**✅ GABARITO COM JUSTIFICATIVAS**

**Tema**: Diagramas de Casos de Uso (UML) e Encapsulamento (POO - Python)

**1. (c) Descrever as interações entre os usuários e o sistema.**

**Justificativa:**  
Diagramas de casos de uso são usados para representar, de forma visual, como os **atores (usuários ou sistemas externos)** interagem com as funcionalidades do sistema. As demais alternativas confundem com documentação técnica ou geração de código, que não são o foco da UML.

**2. (d) Uma entidade que interage com o sistema.**

**Justificativa:**  
O “ator” é qualquer entidade externa (humana ou sistema) que **interage com o sistema**, iniciando ou participando de casos de uso. Não é um componente de hardware, nem uma função executada pelo sistema.

**3. (c) O relacionamento “extend” adiciona comportamentos sob certas condições.**

**Justificativa:**  
Relacionamento extend é usado quando há **comportamentos opcionais**, ou seja, adiciona funcionalidades adicionais quando certas condições são satisfeitas. Já o include representa funcionalidade obrigatória. Casos de uso podem sim se relacionar.

**4. (b) Proteger o acesso direto aos atributos da classe.**

**Justificativa:**  
O encapsulamento **protege os dados** ao evitar que atributos internos sejam acessados ou alterados diretamente. Em vez disso, usa-se métodos (getters e setters) para esse controle.

**5. (c) Setters servem para modificar atributos privados com controle.**

**Justificativa:**  
Setters permitem modificar atributos privados de forma **segura e validada**. Getters recuperam os dados. Ambos mantêm o controle do acesso e garantem o princípio do encapsulamento.

**6. (c) Encapsulamento**

**Justificativa:**  
Encapsulamento refere-se ao ato de **esconder os detalhes internos** de funcionamento de uma classe, expondo apenas o necessário para o uso externo. Isso difere de herança ou polimorfismo.

**7. (c) Um sublinhado antes do nome do atributo (ex: \_nome).**

**Justificativa:**  
Em Python, a **convenção para indicar atributos privados** é o uso de um único sublinhado antes do nome. Isso não torna o atributo inacessível, mas sinaliza que ele é interno à classe.

**8. (c) Exibir como um cliente pode fazer login em um sistema bancário.**

**Justificativa:**  
Isso é um exemplo clássico de uso de diagrama de casos de uso: **representar a interação entre um ator (cliente)** e uma funcionalidade (fazer login). As outras alternativas fogem da finalidade da modelagem.

**9. (d) Retornar o valor de um atributo de forma segura.**

**Justificativa:**  
O getter permite que um atributo seja **lido** sem ser exposto diretamente. Ele faz parte do controle de acesso aos dados no encapsulamento. Ele **não altera**, apenas **lê com segurança**.

**10. (b) Permite esconder implementações complexas e proteger os dados.**

**Justificativa:**  
Encapsulamento promove **segurança, manutenção e abstração**, escondendo implementações e evitando acessos indevidos aos dados. Isso **não substitui herança**, nem facilita acesso irrestrito.