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)**

RESPOSTA: A classe Ellipse é pai de Circle, dessa forma, nos conceitos de POO, todo o Circle também é uma elipse, e além de herdar as responsabilidades de Ellipse, ele também terá suas responsabilidades específicas no qual elipse não tem.

Assim, seria natural que o método Circle tenha um método getRadius no qual Ellipse não tem. O problema acontece porque o Circle limita o comportamento de Ellipse, no caso do método de stretchMaior, o que quebraria o Princípio de substituição de Liskov, uma vez que o filho está restringindo uma responsabilidade que existia na classe pai.

Analisando o que aconteceria se uma função que espera um círculo porém recebe uma elipse: a menos que se fizesse um casting perigoso de ellipse para circle, haveria um exceção na execução do trecho de código que utiliza uma ellipse onde se espera um circle, uma vez que uma ellipse não é um circle. E no caso do casting perigoso, o código funcionaria caso fossem acessados apenas métodos e atributos que de fato existam na Ellipse.

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)**

RESPOSTA: Uma solução para esse dilema é sobrescrever o método strechMaior no Circle, de tal forma que o strech acontece nos dois eixos, fazendo com que o resultado continue sendo um Circle.

A sua desvantagem é que esse comportamento pode ser inesperado para um programador que esteja utilizando a classe, pois isso quebra o Princípio de Substituição de Liskov, uma vez que a classe filha restringe o comportamento da classe pai. Nesse caso, strechMaior no circulo seria o mesmo que um strechAll, pois ele não aumentaria apenas o eixo maior, mas aumentaria ambos os eixos.

O método getRadius está presente apenas no círculo, que é um novo método dele.

```
Public Circle extends Ellipse {
        public void strechMaior() {
                // Aumenta ambos os eixos, fazendo com que o resultado continue sendo circulo
        }
        public double getRadius() {
                return radius;
        }
}
```