Estrutura de Dados - Tree

Desenvolver Código Orientado a Objetos

Msc. Lucas G. F. Alves
e-mail: lucas.g.f.alves@gmail.com

Senac

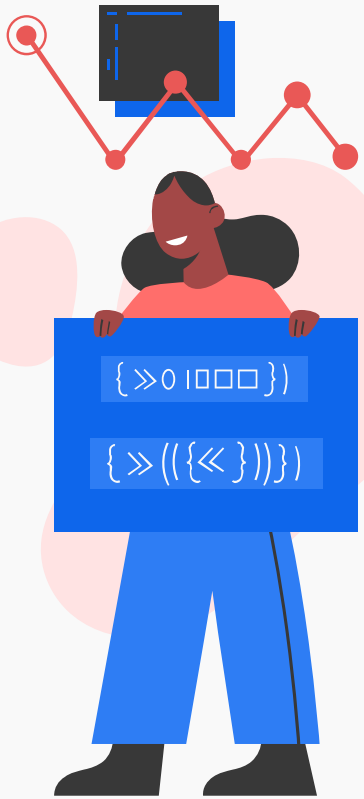
Planejamento de Aula

Entrega Atividade Decks

Árvores

Tipos de Árvores

Dinâmica



Entrega Atividade Decks

Dinâmica Uno

Criar a classe **Carta** com propriedades para cor (azul, amarelo, verde, vermelho) e valor (0-9, Comprar2, comprar4, Inverter, Bloquear).
Criar a classe filha de Decks **BaralhoUno** contendo um array de CartaUno. Esta classe deverá ter métodos para criar o baralho inicial, embaralhar, comprar uma carta do topo e verificar se o baralho está vazio.
Criar uma classe abstrata **Jogador**, com uma mão de cartas (array de CartaUno) e métodos para receber uma carta, jogar uma carta e verificar se tem cartas na mão.
Criar a classe filha **Humano** com o método para o usuário escolher qual carta jogar.
Criar a classe filha **Computador** que irá jogar uma carta válida.
DESAFIO: Criar a classe **Uno** que irá gerenciar o fluxo do jogo, incluindo a criação dos jogadores, a distribuição inicial de cartas, o controle do turno, a pilha de descarte e a lógica para verificar se uma jogada é válida e determinar o vencedor.

Trees

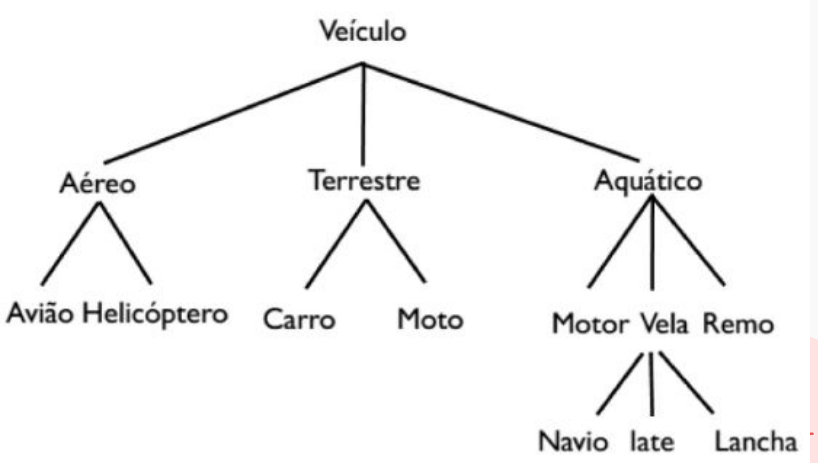
Trees

Introdução

Árvores são estruturas de dados fundamentais em ciência da computação.

Elas são usadas para representar:

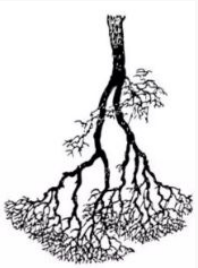
- Hierarquias;
- Estruturas de dados balanceados;
- E são a base para muitos algoritmos importantes;

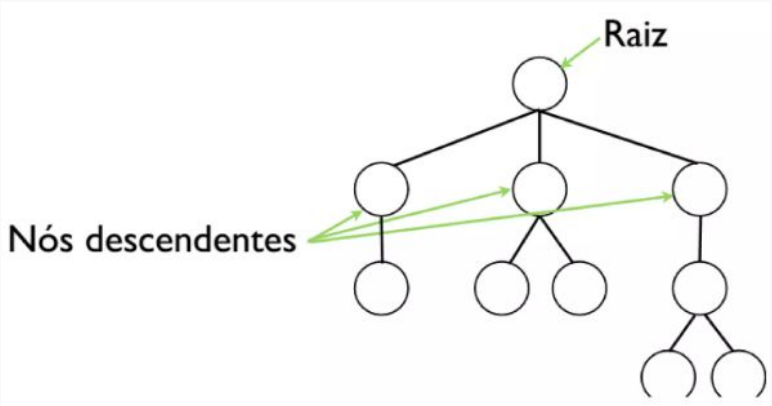


Trees

O que é uma árvore?

Uma árvore é uma estrutura de dados que consiste em **nós conectados por arestas**.
Cada nó em uma árvore possui um **valor** (ou dado) e **zero ou mais nós filhos**.
O **nó no topo** da árvore é chamado de **root ou "raiz"**, os nós sem filhos são cha
de leafs ou **"folhas"**, e os nós que estão entre a raiz e as folhas são chamados d
internos".





Trees

Terminologia

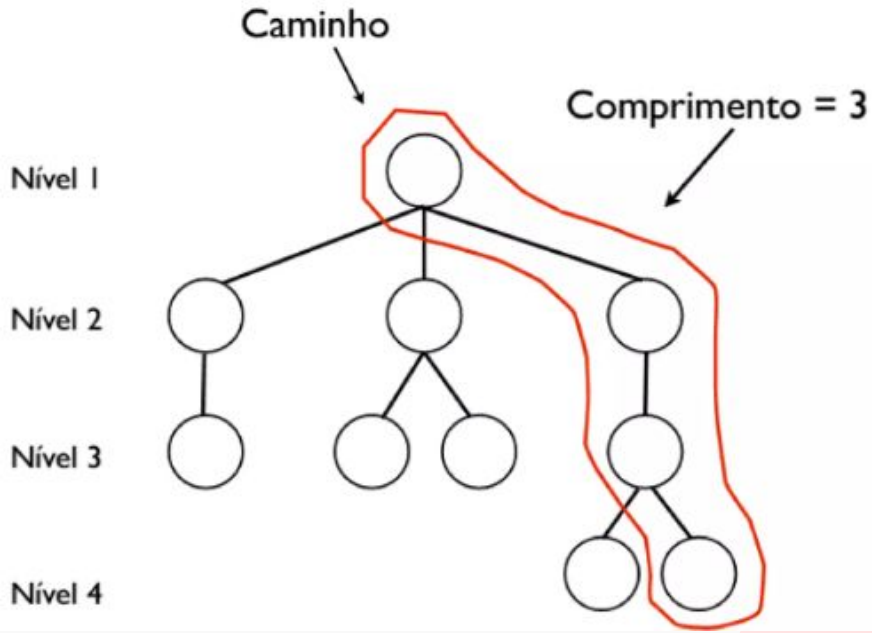
Nó (Node): Um elemento em uma árvore que possui um valor e zero ou mais filhos.
Raiz (Root): O nó no topo da árvore, que liga todos os outros nós alcançáveis.
Filho (Child): Um nó que está diretamente abaixo de outro nó.
Pai (Parent): Um nó que está diretamente acima de outro nó.
Folha (Leaf): Um nó que não possui filhos.
Subárvore (Subtree): Um conjunto de nós e arestas dentro de uma árvore, incluindo a raiz dessa subárvore.

Trees

Terminologia

Nível (Level): A distância entre a raiz e um nó. O nível da **raiz** é 0.

Altura ou profundidade (Height): O comprimento do caminho mais longo da raiz a uma folha.
A altura da árvore é igual ao maior nível de seus nós.



Trees

```
class Node<T> {
        value: T;
        children: Node<T>[] = [];
        constructor(value: T) {
                this.value = value;
        }
        addChild(child: Node<T>): void {
                this.children.push(child);
        }
}
```

```
// Exemplo de uso
const rootNode = new Node("A");
const nodeB = new Node("B");
const nodeC = new Node("C");

rootNode.addChild(nodeB);
rootNode.addChild(nodeC);
```

Tipos de árvores

Tipos de árvores

Tipos Específicos de Árvores

Existem muitos tipos específicos de árvores em ciência da computação, como **árvores binárias**, **árvores de busca binária**, **árvores AVL**, **árvores B**, **árvores Rubro-Negras**, entre outras. Cada tipo de árvore possui características e propriedades únicas que a tornam adequada para diferentes tipos de problemas.

Por exemplo, uma **árvore binária de busca** é uma árvore em que cada nó tem no máximo **dois filhos**, e os nós à esquerda têm valores menores que o nó pai, enquanto os nós à direita têm valores maiores.

Dinâmica

Dinâmica Árvore Genealógica

Crie uma interface **Relacionavel** que defina um método descreverRelacao(): string. Faça com que a classe **Pessoa** implemente essa interface.
Criar a classe **Pessoa**, incorporando os atributos como nome, sobrenome, dataNascimento, genero, outrasRelacoes que é um array de relacionavel.
Crie uma classe **ArvoreGenealogica** que vai gerenciar a nossa árvore. Ela precisa ter um 'nó raiz' (a primeira pessoa da árvore). Implemente métodos para: adicionarFilho(pai: Pessoa, filho: Pessoa): void que adiciona um filho a um pai existente na árvore e encontrarPessoa(nome: string): Pessoa | undefined que busca uma pessoa pelo nome na árvore.
Criem a árvore da sua família e definam qual o **nível** e **profundidade** da sua árvore.
Crie classes derivadas de Pessoa para representar tipos específicos de pessoas ou relações, como **PessoaAdotada** e/ou **Cônjuge**, que implementem descreverRelacao() de formas diferentes.
Desafio: Crie um método na Pessoa para listarRelacoes().

Atividade

Atividade em grupo tipos de árvores

1. Cada grupo deverá escolher UM dos seguintes tipos de árvores para implementar:

Árvore Binária: a base de tudo, **árvore de Busca Binária (BST):** ótima para buscas e ordenação, **árvore AVL:** garante balanceamento para buscas rápidas, **árvore B:** perfeita para grandes volumes de dados em disco (BD), **árvore Rubro-Negra:** outra opção para garantir o balanceamento e performance.

2. Implementar árvore escolhida e simular uma opção de uso da árvore escolhida.

3. Apresentar a escolha da árvore, o algoritmo e a simulação.

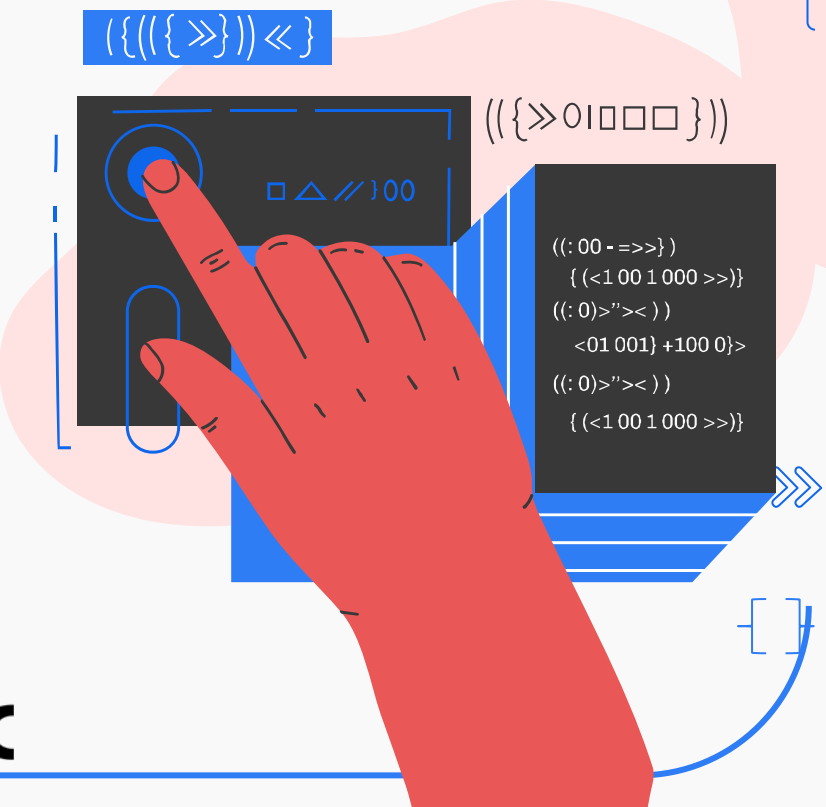
Professor

Lucas G. F. Alves

Senac

Obrigado!

E-mail :lucas.g.f.alves@gmail.com



Senac