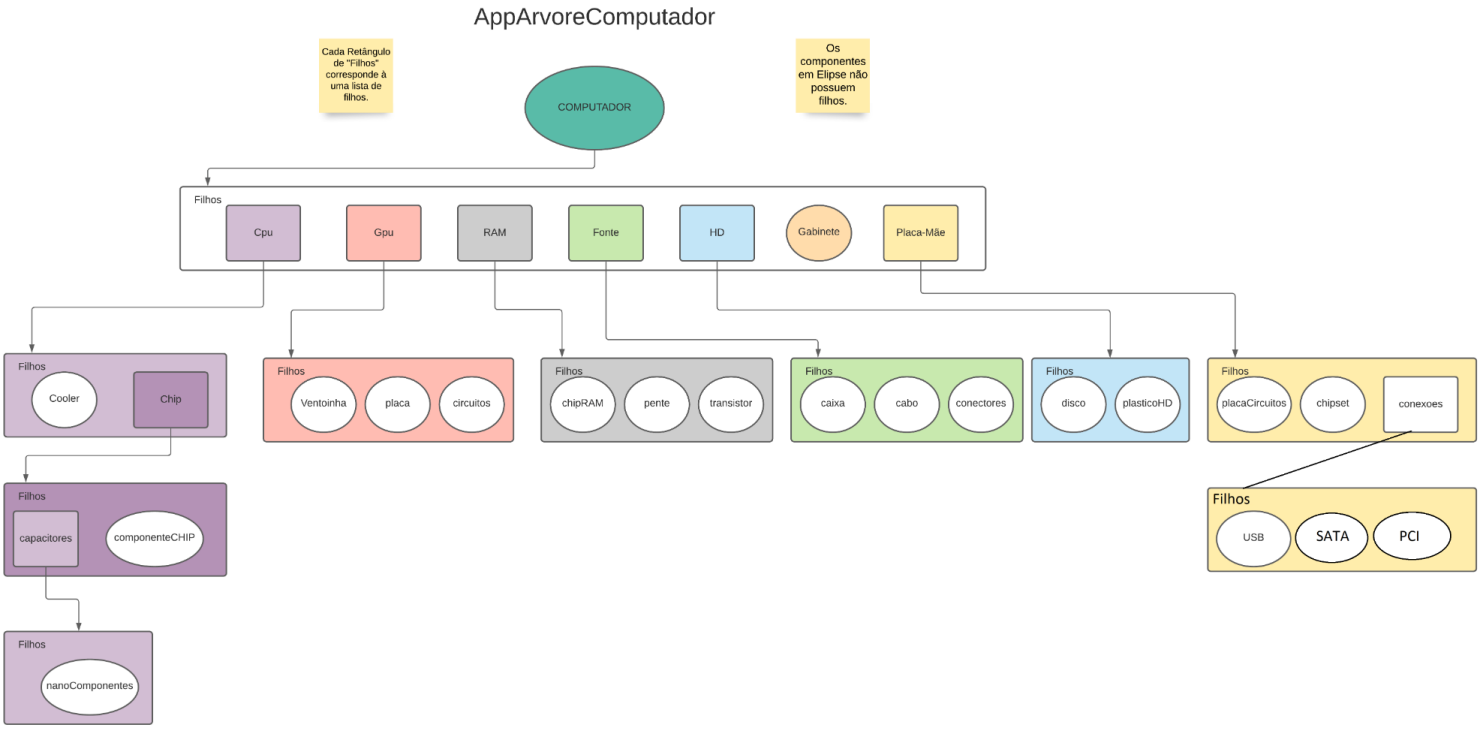
Trabalho 2 de Estrutura de Dados

Michel Ribeiro Corrêa

Exemplo AppArvoreComputador



Árvore - Raiz - Produto: Computador.

Lista de Filhos;

Classe Arvore

```java
public class Arvore {
    private No raiz;

    public Arvore(String n) {
        this.raiz = new No(n);
    }

    public Arvore() {

    }

    public No getRaiz() {
        return raiz;
    }

    public No buscar(String procurado) {
        No achado = raiz.buscar(procurado);
        return achado;
    }

}
```

A classe Arvore possui somente um atributo, o Nó Raiz.
Um construtor para a criação da árvore e atribuir sua raiz com um Nome, que no exemplo aplicado, será “Computador”.

2 Métodos, um getRaiz e um método para buscar um nó/subproduto na Árvore usando um nome como comparação de igualdade.

Classe No

```java
public class No {
    private String nome;
    private float valor;
    private int quantidade;
    private List<No> filhos = new ArrayList<>();
    public static float soma;
    public static List<No> folhas = new ArrayList<>();
    public static List<No> componentes = new ArrayList<>();
```

A classe No possui 4 atributos: nome, valor, quantidade e uma Lista de filhos, que pode ou não existir.

Os outros 3 atributos são estáticos, o atributo soma é a soma total do produto.
O atributo folhas, é uma lista com todos os produtos que não possuem filhos, os componentes da árvore.
O último atributo componente é uma outra lista, com todos os nós cadastrados na Árvore, independente da existência dos filhos do nó.

Métodos importantes da classe Nó

A classe Nó tem diversos construtores, caso um subproduto não tenha sub peças seu construtor pode ser diferente de um construtor pai.

Atributos get and set dos atributos não estáticos.

```
public void acrescentarFilho(No filho) {
    filhos.add(filho);
}
```

Esse método adiciona um Nó/Subproduto numa lista de outro Nó, ou seja, um filho/componente de uma peça.

```
public No buscar(String procurado) {
    if (procurado.equals(this.nome))
        return this;
    for (No filho : filhos) {
        No achou = filho.buscar(procurado);
        if (achou != null)
            return achou;
    }
    return null;
}
```

O método buscar retorna um nó com o mesmo nome do que foi passado como parâmetro de busca, por meio da recursividade esse método percorre toda árvore procurando um produto com o nome do procurado. O método retorna null quando não encontra.

```java
public void valorArvore() {
    if(filhos.size() == 0){
        soma = soma + (quantidade * valor);
    }else {
        for(No filho : filhos){
            if(quantidade > 1 && quantidade != 0){
                filho.valorArvore(quantidade);
            }else {
                filho.valorArvore();
            }
        }
    }
}

public void valorArvore(int param) {
    if(filhos.size() == 0){
        soma = soma + (quantidade * valor * param);
    }else {
        for(No filho : filhos){
            if(quantidade > 1 && quantidade != 0){
                filho.valorArvore(quantidade);
            }else {
                filho.valorArvore();
            }
        }
    }
}
```

Os métodos valorArvore() somam todos os componentes da árvore.
No caso do método sem parâmetro o nó pai não tem uma quantidade maior que 1, assim não precisa fazer a multiplicação no valor.
O método com parâmetro realiza a multiplicação com a quantidade estabelecida, que vem da quantidade do elemento Pai.
Ambos os métodos usam a recursividade para percorrer a árvore e realizar a soma de todos os Nós da árvore que não possuem filhos, fazendo uma soma de todo o produto.
No exemplo usado, o Computador possui diversos componentes, e o Gabinete é um elemento filho do Computador, mas não tem filhos, então é somado no valor. Já os outros filhos da raiz, possuem outra lista de filhos, logo, seu valor é definido pela soma de seus filhos.

```java
public void folhas() {
    if (filhos.size() == 0) {
        if (this != null) {
            folhas.add(this);
        }
    } else {
        for (No filho : filhos) {
            filho.folhas();
        }
    }
}

public void componentes(){
    for(No filho : filhos){
        componentes.add(filho);
        if(filho.filhos.size() != 0){
            filho.componentes();
        }
    }
}
```

Esses 2 métodos fazem a adição nas listas de folhas e componentes.

Em ambos os casos, usam recursividade para percorrer a árvore porém no método folhas(), somente os componentes sem filhos são adicionados, são os componentes que representam o valor do Produto/Árvore.

No método componentes(), todos itens cadastrados na Árvore, menos a raiz, não adicionados à lista.

Funcionalidade - Main

```
Arvore arvore = new Arvore("Computador");


//CADASTRO DE SUB-PEÇAS
No cpu = new No ("CPU", 1000f, 1);
No gpu = new No ("GPU", 2000f, 1);
No ram = new No ("RAM", 200f, 4);
No fonte = new No ("Fonte", 400f, 1);
No hd = new No ("HD", 200f, 2);
No gabinete = new No("Gabinete", 400f, 1);
No placamae = new No("Placa-Mãe", 500f, 1);


arvore.getRaiz().acrescentarFilho(cpu);
arvore.getRaiz().acrescentarFilho(gpu);
arvore.getRaiz().acrescentarFilho(ram);
arvore.getRaiz().acrescentarFilho(hd);
arvore.getRaiz().acrescentarFilho(fonte);
arvore.getRaiz().acrescentarFilho(gabinete);
arvore.getRaiz().acrescentarFilho(placamae);
```

Inserção do nome do Produto / Raiz = “Computador”
Criação de vários Nós/Sub peças, com nome, valor e quantidade, não sendo necessário o valor, mas coloquei pro cadastro.

Inserção desses componentes na árvore, como filhos da raiz.

Ao longo do código foram inseridos diversos filhos para esses componentes, como desenhado no diagrama.

```
buscado = arvore.getRaiz().buscar("FONTE");
```

Realiza a busca do componente com o nome “FONTE” e retorna no Nó buscado. Nesse exemplo, o elemento FONTE não existia na árvore.

```
arvore.getRaiz().valorArvore();
System.out.println("\n \n \n :::::: VALOR TOTAL DO PRODUTO: "+No.soma);
```

Realiza a soma de toda árvore, e imprime o custo de todo o produto, Computador.