AP® COMPUTER SCIENCE A
2014 GENERAL 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.

1-Point Penalty

(w) Extraneous code that causes side effect (*e.g., writing to output, failure to compile*)

(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

- o Extraneous code with no side effect (*e.g., precondition check, no-op*)
- o Spelling/case discrepancies where there is no ambiguity*
- o Local variable not declared provided other variables are declared in some part
- o `private` or `public` qualifier on a local variable
- o Missing `public` qualifier on class or constructor header
- o Keyword used as an identifier
- o Common mathematical symbols used for operators ($\times$ • $\div$ $\leq$ $\geq$ <> $\neq$)
- o `[]` vs. `()` vs. `<>`
- o `=` instead of `==` and vice versa
- o Array/collection access confusion (`[] get`)
- o `length/size` confusion for array, `String`, `List`, or `ArrayList`, with or without `( )`
- o Extraneous `[]` when referencing entire array
- o `[i,j]` instead of `[i][j]`
- o Extraneous size in array declaration, *e.g.,* `int[size] nums = new int[size];`
- o Missing `;` provided majority are present and indentation clearly conveys intent
- o Missing `{ }` where indentation clearly conveys intent and `{ }` are used elsewhere
- o Missing `( )` on parameter-less method or constructor invocations
- o 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* "`Bug bug;`", *then uses* "`Bug.move()`" *instead of* "`bug.move()`", *the context does* ***not*** *allow for the reader to assume the object instead of the class.*

AP® COMPUTER SCIENCE A
2014 SCORING GUIDELINES

Question 4: Trio

Class: `Trio`	**9 points**

Intent: *Define implementation of* `MenuItem` *interface that consists of sandwich, salad, and drink*

+1 `public class Trio implements MenuItem`

+1 Declares appropriate `private` instance variables

+2 Implements constructor

- **+1** `public Trio(Sandwich` *`sandwich`*`, Salad` *`salad`*`, Drink` *`drink`*`)`
- **+1** Initializes appropriate instance variables using parameters

+1 Implements interface methods
(`public String getName(){…}`, `public double getPrice(){…}`)

+1 Constructs correct name string and makes available for return in `getName`

+1 Returns constructed name string in `getName`

+1 Computes correct price and makes available for return in `getPrice`

+1 Returns computed price in `getPrice`

Question-Specific Penalties

-0 Missing or extra spaces in name string, "trio"

-1 (w) Extraneous default constructor that causes side effect

Question 4: Trio

```
public class Trio implements MenuItem {
   private Sandwich sandwich;
   private Salad salad;
   private Drink drink;

   public Trio(Sandwich s, Salad sal, Drink d){
      sandwich = s;
      salad = sal;
      drink = d;
   }

   public String getName(){
      return sandwich.getName() + "/" + salad.getName() + "/" +
         drink.getName() + " Trio";
   }

   public double getPrice(){
      double sandwichPrice = sandwich.getPrice();
      double saladPrice = salad.getPrice();
      double drinkPrice = drink.getPrice();
      if (sandwichPrice <= saladPrice && sandwichPrice <= drinkPrice)
         return saladPrice + drinkPrice;
      else if (saladPrice <= sandwichPrice && saladPrice <= drinkPrice)
         return sandwichPrice + drinkPrice;
      else
         return sandwichPrice + saladPrice;
   }
}
```

Alternate

```
public class Trio implements MenuItem {
   private String name;
   private double price;

   public Trio(Sandwich s, Salad sal, Drink d){
      double sandwichPrice = s.getPrice();
      double saladPrice = sal.getPrice();
      double drinkPrice = d.getPrice();
      if (sandwichPrice <= saladPrice && sandwichPrice <= drinkPrice)
         price = saladPrice + drinkPrice;
      else if (saladPrice <= sandwichPrice && saladPrice <= drinkPrice)
         price = sandwichPrice + drinkPrice;
      else
         price = sandwichPrice + saladPrice;
      name = s.getName()+ "/" + sal.getName()+ "/" + d.getName()+ " Trio";
   }
```

Question 4: Trio continued

```
   public String getName(){
      return name;
   }

   public double getPrice(){
      return price;
   }
}
```

4A

Write the complete `Trio` class below.

```
public class Trio implements MenuItem
{
   private Sandwich  sand;
   private Salad     sal;
   private Drink    dri;
   public Trio( Sandwich a, salad b, Drink c)
   {
       sand = a;
       sal = b;
       dri = c;
   }
   public  String getName()
   {
      return  sand.getName() + "/"+ sal.getName() + "/"+ dri.getName();
   }
   public  double getPrice()
   {
      double p1 = sand.getPrice();
      double p2 : ~~sand~~. sal.getPrice();
      double p3 = dri.getPrice();
      if ( p2>=p1 && p3>=p1)
         return p2 + p3;
      else if (p1>=p2 && p3>=p2)
          return p1 + p3;
      return p1 + p2;
   }
}
```

GO ON TO THE NEXT PAGE.

4B

Write the complete Trio class below.

```
public class Trio implements MenuItem
{
    Sandwich sandwich;
    Salad salad;
    Drink drink;
    public Trio (Sandwich s, Salad sa, Drink d)
    {
        sandwich=s;
        salad=sa;
        drink=d;
    }
    public String getName()
    {
        return sandwich.getName() + salad.getName()+ drink.getName();
    }
    public double getPrice()
    {
        double a= sandwich.getPrice();
        double b= drink.getPrice();
        double c= salad.getPrice();      if (b==c){    // forgot to include
          if (a>b && a>c){                 return a+b; // conditional
             if (b>c) {                    } else
               return a+b;
             }
             else{
               return a+c;
             }
          }
          else if (a>b && a<c) {
             return a+c;
          }
          else if (a<b && a<c) {
             return b+c;
          }
          else if (a<b && a>c) {
             return a+b;
          }
}
```

Unauthorized copying or reuse of any part of this page is illegal.

GO ON TO THE NEXT PAGE.

4C

Write the complete `Trio` class below.

```
public class Trio implements MenuItem {
        public Trio (Sandwich sandwich, Salad salad, Drinl drink) {
        }

        public String toString(Sandwich sandwich, Salad salad, Drink drink){
          return sandwich.getName() + "/" + salad.getName() + "/" + drink.getName() + " Trio";
        }
        public double trioPrice(Sandwich sandwich, Salad salad, Drink drink){
            int mid = Math.max (sandwich, salad);
            int max = Math.max( mid, drink );
            return mid + max ;
          }
```

GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2014 SCORING COMMENTARY

Question 4

Overview

This question evaluated the ability of a student to read an interface and then define a class that implements that interface. As part of the implementation students would define a constructor with a specified order of parameters to instantiate a `Trio` object. Students were also required to include code to construct a `String` object from the supplied parameters of the component `MenuItem` objects within the `Trio`. This string was composed of the result of calling `getName()` on each `MenuItem` object in turn separated by "/" and ending with the `String` "trio". The resulting string was to be returned in the interface method `getName`. Students were also to compute the price of the `Trio` by identifying the lowest price of the three component items and excluding that amount from the total price of the `Trio` to be returned in the interface method `getPrice`.

Sample: 4A
Score: 8

The student correctly creates the class header of `Trio` implementing the interface `MenuItem`. The instance variables for a sandwich, salad and drink were declared as private. The student implements the constructor to create a Trio with the parameters in the correct order. The instance variables were initialized using the parameters from the constructor. The student implements the two interface methods with code in the body of the methods. The name of the Trio was not constructed correctly leaving off the required string `"Trio"` at the end of the concatenation of the sandwich, salad and drink names. The `getName` method returned the constructed string using the instance variables. The price is computed correctly and was returned in the method `getPrice`. The response earned 8 points.

Sample: 4B
Score: 6

The student correctly creates the class header of `Trio` implementing the interface `MenuItem`. The instance variables for a sandwich, salad and drink were declared without private; therefore, the point for the declaration of variables was not earned. The student implements the constructor to create a Trio with the parameters in the correct order. The instance variables were initialized using the parameters from the constructor. The student implements the two interface methods with code in the body of the methods. The name of the Trio was not constructed correctly leaving off the required string `"Trio"` at the end of the concatenated sandwich, salad and drink names. The price is not computed correctly so the calculation point was not earned. The student did compute a price using the sandwich, salad, drink and it was returned in the method `getPrice`. The response earned 6 points.

Sample: 4C
Score: 2

The student correctly creates the class header of `Trio` implementing the interface `MenuItem`. The instance variables were not declared. The student declaration of the constructor is correct. There were no instance variables to initialize. The student does not implement the two interface methods. The student writes an additional public method with parameters that could provide incorrect names for the Trio, losing the "constructs correct name string" point. Although the constructed name was available, it was not returned in a `getName` method. The student writes an additional public method with parameters that could provide an incorrect price for the Trio object earning no point for "computes correct price." The student solution did not return the computed price in `getPrice`. The response earned 2 points.