Qual é a saída do programa a seguir? (Escolha todas as opções aplicáveis.)

```
1: interface HasTail { private int getTailLength(); }
2: abstract class Puma implements HasTail {
3: String getTailLength() { return "4"; }
4: }
5: public class Cougar implements HasTail {
6: public static void main(String[] args) {
7: var puma = new Puma() {};
8: System.out.println(puma.getTailLength());
9: }
10: public int getTailLength(int length) { return 2; }
11: }
```

A. 2
B. 4
C. O código não vai compilar por causa da linha 1.
D. O código não vai compilar por causa da linha 3.
E. O código não vai compilar por causa da linha 5.
F. O código não vai compilar por causa da linha 7.
G. O código não vai compilar por causa da linha 10.
H. A saída não pode ser determinada a partir do código fornecido.

ANÁLISE DA QUESTÃO

Vamos entender esse código Java para ver qual é a saída ou onde ele pode dar erro, analisando o código linha por linha:

```
1: interface HasTail { private int getTailLength(); }
```

Aqui temos uma interface chamada `*HasTail*` com um método privado `*getTailLength()*`. **Métodos em interfaces são public e abstract por padrão.** A partir do Java SE9, agora podemos definir métodos privados em uma interface da mesma maneira que os definimos em uma classe. Usamos esses métodos para código interno privado. Eles são acessíveis apenas dentro da interface. Eles não podem ser acessados ou herdados por outra interface ou classe. Entretanto, Segundo o especificação Java (§9.4.3) :

> *It is a compile-time error if an interface method declaration is default, private, or static, and has a semicolon for its body.*

O que significa que o método do tipo *private* tem que ter um corpo, ou seja, devem ser implementados. Logo o a alternativa abaixo está correta.

C. O código não vai compilar por causa da linha 1.

Haveria algum outro erro de compilação caso o método privado da interface fosse implementado?

Veja que na linha '3' eu declaro o mesmo método da interface mas com assinatura diferente na classe abstrata *Puma*, mas isso seria um problema SOMENTE SE o método não fosse privado (*default* ou *public*). Como o método é privado **não irá haver erro de compilação ou execução** 😊.

O único *warning* seria *'The method getTailLength() from the type HasTail is never used locally'*, uma vez que o método foi declarado mas jamais utilizado no código.

Da mesma forma o método implementado na classe *Cougar* não entra em conflito com o método privado da interface.

Perceba outra sutileza nesta questão, veja o Código abaixo:

```
7: var puma = new Puma() {};
```

No código acima, a variável puma é inicializada como **uma instância de uma classe anônima que estende a classe Puma**. Isso é indicado pela sintaxe ‘{}’ após *new* Puma(), o que cria uma nova classe anônima que é uma subclasse de Puma. **Classes anônimas** são usadas quando você deseja **implementar uma classe rapidamente sem precisar criar uma nova classe com um nome específico**. Neste caso, a classe anônima criada não tem um nome específico (daí "anônima"), mas ainda é uma subclasse de Puma, herdando suas características, incluindo métodos e comportamentos.

Assim, caso não houvesse o erro de compilação na linha ‘1’, o resultado da execução deste código seria ‘4’ impresso no console, que é o retorno do método *getTailLength()* implementado na classe Puma