Computação Orientada a Objetos - P1 - 1°sem/2019 - T04

1. Você está participando do desenvolvimento de um jogo que, em resumo, simula um ambiente bidimensional no qual inimigos se movimentam e podem ser atingidos pelo jogador (por hora, eles não possuem comportamento de ataque, recurso que será implementado em uma futura expansão). No código abaixo encontra-se a declaração de uma primeira versão da classe que modela os inimigos deste jogo. Apesar de compilar, tal código apresenta problemas no que diz respeito a uma boa modelagem orientada a objetos. Identifique os problemas presentes no código e proponha as alterações necessárias para corrigi-los [2.0 pontos].

```
class Inimigo {
    private static final int VELOCIDADE_X = 10; // velocidade horizontal dos inimigos
    private static final int VELOCIDADE_Y = 10; // velocidade vertical dos inimigos
    private int x; // coordenada horizontal
    private int y; // coordenada vertical
    private int hp = 10; // pontos de vida
    private boolean ativo = true; // indica se o inimigo esta ativo ou nao
    private int totalAtivos = 0; // contador do total de inimigos ativos do jogo

    public Inimigo(int x_ini, int y_ini) {
        setX(x_ini); setY(y_ini); setTotalAtivos(getTotalAtivos() + 1);
    }

    // getters
    public int getX() { return x; }
    public int getY() { return y; }
    public int getHp() { return hp; }
    public boolean getAtivo() { return ativo; }
    public int getTotalAtivos() { return totalAtivos; }

    // setters
    public void setX(int i) { x = i; }
    public void setY(int i) { y = i; }
    public void setHp(int i) { hp = i; }
    public void setAtivo(boolean b) { ativo = b; }
    public void setTotalAtivos(int i) { totalAtivos = i; }

    // metodo chamado para realizar a movimentacao do inimigo. dirX e dirY determinam o
    // sentido do movimento na horizontal e na vertical, respectivamente (apenas o sinal
    // destes parametros sao relevantes).
    public void move(int dirX, int dirY) {
        setX(getX() + Integer.signum(dirX) * VELOCIDADE_X);
        setY(getY() + Integer.signum(dirY) * VELOCIDADE_Y);
    }

    // metodo chamado sempre que o jogador acerta um inimigo.
    public void recebeDano() {
        if(getAtivo()) {
            setHp(getHp() - 1);
            if(getHp() == 0) { setAtivo(false); setTotalAtivos(getTotalAtivos() - 1); }
        }
    }
}
```

2. Explique como **composição** pode ser utilizada como uma alternativa à **herança** [2.0 pontos].

3. Reescreva o código abaixo de modo que o mesmo tire proveito do mecanismo para tratamento de exceções da linguagem Java e continue funcionando exatamente da mesma forma (inclusive informando as mesmas mensagens de erro) [3.0 pontos].

```
class Q3 {

    public static final int INDEFINIDO = -1;
    public static final int FALHA = 0;
    public static final int SUCESSO = 1;
    public static int status_flag = INDEFINIDO;

    public static double realizaCalculo(double x, double y){
        if(x >= y) {status_flag = FALHA; return 0; }
        else { status_flag = SUCESSO; return x / (y - x); }
    }

    public static void main(String[] args){
        double a, b, c, d, r1, r2, r3, r_final = 0;
        boolean tudo_OK = false;

        while(!tudo_OK){

            // Faz a leitura dos valores a, b, c e d.

            r1 = realizaCalculo(a, b);
            if (status_flag != SUCESSO){
                System.out.println("Falha em realizaCalculo para os valores: " + a + ", " + b);
                continue;
            }
            r2 = realizaCalculo(b, c);
            if (status_flag != SUCESSO){
                System.out.println("Falha em realizaCalculo para os valores: " + b + ", " + c);
                continue;
            }
            r3 = realizaCalculo(c, d);
            if (status_flag != SUCESSO){
                System.out.println("Falha em realizaCalculo para os valores: " + c + ", " + d);
                continue;
            }
            r_final = r1 + r2 + r3;
            tudo_OK = true;
        }
        System.out.println("Resultado final: " + r_final);
    }
}
```

4. Escreva um método genérico **eficiente** que recebe um vetor (*array*) de elementos do tipo genérico T (no qual espera-se que elementos considerados iguais apareçam repetidas vezes) e determine o número total de ocorrências de cada elemento distinto. Para cada elemento distinto seu método deve imprimir uma linha, na saída padrão, com o formato "ELEMENTO: NUMERO_DE_OCORRÊNCIAS" [3.0 pontos].