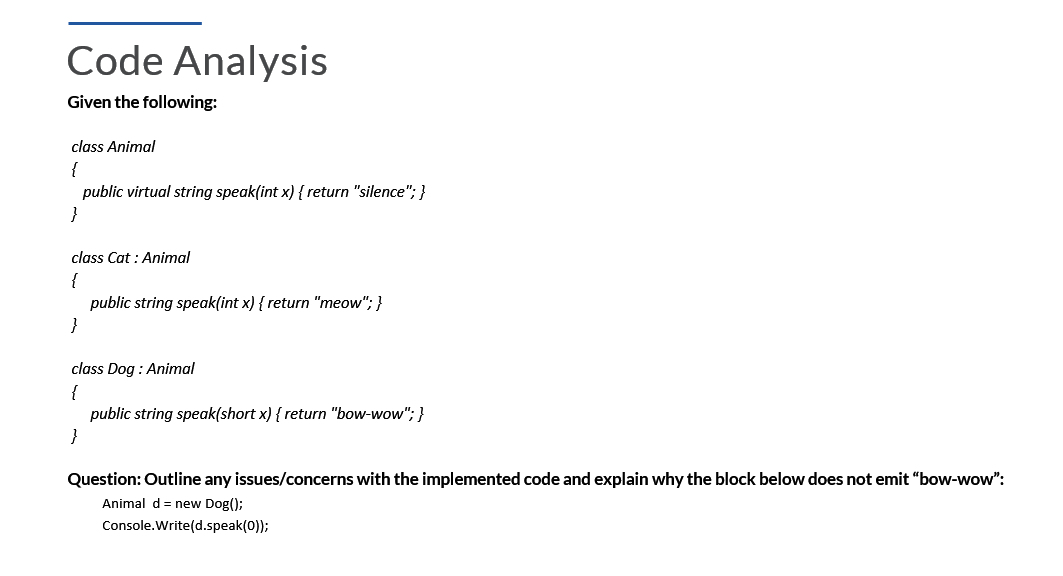
The Problem 1.



Answer:

The reason why the code does not emit "**bow-wow**" comes down to a couple of important details in method overloading and overriding:

1. Mismatch in Method Signature:

The `**Dog**` class has a method `**speak(short x)**`, but the `**Animal**` base class defines `**speak(int x)**`. This means that the method `**speak(int x)**` from `**Animal**` is never overridden in `**Dog**`. The `**Dog**` class instead adds an entirely different method `**speak(short x)**`. So, when you call `**d.speak(0)**`, you're passing `**0**` as an `**int**` by default, which means the method from the `Animal` class (`**speak(int x)**`) is invoked, not the `**speak(short x)**` from `**Dog**`.

2. Why No "bow-wow"?:

Since `**speak(int x)**` from the `**Animal**` class returns "**silence**", that’s exactly what gets printed, because the call to `**d.speak(0)**` is treated as an `**int**` method call from the `**Animal**` class. The `**Dog**`'s `**speak(short x)**` is never even considered, since `**0**` is implicitly an `**int**` in this context, not a `**short**`.

3. How to Fix It?:

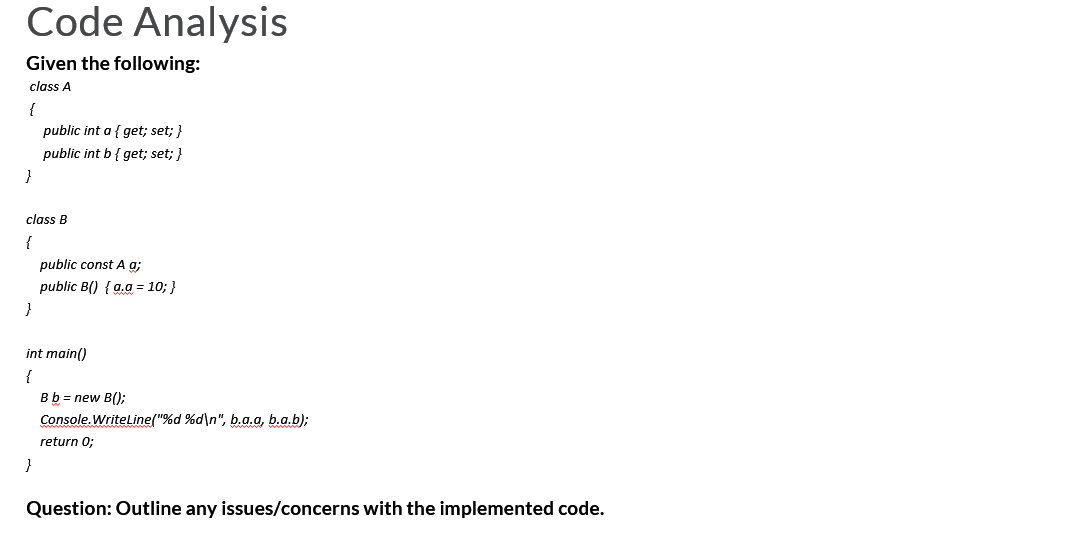
- By overriding the method with the correct signature:

public override string speak(int x) { return "bow-wow"; }

- By invoking the `**short**` version, you can explicitly cast `**0**` to `**short**` like this:

Console.WriteLine(d.speak((short)0));

**The Problem 2.**



The answer:

Let's break this down in a human-like and creative way:

**1. The Code at a Glance:**

- **Class A** has two integer properties: `**a**` and `**b**`.

- **Class B** contains a constant reference to an instance of `**A**` named `**q**`, and its constructor initializes `**q.a**` to `**10**`.

- In `**main()**`, an object of `**B**` is created, and the program tries to print `**b.a.a**` and `**b.a.b**`.

**2. Issues and Concerns:**

1) Constant `**A**` Object in **Class B**:

- The declaration `**public const A q;**` is problematic. `**const**` can only be used for simple data types or immutable objects like `**int**`, `**string**`, or `**double**`. You cannot declare a reference type like `**A**` as `const` because it can be modified. This should probably be `**readonly**` instead of `const`.

- `**readonly**` allows the reference to be initialized in the constructor (like you're doing here), but prevents it from being changed afterward, which seems to be the intended design.

2) Accessing Properties:

- When trying to access `**b.a.a**` and `**b.a.b**`, it assumes that `**B**` has a property or field `**a**`, but there’s no such member defined in class `**B**`. Class `**B**` only has the constant `**q**`. The code should be referring to `**b.q.a**` and `**b.q.b**`.

3) Uninitialized Property:

- The property `**b**` of class `**A**` is never set in your code, meaning that when the program tries to access `**b.a.b**`, it will result in a null reference error. There should be initialization for `**b**`.

**3. How to Fix:**

- Change `**const**` to `**readonly**` in class `**B**`.

- Fix the access of `**a**` and `**b**` to use the `**q**` reference.

- Ensure that both `**a**` and `**b**` properties of class `**A**` are properly initialized before being accessed.

**4. Creative Analogy:**

Imagine you're trying to access two doors in a house. You've labeled the first door as "**A**", but instead of specifying how to open it, you welded it shut by using `**const**`. Also, you're trying to open the second door (`**b**`) that doesn’t even exist yet. To fix this, you need to make sure the door can be unlocked (`**readonly**`) and ensure the house (`**b**`) has all the right doors installed before attempting to open them!