead of returning the string. The string is also built improperly since there is a missing second hyphen and the string contains the team with the highest score rather than the active team.

Question 2 (continued)

Sample: 2C
Score: 2

Point 1 was not earned because the class header contains parentheses. The access for the class header is `public` and the response specifies a class name of `Scoreboard`. Point 2 was not earned because the response fails to declare any `int` instance variables to hold scores. Point 3 was earned because the constructor header is correct with `public` access and the name `Scoreboard`. The response then correctly initializes both team name instance variables, `team` and `team2`, using the parameters. Point 4 was not earned because the `recordPlay` method has an incorrect return type. Point 5 was earned because the recording method checks whether the parameter `x` has a value of zero. Note that the use of `=` instead of `==` (and vice versa) is one of the minor errors for which no penalty is assessed on this exam. (See the "No Penalty" section of the Scoring Guidelines for a complete list.) Point 6 was not earned because the recording method attempts to concatenate the parameter `x` with a local `String` variable, `team`, instead of adding the parameter to an integer variable containing a score. The variable being changed, `team`, represents a team name, not a team score. Point 7 was not earned because, when the recording method sets `team1` to `false`, a subsequent `if` statement switches it back to `true`. Point 8 was not earned because the response fails to declare instance variables to hold both teams' scores and therefore does not add points to the scores correctly. Point 9 was not earned because, while the response returns a `String`, it is not in the specified format and the accessor method does not test for the active team.