QUESTÃO 1 - CIRCLE X ELLIPSE

Dado que a classe **ELLIPSE** é pai da classe **CIRCLE** (*faz sentido, porque círculos são elipses*), a classe **CIRCLE** pode reusar todo o conteúdo da classe **ELLIPSE**, bastando para isso apenas sobrescrever os métodos, visando garantir que os eixos maior e menor permaneçam iguais. No entanto, o método

> *void Ellipse.stretchMaior() // "estica" a elipse na direcao do eixo maior*

não funciona com **CIRCLE**, pois o resultado deixa de ser um círculo.

Não é possível fazer **CIRCLE** pai de **ELLIPSE**, pois seria conceitualmente errado, já que nem toda **ELLIPSE** é um **CIRCLE**. Analise, o que aconteceria se uma função espera um círculo e recebe uma elipse?

Além disso, o método *double Circle.getRadius()* não faz sentido com uma elipse.

a) Explique este dilema conceitualmente, usando para isso apenas os conceitos e vocabulários constantes de POO, especialmente àqueles relacionados a responsabilidade e herança. **(1.0 PT)**

R: Na primeira suposição criada pela questão, a classe CIRCLE é filha de classe ELLIPSE, ou seja, ela herda seus métodos. Chama-se isso de herança por especificação. Dessa forma, o método getRadius() poderia ser implementado normalmente em CIRCLE dado que será um método específico dessa classe, porém o método *stretchMaior()* advindo da classe ELLIPSE teria que ser sobrescrito na classe CIRCLE para que a utilização dela responda uma exceção – no entanto, essa solução não é uma boa prática de programação. A segunda suposição da questão, toca também no conceito da relação é-um e que se mudada será violada, pois a herança pressupõe uma relação de generalização entre o pai sobre os filhos ou de especificação entre os filhos do pai. Isto é, CIRCLE é um ELLIPSE com eixos iguais simplesmente.

b) Forneça uma solução que ainda promova o reuso de código. A sua solução pode ter uma desvantagem, no ponto de vista do programador que usa as suas classes. Explique-a conceitualmente a solução e a desvantagem, usando o vocabulário de POO, do ponto de vista do programador que usa as suas classes. **(1.0 PT)**

R: Primeiramente, pode-se manter a relação de herança por especificação entre a classe ELLIPSE e a classe CIRCLE de tal forma que: public class CIRCLE extend ELLIPSE. Dessa forma, mantendo os métodos que seria possível herdar e os métodos que seriam cabíveis de uma sobrescrita válida – sem a necessidade de utilizar exceções. Como o método getRadius() seria específico de CIRCLE poderia ser implementado nele. Sobre o método strechMaior() poderia ou ser eliminado de ELLIPSE ou retornar uma exceção em CIRCLE. Uma solução prática, porém, chula.

Por outro lado, poderíamos utilizar a técnica do polimorfismo, ou seja, criar interfaces que possuam conjunto de comportamentos cabíveis para CIRCLE e ELLIPSE. Por exemplo,

public class CIRCLE implements PropriedadesComunsDeCirculoElipse, Radius

public class ELLIPSE implements PropriedadesComunsDeCirculoElipse, Strech

Em que, poderíamos separa de comportamento de CIRCLE e ELLIPSE em diversos blocos de interface sem prejudicar individualmente suas implementações.