Frequência intermédia

14 Novembro - 9h30

Salas publicadas no Moodle

Importante: Quem chegar atrasado não poderá fazer o teste

Matéria que sai:
aula teórica 1 até à aula teórica 7

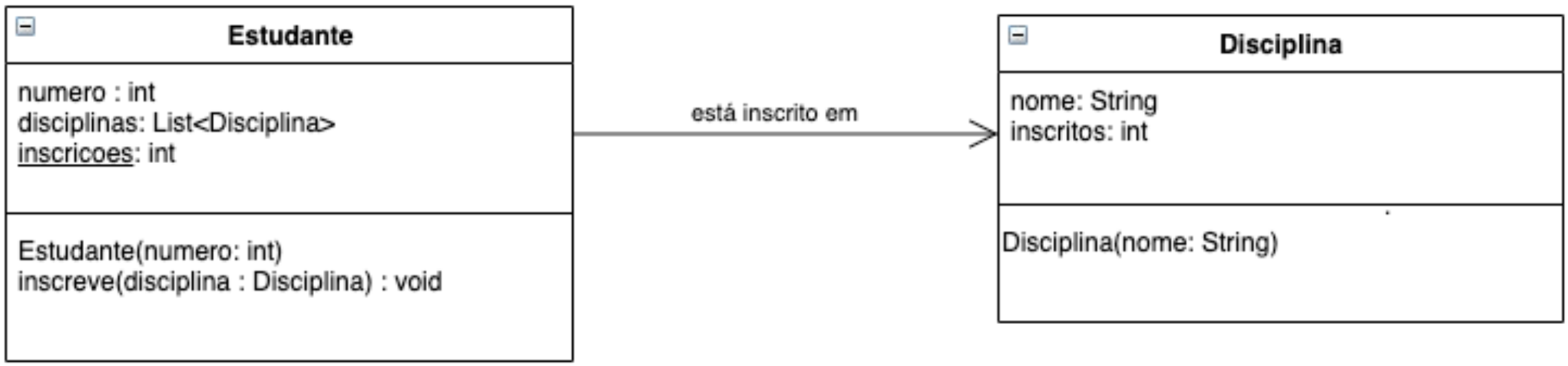
```java
class Disciplina {
    String nome;
    int inscritos;

    public Disciplina(String nome) {
        this.nome = nome;
        this.inscritos = 0;
    }
}

class Estudante {
    int numero;
    List<Disciplina> disciplinas = new ArrayList<>();
    static int inscricoes = 0;

    public Estudante(int numero) {
        this.numero = numero;
    }

    public void inscreve(Disciplina disciplina) {
        if (inscricoes < 3) {
            disciplina.inscritos++;
            disciplinas.add(disciplina);
            inscricoes++;
        }
    }
}

(…)
Disciplina d1 = new Disciplina("FP");
Disciplina d2 = new Disciplina("LP2");

Estudante e1 = new Estudante(21500000);
e1.inscreve(d2);

Estudante e2 = new Estudante(21501111);
e2.inscreve(d1);
e2.inscreve(d2);

Estudante.inscricoes += d2.inscritos;
e1.inscreve(d1);
```

Exercício

Desenhe o diagrama UML desta aplicação

Enviar por teams para p4997

Resolução



```java
class Disciplina {
    String nome;
    int inscritos;

    public Disciplina(String nome) {
        this.nome = nome;
        this.inscritos = 0;
    }
}

class Estudante {
    int numero;
    List<Disciplina> disciplinas = new ArrayList<>();
    static int inscricoes = 0;

    public Estudante(int numero) {
        this.numero = numero;
    }

    public void inscreve(Disciplina disciplina) {
        if (inscricoes < 3) {
            disciplina.inscritos++;
            disciplinas.add(disciplina);
            inscricoes++;
        }
    }
}

(…)
Disciplina d1 = new Disciplina("FP");
Disciplina d2 = new Disciplina("LP2");

Estudante e1 = new Estudante(21500000);
e1.inscreve(d2);

Estudante e2 = new Estudante(21501111);
e2.inscreve(d1);
e2.inscreve(d2);

Estudante.inscricoes += d2.inscritos;
e1.inscreve(d1);
```

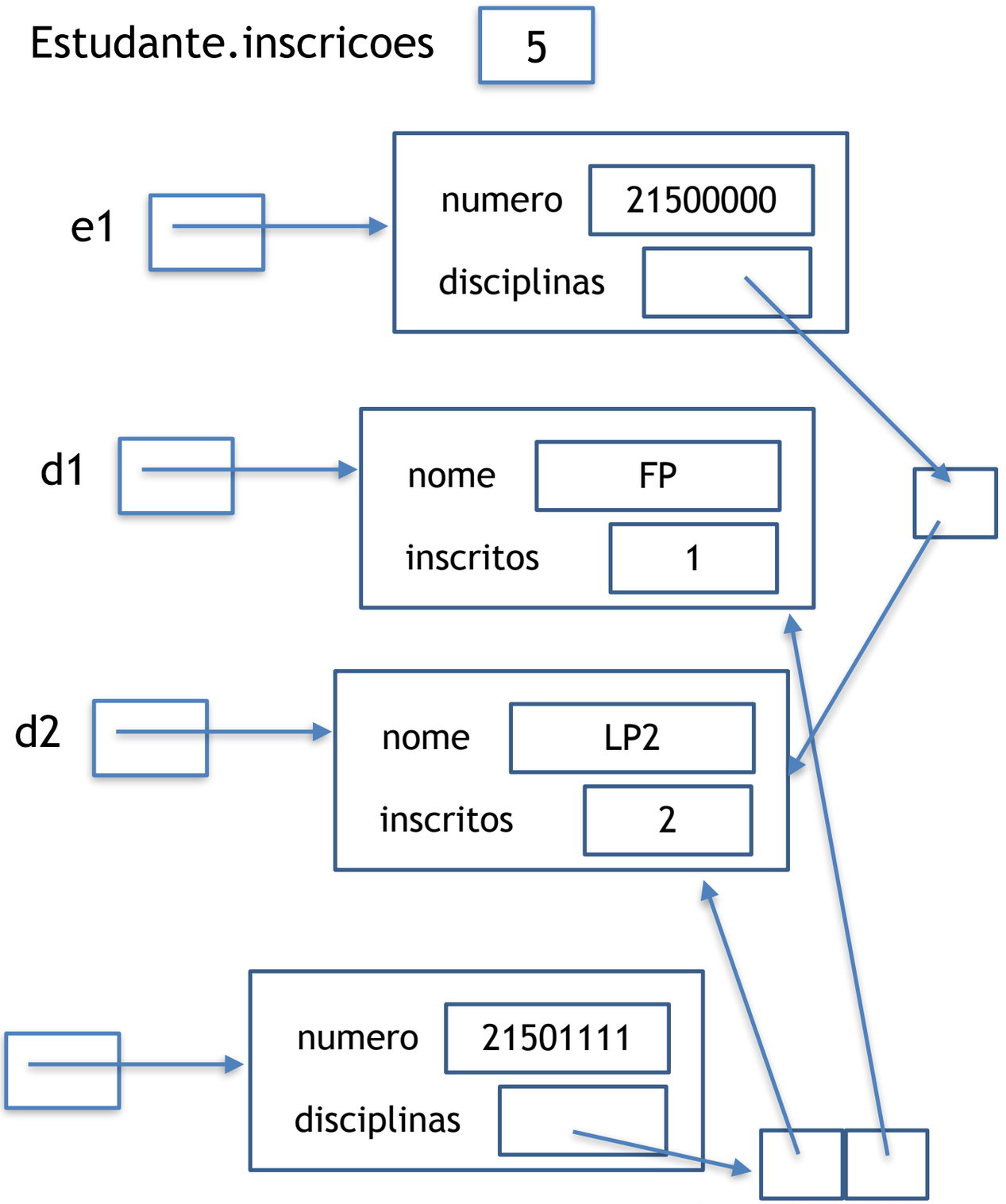
Exercício

Desenhe a estrutura de memória, no final da execução deste programa

Enviar por teams para p4997

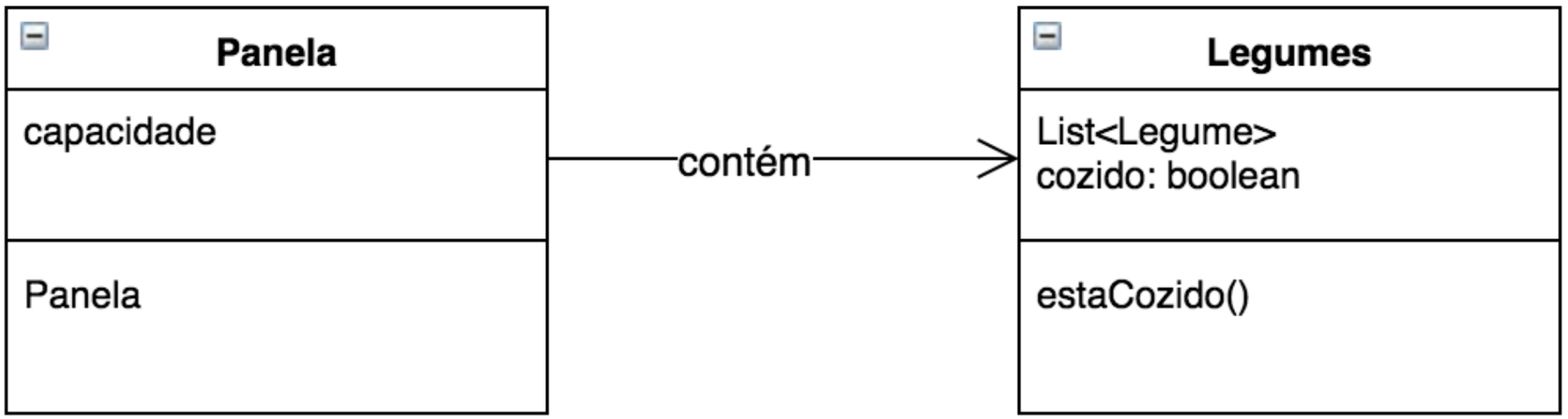
```java
class Disciplina {
    String nome;
    int inscritos;

    public Disciplina(String nome) {
        this.nome = nome;
        this.inscritos = 0;
    }
}

class Estudante {
    int numero;
    List<Disciplina> disciplinas = new ArrayList<>();
    static int inscricoes = 0;

    public Estudante(int numero) {
        this.numero = numero;
    }

    public void inscreve(Disciplina disciplina) {
        if (inscricoes < 3) {
            disciplina.inscritos++;
            disciplinas.add(disciplina);
            inscricoes++;
        }
    }
}

(…)
Disciplina d1 = new Disciplina("FP");
Disciplina d2 = new Disciplina("LP2");

Estudante e1 = new Estudante(21500000);
e1.inscreve(d2);

Estudante e2 = new Estudante(21501111);
e2.inscreve(d2);
e2.inscreve(d1);

Estudante.inscricoes += d2.inscritos;
e1.inscreve(d1);
```

Resolução



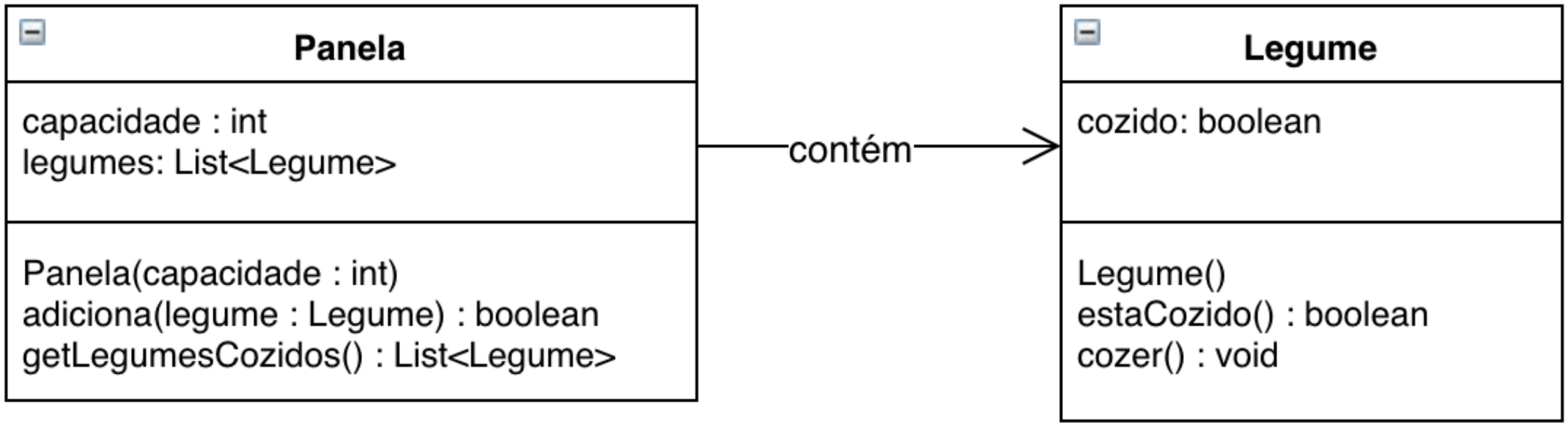
Exercício (breakout rooms)

Descobrir os erros neste diagrama

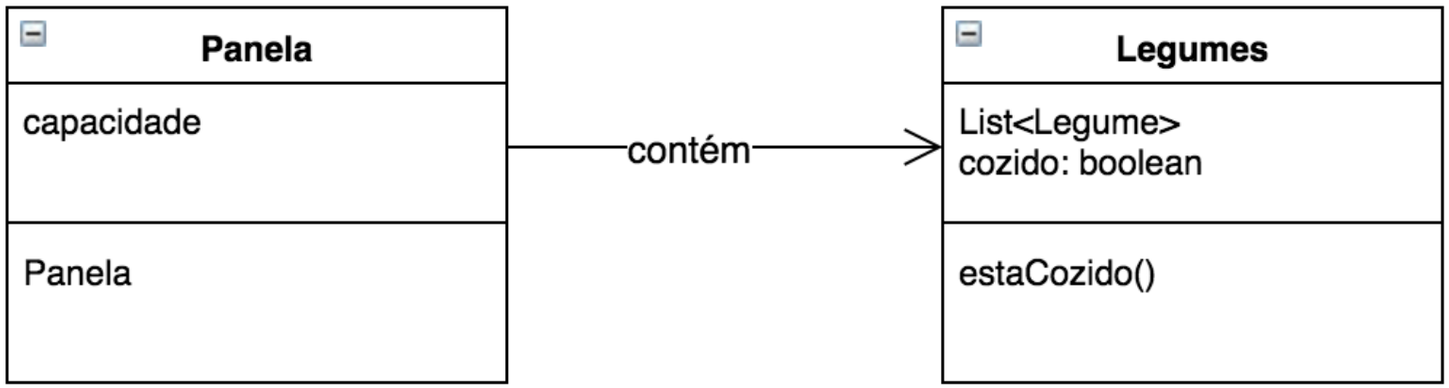


Vai ser chamado um aluno de cada grupo para responder e justificar

Exercício
Possível Solução

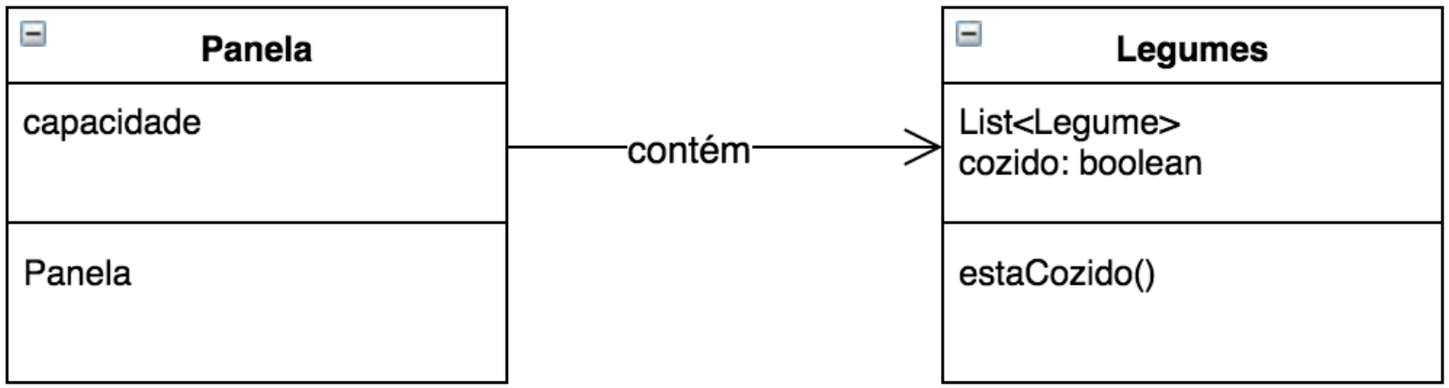


Exercício
Descobrir os erros neste diagrama



1. A seta “contém” não se reflete em nenhuma variável da Panela
2. A variável **capacidade** não tem tipo
3. O construtor **Panela** não abre/fecha parêntesis
4. O nome da classe **Legumes** devia estar no singular
5. Não faz sentido um Legume ter uma lista de legumes (List<Legume>)
6. O método estaCozido devia retornar um boolean
7. Deveria haver uma acção para meter legumes na panela
8. Todos os atributos devem ter métodos que leiam e inicializem/alterem o seu valor

Exercício
Descobrir os erros neste diagrama



1. A seta “contém” não se reflete em nenhuma variável da Panela
2. A variável **capacidade** não tem tipo
3. O construtor **Panela** não abre/fecha parêntesis
4. O nome da classe **Legumes** devia estar no singular
5. Não faz sentido um Legume ter uma lista de legumes (List<Legume>)
6. O método estaCozido devia retornar um boolean
7. Deveria haver uma acção para meter legumes na panela
8. Todos os atributos devem ter métodos que leiam e inicializem/alterem o seu valor

```java
public class CalculadoraEstranha {

    private long valorActual;
    private int numOperacoes;

    public CalculadoraEstranha(long valorInicial) {
        this.valorActual = valorInicial;
        this.numOperacoes = 0;
    }

    public void soma(int numero) {
        if (valorActual >= 0) {
            valorActual += numero;
        } else {
            valorActual -= numero;
        }
        numOperacoes++;
    }

    public void soma(int[] numeros) {
        if (numeros == null || numeros.length == 0) {
            numOperacoes++;
        }

        for (int numero: numeros) {
            valorActual += numero;
            numOperacoes++;
        }
    }

    public long getValorActual() {
        return valorActual;
    }

    public int getNumOperacoes() {
        return numOperacoes;
    }
}
```

Testes unitários

Preencha a tabela de testes com o número mínimo de testes necessário para testar todos os cenários possíveis (evite testes redundantes)

estado inicial	acção	estado final esperado

Resolução

estado inicial	acção	estado final esperado
valorActual=3 numOperacoes=0	soma(4)	valorActual=7 numOperacoes=1
valorActual= -3 numOperacoes=0	soma(4)	valorActual= -7 numOperacoes=1
valorActual=3 numOperacoes=0	soma([2,3])	valorActual=8 numOperacoes=2
valorActual=3 numOperacoes=0	soma([])	valorActual=3 numOperacoes=1
valorActual=3 numOperacoes=0	soma(null)	valorActual=3 numOperacoes=1

Resolução Quizzes

Os paradigmas de programação representam formas diferentes de programar. Um deles envolve a criação de entidades com estado e comportamento, chamado programação orientada a objectos. A linguagem Ruby está associada a esse paradigma.

Existe outro paradigma que olha para os programas como representações de fluxogramas, chamado programação imperativa. É adequado para problemas algorítmicos simples e a linguagem C é um exemplo de linguagem associada a este paradigma.

Finalmente existe um paradigma em que o elemento principal é a função. Este paradigma tem vindo a ganhar adesão nos últimos anos pois facilita a programação paralela. Um exemplo de linguagem desse paradigma é o Javascript.

Resolução Quizzes

Na fase de análise orientada a objetos, pega-se no texto com os requisitos e começamos por identificar as entidades da aplicação. De seguida, para cada uma delas, identificamos os atributos (nomes e adjetivos) e as ações (verbos). Uma das regras fundamentais é que todas as ações têm que ler ou modificar os tais atributos.

De seguida, passamos para a fase da programação . Nesta fase, cada entidade é transformada numa classe , cada atributo é transformado numa variável e cada ação é transformada num método . Uma vez definidas as classes, vamos criar instâncias (feminino, plural) de cada classe com a palavra reservada `new`, as quais vão estar possivelmente associadas a variáveis. Quando uma variável está associada ao resultado do `new`, dizemos que representa um objeto .

O princípio do encapsulamento é uma boa prática de programação orientada a objetos e diz-nos que uma classe não deve manipular diretamente as variáveis de outra classe. Em vez disso, deve usar os métodos da classe, que por sua vez vão alterar o seu estado interno.

Resolução Quizzes

Analise o seguinte programa:

```
1  class A {
2     static int x;
3     int y = 3;
4     final int z = 3;
5
6     static void f1(int y) {
7         x += y;
8         System.out.println(this.y);
9         System.out.println(z);
10    }
11
12    int f2(int z) {
13        x += z;
14        System.out.println(y);
15        System.out.println(z);
16        return x;
17    }
18 }
19
20 public class StaticExercise {
21
22    public static void main(String[] args) {
23
24        int k = 2;
25
26        A.f1(3);
27        A.f2(4);
28        new A().f2(k);
29    }
30 }
```

Identifique as linhas que dão erro de compilação, escrevendo o respetivo número nas caixas abaixo mas apenas um número por caixa. Deve escrever os números por ordem crescente. Se as caixas forem mais do que o número de erros que encontrou, deve responder 99 nas caixas que sobrarem.