

Question 1: Methods and Control Structures**9 points****Canonical solution**

a.

```
public int walkDogs(int hour)
{
    int dogsToWalk = company.numAvailableDogs(hour);

    if (dogsToWalk > maxDogs)
    {
        dogsToWalk = maxDogs;
    }

    company.updateDogs(hour, dogsToWalk);
    return dogsToWalk;
}
```

4 points

b.

```
public int dogWalkShift(int startHour, int endHour)
{
    int totalPay = 0;

    for (int hour = startHour; hour <= endHour; hour++)
    {
        int dogs = walkDogs(hour);
        int hourPay = 5 * dogs;
        if (dogs == maxDogs || (hour >= 9 && hour <= 17))
        {
            hourPay += 3;
        }
        totalPay += hourPay;
    }
    return totalPay;
}
```

5 points

a. walkDogs

Scoring Criteria		Decision Rules	
1	Calls <code>DogWalkCompany</code> method(s) on <code>company</code>	Responses can still earn the point even if they - fail to save or use the returned value - call <code>numAvailableDogs</code> more than once - only call one of the methods Responses will not earn the point if they - use the incorrect parameter types - call either <code>numAvailableDogs</code> or <code>updateDogs</code> on nothing or on something other than <code>company</code>	1 point
2	Compares available dogs and <code>maxDogs</code> to determine number of dogs to walk	Responses can still earn the point even if they - call <code>numAvailableDogs</code> incorrectly - calculate an incorrect number of dogs that were walked	1 point
3	Calls <code>numAvailableDogs</code> with <code>hour</code> and <code>updateDogs</code> with <code>hour</code> and correct number of dogs to walk (<i>algorithm</i>)	Responses can still earn the point even if they - call either method on nothing or on something other than <code>company</code> Responses will not earn the point if they - calculate an incorrect number of dogs that were walked	1 point
4	Returns calculated value	Responses can still earn the point even if they - calculate an incorrect number of dogs that were walked - call <code>numAvailableDogs</code> multiple times Responses will not earn the point if they - return something other than an <code>int</code> - fail to return a value in some cases - print a value in addition to or instead of returning it	1 point

b. `dogWalkShift`

Scoring Criteria		Decision Rules	
5	Loops from <code>startHour</code> to <code>endHour</code> , inclusive	Responses can still earn the point even if they return from the loop early, as long as bounds would otherwise be correct	1 point
6	Calls <code>walkDogs</code> with <code>int</code> parameter (<i>in the context of a loop</i>)	Responses can still earn the point even if they - fail to save or use the returned value Responses will not earn the point if they - use an incorrect parameter type on any call to <code>walkDogs</code> - call the method on the class or on an object other than <code>this</code> (use of <code>this</code> is optional)	1 point
7	Calculates base pay for an hour	Responses can still earn the point even if they - fail to include the calculation in the loop - call <code>walkDogs</code> incorrectly	1 point
8	Calculates possible bonus for an hour's pay	Responses can still earn the point even if they - call <code>walkDogs</code> multiple times inside the loop - fail to include the calculation in the loop Responses will not earn the point if they - count the bonus twice when both conditions are true - count the bonus only when both conditions are true - count the bonus when it has not been earned - calculate bonus incorrectly	1 point
9	Accumulates total pay (<i>algorithm</i>)	Responses can still earn the point even if they - calculate base pay or bonus incorrectly - fail to return total pay (<i>return not assessed in this part</i>) Responses will not earn the point if they - fail to initialize variable used for total pay - call <code>walkDogs</code> multiple times inside the loop - do not accumulate base and bonus in the context of a loop - return from the loop early	1 point
Question-specific penalties			
None			

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 ($\times$ • $\div$ $\leq$ $\geq$ $<>$ $\neq$)
- `[]` 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, “ArrayList” 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 1: Methods and Control Structures**9 points****Canonical solution****(a)**

```
public int getScore()  
{
```

4 points

```
    int score = 0;  
  
    if (levelOne.goalReached())  
    {  
        score = levelOne.getPoints();  
  
        if (levelTwo.goalReached())  
        {  
            score += levelTwo.getPoints();  
  
            if (levelThree.goalReached())  
            {  
                score += levelThree.getPoints();  
            }  
        }  
    }  
  
    if (isBonus())  
    {  
        score *= 3;  
    }  
  
    return score;  
}
```

(b)

```
public int playManyTimes(int num)  
{
```

5 points

```
    int max = 0;  
  
    for (int i = 0; i < num; i++)  
    {  
        play();  
        int score = getScore();  
        if (score > max)  
        {  
            max = score;  
        }  
    }  
  
    return max;  
}
```

(a) `getScore`

Scoring Criteria		Decision Rules	
1	Calls <code>getPoints</code> , <code>goalReached</code> , and <code>isBonus</code>	Responses will not earn the point if they - fail to call <code>getPoints</code> or <code>goalReached</code> on a <code>Level</code> object - call <code>isBonus</code> on an object other than <code>this</code> (use of <code>this</code> is optional) - include parameters	1 point
2	Determines if points are earned based on <code>goalReached</code> return values	Responses can still earn the point even if they - calculate the score total incorrectly - call <code>goalReached</code> incorrectly - fail to distinguish all cases correctly Responses will not earn the point if they - fail to use a nested <code>if</code> statement or equivalent	1 point
3	Guards update of score for bonus game based on <code>isBonus</code> return value	Responses can still earn the point even if they - triple the calculated score incorrectly - update the score with something other than tripling - call <code>isBonus</code> incorrectly Responses will not earn the point if they - use the <code>isBonus</code> return value incorrectly	1 point
4	Initializes and accumulates appropriate score (<i>algorithm</i>)	Responses can still earn the point even if they - call methods incorrectly, as long as method calls are attempted - fail to return the score (<i>return is not assessed</i>) Responses will not earn the point if they - calculate the score total incorrectly - triple the calculated score incorrectly	1 point
Total for part (a)			4 points

(b) `playManyTimes`

Scoring Criteria		Decision Rules	
5	Loops <code>num</code> times	Responses can still earn the point even if they return early	1 point
6	Calls <code>play</code> and <code>getScore</code>	Responses will not earn the point if they - call either method on an object other than <code>this</code> (use of <code>this</code> is optional) - include parameters	1 point
7	Compares a score to an identified max or to another score	Responses can still earn the point even if they - make the comparison outside the loop - call <code>getScore</code> incorrectly - fail to call <code>play</code> between calls to <code>getScore</code>	1 point
8	Identifies the maximum score (<i>algorithm</i>)	Responses will not earn the point if they - fail to initialize the result variable - compare a score to an identified max or to another score outside the loop - fail to call <code>play</code> exactly once each time through the loop	1 point
9	Returns identified maximum score	Responses can still earn the point even if they - calculate the maximum score incorrectly Responses will not earn the point if they - assign a value to the identified maximum score without any loop or logic to find the maximum	1 point
Total for part (b)			5 points
Question-specific penalties			
None			
Total for question 1			9 points

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, “ArrayList” 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.*