Apply the question-specific rubric first. To maintain scoring intent, a single error is generally accounted for only once per question thereby mitigating multiple penalties for the same error. The error categorization below is for cases not adequately covered by the question-specific rubric. Note that points can only be deducted if the error occurs in a part that has earned credit via the question-specific rubric. Any particular error is penalized only once in a question, even if it occurs on different parts of that question.

### Nonpenalized Errors
- spelling/case discrepancies if no ambiguity*
- local variable not declared if others are declared in some part
- use keyword as identifier
- [] vs. () vs. <>
- = instead of == (and vice versa)
- length/size confusion for array, String, and ArrayList, with or without ()
- private qualifier on local variable
- extraneous code with no side effect; e.g., precondition check
- common mathematical symbols for operators (x • ÷ ≤ ≥ < >) ≠
- missing {} where indentation clearly conveys intent and {} used elsewhere
- default constructor called without parens; e.g., new Fish;
- missing () on parameterless method call
- missing () around if/while conditions
- missing ; when majority are present
- missing public on class or constructor header
- extraneous [] when referencing entire array
- extraneous size in array declaration, e.g., int[size] nums = new int[size];

### Minor Errors (1/2 point)
- confused identifier (e.g., len for length or left() for getLeft())
- local variables used but none declared
- modifying new in constructor call
- use equals or compareTo method on primitives, e.g., int x; ...x.equals(val)
- array/collection access confusion ([] get)
- assignment dyslexia, e.g., x + 3 = y; for y = x + 3;
- super(method()) instead of
  super.method()
- formal parameter syntax (with type) in method call, e.g., a = method(int x)
- missing public from method header when required
- "false"/"true" or 0/1 for boolean values
- "null" for null

### Major Errors (1 point)
- extraneous code that causes side effect; e.g., information written to output
- interface or class name instead of variable identifier; e.g., Bug.move() instead of aBug.move()
- aMethod(obj) instead of obj.aMethod()
- attempt to use private data or method when not accessible
- destruction of persistent data (e.g., changing value referenced by parameter)
- use class name in place of super in constructor or method call
- void method (or constructor) returns a value

---

*Spelling and case discrepancies for identifiers fall under the “nonpenalized” category only if the correction can be unambiguously inferred from context; for example, "ArayList" instead of "ArrayList". As a counter example, note that if a student 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.
Question 2: APLine

**Intent:** Design complete APLine class including constructor, getSlope and isOnLine methods

+1 Complete, correct header for APLine [class APLine]

*Note: Accept any visibility except private*

+1 1/2 State maintenance

+1/2 Declares at least one instance variable capable of maintaining numeric value

+1/2 Declares at least three instance variables capable of maintaining numeric values

+1/2 All state variables have private visibility

*Note: Accept any numeric type (primitive or object)*

*Note: Accept any distinct Java-valid variable names*

+1 1/2 APLine Constructor

*Method header*

+1/2 Correctly formed header (visibility not private; name APLine)

+1/2 Specifies exactly three numeric parameters

*Method body*

+1/2 Sets appropriate state variables based on parameters (no shadowing errors)

*Note: Interpret instance fields by usage not by name*

+2 1/2 getSlope

*Method header*

+1/2 Correct method header

(visibility not private; type double or Double; name getSlope; parameterless)

*Method body*

+1/2 Computation uses correct formula for slope

+1 Computation uses double precision (no integer division)

+1/2 Returns computed value

+2 1/2 isOnLine

*Method header*

+1/2 Correct formed header (visibility not private; type boolean or Boolean, name isOnLine)

+1/2 Specifies exactly two numeric parameters

*Method body*

+1/2 Computation uses correct formula involving state and parameters

(a*x + b*y + c)

+1/2 Computation uses correct comparison test (equal to zero)

+1/2 Returns true if is on this APLine; false otherwise

© 2010 The College Board.
Visit the College Board on the Web: www.collegeboard.com.
public class APLine {
    /** State variables. Any numeric type; object or primitive. */
    private int a, b, c;

    /** Constructor with 3 int parameters. */
    public APLine(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    /** Determine the slope of this APLine. */
    public double getSlope() {
        return (- (this.a / (double) this.b));
    }

    /** Determine if coordinates represent a point on this APLine. */
    public boolean isOnLine(int x, int y) {
        return (0 == (this.a * x) + (this.b * y) + this.c);
    }
}

// Alternative solution (state variables of type double):
public class APLine {
    private double a1, b1, c1;

    public APLine(int a, int b, int c) {
        this.a1 = a;
        this.b1 = b;
        this.c1 = c;
    }

    public double getSlope() {
        return -(this.a1 / this.b1);
    }

    public boolean isOnLine(int x, int y) {
        return (0 == (this.a1 * x) + (this.b1 * y) + this.c1);
    }
}
public class ALine {
    private double a;
    private double b;
    private double c;

    public ALine(double x, double y, double z) {
        a = x;
        b = y;
        c = z;
    }

    public double getSlope() {
        return (-a)/b;
    }

    public boolean isOnLine(double x, double y) {
        double point = (a*x) + (b*y) + c;
        if (point == 0.0) {
            return true;
        }
        return false;
    }
}
public class ALine {

    private int A;
    private int B;
    private int C;

    public ALine(int A, int B, int C) {
        this.A = A;
        this.B = B;
        this.C = C;
    }

    public int getSlope() {
        return A / B;
    }

    public boolean isOnLine(int x, int y) {
        if ((A * x + (B * y) + C) == 0) {
            return true;
        }
        return false;
    }

}
public class ALine {
    public ALine(int a, int b, int c)
    {
    }

do public void ALine(int a, int b, int c)
{
}

    public int getSlope(int a, int b)
    {
        double slope = (-a) / b;
        return slope;
    }

    public boolean isOnLine(int a, int b, int c, int x, int y)
    {
        int total = a * x + b * y + c;
        if (total == 0)
            return true;
        return false;
    }
}
Question 2

Overview

This question focused on the design of a complete class, including state variables (instance data), a constructor, and instance methods. Students were provided with specifications for the APLine class, which included private state variables capable of holding numeric values, a constructor, and getSlope and isOnLine methods. They were required to implement the entire class, including the class header. Formulas were provided for values that needed to be computed. The question tested class definition and construction, method implementation, and knowledge of how to address unwanted truncation of decimal digits that would result from integer division.

Sample: 2A
Score: 9

The student earned 1 point for the correctly formed class header for APLine and two ½ points for declaring at least three instance variables that are capable of maintaining a numeric value. The student earned ½ point for providing the state variables with private visibility.

The student earned ½ point for the correctly formed method header for the class constructor. The header contains exactly three numeric parameters, earning ½ point. The student earned ½ point for setting the state variables appropriately.

The student earned ½ point for the correctly formed method header for getSlope and ½ point for using the correct formula for slope. The instance variables are of type double so a conversion is not needed, earning 1 point. The student earned ½ point for returning the computed value.

The student earned ½ point for the correctly formed method header for isOnLine and ½ point for declaring exactly two numeric parameters. The student earned both the ½ point for using the correct formula with the correct state variables and the correct parameter variables and the ½ point for comparing the computation for the line to zero. Issues related to possible integer overflow are ignored. The student earned ½ point for returning true if the point represented by its two parameters (x and y, in that order) is on the APLine and returning false otherwise.

Sample: 2B
Score: 6

The student did not earn 1 point for the incorrectly formed class header for APLine owing to the use of parentheses. The student earned two ½ points for declaring at least three instance variables that are capable of maintaining a numeric value and ½ point for providing the state variables with private visibility.

The student earned ½ point for the correctly formed method header for the class constructor. The header contains exactly three numeric parameters, earning ½ point. The student earned ½ point for setting the state variables appropriately.
The student did not earn ½ point for the method header for `getSlope` because of the incorrect return type. The formula for slope is incorrect; thus, the student did not earn that ½ point. The computation for double precision is missing; thus, the student did not earn 1 point. The student earned ½ point for returning the computed value.

The student earned ½ point for the correctly formed method header for `isOnLine` and ½ point for declaring exactly two numeric parameters. The student earned ½ point for using the correct formula with the correct state variables and the correct parameter variables, and ½ point for comparing the computation for the line to zero. Issues related to possible integer overflow are ignored. The student earned ½ point for returning `true` if the point represented by its two parameters (x and y, in that order) is on the `APLine` and returning `false` otherwise.

Sample: 2C  
Score: 3

The student did not earn 1 point because the class header for `APLine` is formed incorrectly owing to the addition of a parameter list. The student does not declare at least three instance variables that are capable of maintaining a numeric value, so these two ½ points were not earned. The student does not set state variables with private visibility, so that ½ point was not earned.

The method header for the class constructor should not include a type; consequently, the student did not earn that ½ point. The header contains exactly three numeric parameters, earning ½ point. The state variables are not set appropriately, so that ½ point was not earned.

The method header for `getSlope` has the incorrect return type as well as including a parameter list. Consequently, the student did not earn that ½ point. The student uses the correct formula for slope, earning ½ point. The computation for double precision is missing, so that 1 point was not earned. The student returns the computed value, earning ½ point.

The student earned ½ point for the correctly formed method header for `isOnLine`. The student does not declare exactly three numeric parameters, so that ½ point was not earned. The state variables are not used in the formula, so that ½ point was not earned. The student earned ½ point for comparing the computation for the line to zero. The student also earned ½ point for returning `true` if the point represented by its two parameters (x and y, in that order) is on the `APLine` and returning `false` otherwise.