Um dos grandes diferenciais da programação orientada a objetos em relação a outros paradigmas de programação, que também permitem a definição de estruturas e de operações sobre essas estruturas, está no conceito de herança, mecanismo através do qual definições existentes podem ser facilmente estendidas. Juntamente com a herança, deve ser enfatizada a importância do polimorfismo, que permite selecionar funcionalidades que um programa irá utilizar de forma dinâmica, durante sua execução. Objetos são instâncias de classes que determinam qual informação um objeto contém e como ele pode manipulá-la.

RICARTE, Ivan Luiz Marques. **Fundamentos da programação orientada a objetos**. Programação Orientada a Objetos: uma abordagem com Java. Universidade Estadual de Campinas, Campinas, SP, Brasil, 2001 (adaptado).

Face ao exposto, analise as seguintes asserções e a relação proposta entre elas.

I. Em um projeto real, deve-se utilizar o encapsulamento para definir critérios de segurança a uma classe.

PORQUE

II. Sem o encapsulamento, os atributos das classes poderiam ser modificados diretamente por qualquer método, ainda que externo.

A respeito das asserções, assinale a alternativa correta.

Alternativas

A)
As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.

B)
As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.

C)
A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

D)
A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

E)
As asserções I e II são proposições falsas.

O paradigma da Programação Orientada a Objetos (POO) está presente na maioria das linguagens de programação atuais, principalmente aquelas consideradas de alto nível (que não têm interação direta com o hardware do dispositivo). Esse paradigma tem duas principais características que se destacam e o tornam popular entre os desenvolvedores: a segurança e a reutilização de código. A segurança é alcançada por meio do encapsulamento, enquanto a reutilização do código ocorre devido a alguns fatores, dentre eles, a possibilidade de herança de métodos e atributos de uma classe para outra.

A herança fornece às classes-filhas (chamadas também de subclasses) todo o escopo pertencente à classe-pai (superclasse), exceto aqueles protegidos pelas regras de encapsulamento. Linguagens como o C++ e Python dão ao desenvolvedor a possibilidade de realizar heranças múltiplas nas classes, ou seja, uma subclasse pode herdar métodos e atributos de duas ou mais classes ao mesmo tempo, enquanto a linguagem Java é mais restritiva nesse aspecto.

Considerando as definições expostas, é recomendado que todo desenvolvedor de sistemas domine ao menos a teoria que envolve a Programação Orientada a Objetos e suas especificidades quanto à herança. Ainda que existam mudanças sutis de linguagem para linguagem, o conhecimento amplo do assunto permite ao desenvolvedor escolher caminhos paralelos para a resolução de problemas.

Na linguagem Java, por exemplo, caso fosse necessário que uma CLASSE C recebesse herança da CLASSE A e da CLASSE B ao mesmo tempo, o desenvolvedor deverá

Alternativas

A)
remover o encapsulamento (segurança) que impede a herança múltipla entre as várias classes do programa.

B)
realizar uma herança da Classe A para a Classe B e, em seguida, uma outra herança da Classe B para a Classe C.

C)
utilizar a sobreposição, para que a Classe A possa interagir diretamente com a Classe C, tornando-se sua classe-pai.

D)
habilitar o encapsulamento na Classe C, para que seu escopo fique acessível para receber herança de outras classes.

E)
realizar a herança da Classe A para a Classe C e criar um objeto na Classe C, que acesse as propriedades da Classe B.