Sergio Amaral de Souza

Prática 11
Laboratório de AEDS

Belo Horizonte, Brasil

2024

1 Introdução

Objetivos e descrição do Laboratório:

- No código foi feita a implementação/virtualização do jogo "Torre de Hanoi"na linguagem Processing Java.
- O objetivo era entender a implementação de Stacks/Pilhas em um código, já que os pilares são muito semelhantes com as stacks, pois você só pode tirar de um pilar/pilha o último disco/elemento colocado.

Objetivo do jogo:

- Mover todos os discos da haste inicial para a haste final, utilizando uma haste intermediária, seguindo as regras estabelecidas.

Regras do jogo:

- Apenas um disco pode ser movido por vez.
- Um disco maior não pode ser colocado sobre um disco menor.
- Todos os discos devem estar empilhados em ordem decrescente de tamanho na haste final.

2 Desenvolvimento

2.1 Disco

```
// Construtor
Disco(int posicaoTorre, int ordem, float tamanho) {
    this.posicaoTorre = posicaoTorre;
    this.ordem = ordem;
    this.largura =  tamanho - ( (ordem + 1)  * 10); // Largura aleatória entre 80 e 160
    this.cor = color(random(255), random(255), random(255)); // Cor aleatória
float[] torresX = {width / 6, width / 2, width - 100}; // Posições X das torres
    this.x = torresX[posicaoTorre] + width / 120; // Posição X da torre ajustada
    this.y = height - (ordem + 1) * altura; // Posição Y baseada na ordem
}
```

Figura 1 – Construtor

- Temos a variável "posiçãoTorre"que diz em qual torre está o disco, "ordem"para saber a posição do disco de baixo para cima, a largura, a cor que é feita do tipo color.
- Para finalizar as posições x e y que são inicializadas a partir de um calculo para que o disco fique na posição correta/centralizada.

```
Disco(){
    this.largura=9999999;
}

// Método para desenhar o disco
void desenhaDisco() {
    fill(cor);
    rect(x - largura / 2, y - altura, largura, altura);
}
```

Figura 2 – Construtor "fake"e desenhaDisco

- O metodo disco foi um contrutos feito para que no código eu conseguisse instanciar tres discos(um para cada torre), desta forma, quando o jogo iniciasse conseguiria comparar o tamanho do último disco da primeira torre com os da segunda e da terceira, no caso eles não existem logo foi criado este construtor com a largura bem grande para que o disco consiga ser movido.

2.2 Main

2.2.1 Draw

```
void draw() {
    background(#FFFFFF);

    // Desenha as torres
    fill(#6A4242);
    rect(width/6, height/3, width/60, height);
    rect(width/2, height/3, width/60, height);
    rect(width-100, height/3, width/60, height);

    fill(#000000);
    rect(0, 580, 600, 590);

    // Desenha todos os discos
    for (int i =0; i<3;i++){
        for (Disco disco : potato[i]){
            disco.desenhaDisco();
        }
    }
    Disco d = new Disco();
    if (!potato[torre].isEmpty()){
        d = potato[torre].peek();
    }
    d.desenhaDisco();

    //Para que o disco siga o mouse
    if (click == -1) {
        d.x = mouseX - d.largura/2;
        d.y = mouseY - d.altura/2;
    }
}
```

Figura 3 – Draw

- Importante ressaltar que no começo do código tem o trecho : import java.util.Stack;. Sem ele nada seria possível.

- Nesta função são desenhados os três pilares e os discos(também instanciamos os discos fakes que serão usados para comparação de maior ou menor que possibilitara a movimentação do disco de um pilar para outro.) e por fim é implementado um pequeno algoritmo para que o disco se movimente de acordo com o mouse.

2.2.2 mousePressed

```
if(mouseX<200&&click==1){
    click*=- 1;
    torre =0;
    dr = potato[torre].peek();


}
else if(mouseX>200&&mouseX<400&&click==1){
    click*=-1;
    torre =1;
    dr = potato[torre].peek();

}
else if(mouseX>400&&mouseX<600&&click==1){
    click*=-1;
    torre =2;
    dr = potato[torre].peek();

}
```

Figura 4 – Selecionar

```
else if(mouseX<200&&click== -1){

    if(dr.largura<=a[0].largura){
        potato[torre].pop();
        dr.x = 105;
        dr.y = height - (potato[0].size() + 1) * 20;
        click*=-1;

        a[0]=potato[0].push(dr);
        torre =0;
    }
}

else if(mouseX>200&&mouseX<400&&click== -1){
    if(dr.largura<=a[1].largura){
        potato[torre].pop();


        dr.x =305;
        dr.y = height - (potato[1].size() + 1) * 20;
        click*=-1;
        a[1]=potato[1].push(dr);
        torre =1;
    }
}

else if(mouseX>400&&click== -1){
    if(dr.largura<=a[2].largura){

        potato[torre].pop();
        dr.x = 505;
        dr.y = height - (potato[2].size() + 1) * 20;
        click*=-1;
        a[2]=potato[2].push(dr);
        torre =2;
    }
}
```

Figura 5 – Colocar

- Nesta função foram feitos 6 IFs, 3 para quando selecionar o último disco de alguma torre, e outros tres para coloca-los em outra torre. Dentro dos seis IFs foi feita uma divisão por coordenada em 3 partes, em cada parte há 1 if de selecionar e outro de colocar.
- Sendo assim, 2 são para o primeiro pilar(posiçãox= 0 a 200), dois para o segundo (posiçãox= 201 a 400) e dois para o terceiro (posiçãox= 401 a 600).
- A varável click é para verificar se você está selecionando ou colocando o disco. Quando for 1 = selecionar/ -1 = seguir o mouse e colocar.

3 Resultados

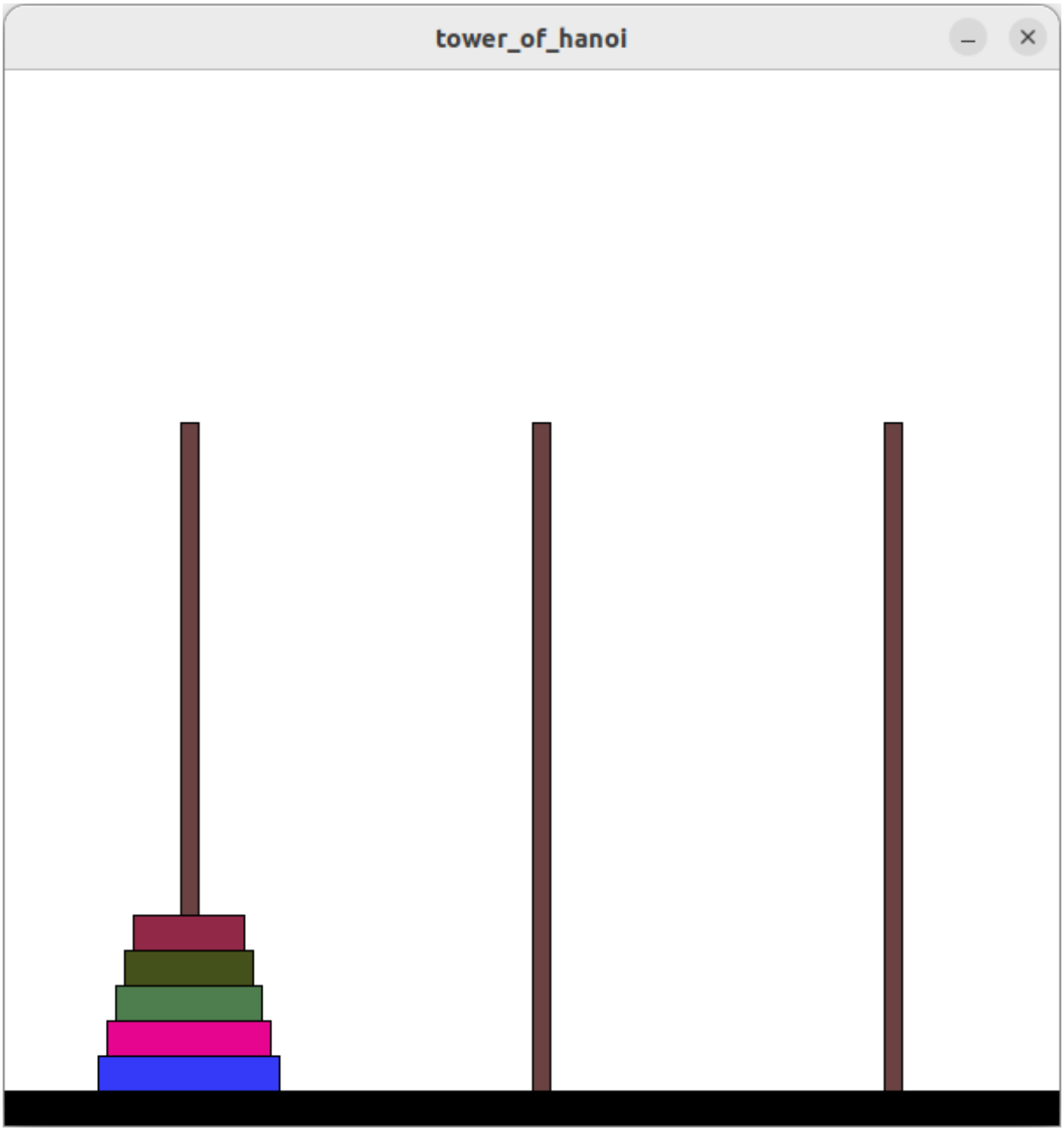

Figura 6 –

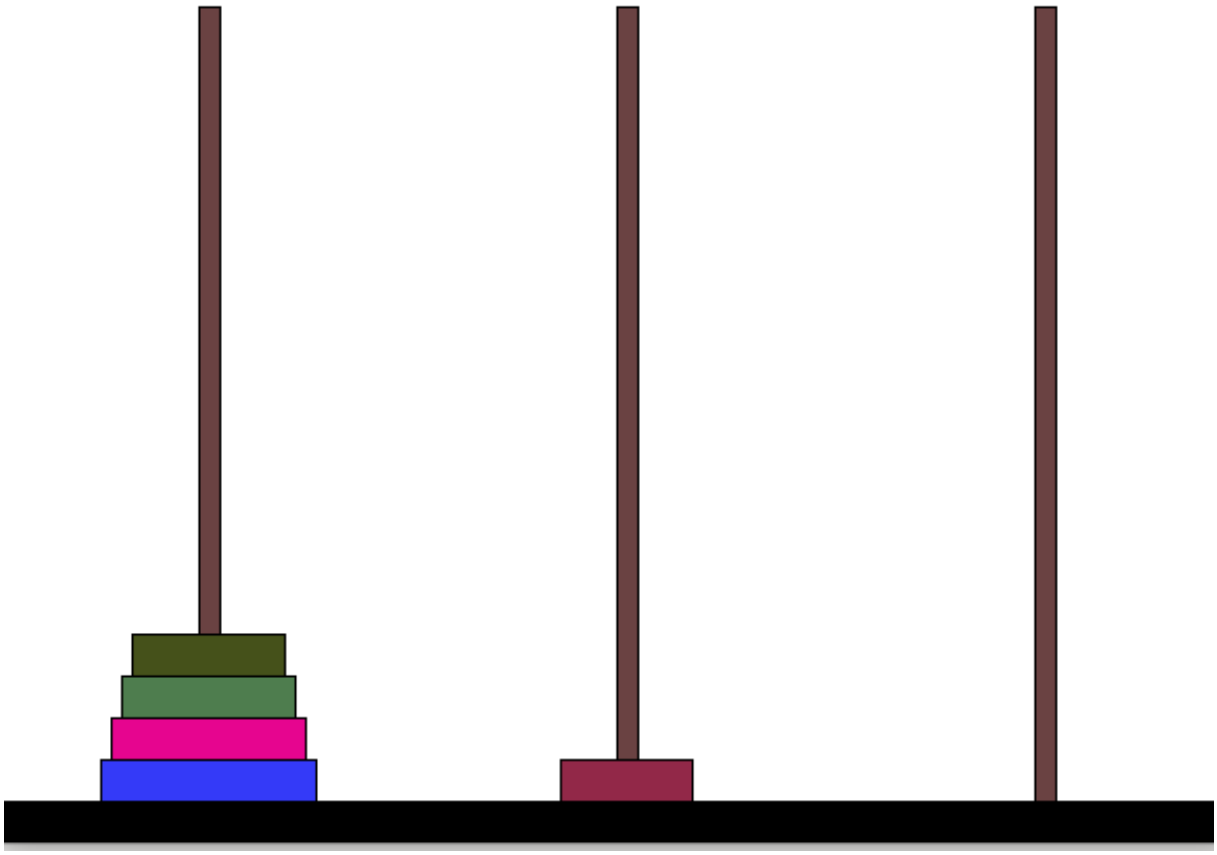

Figura 7 –

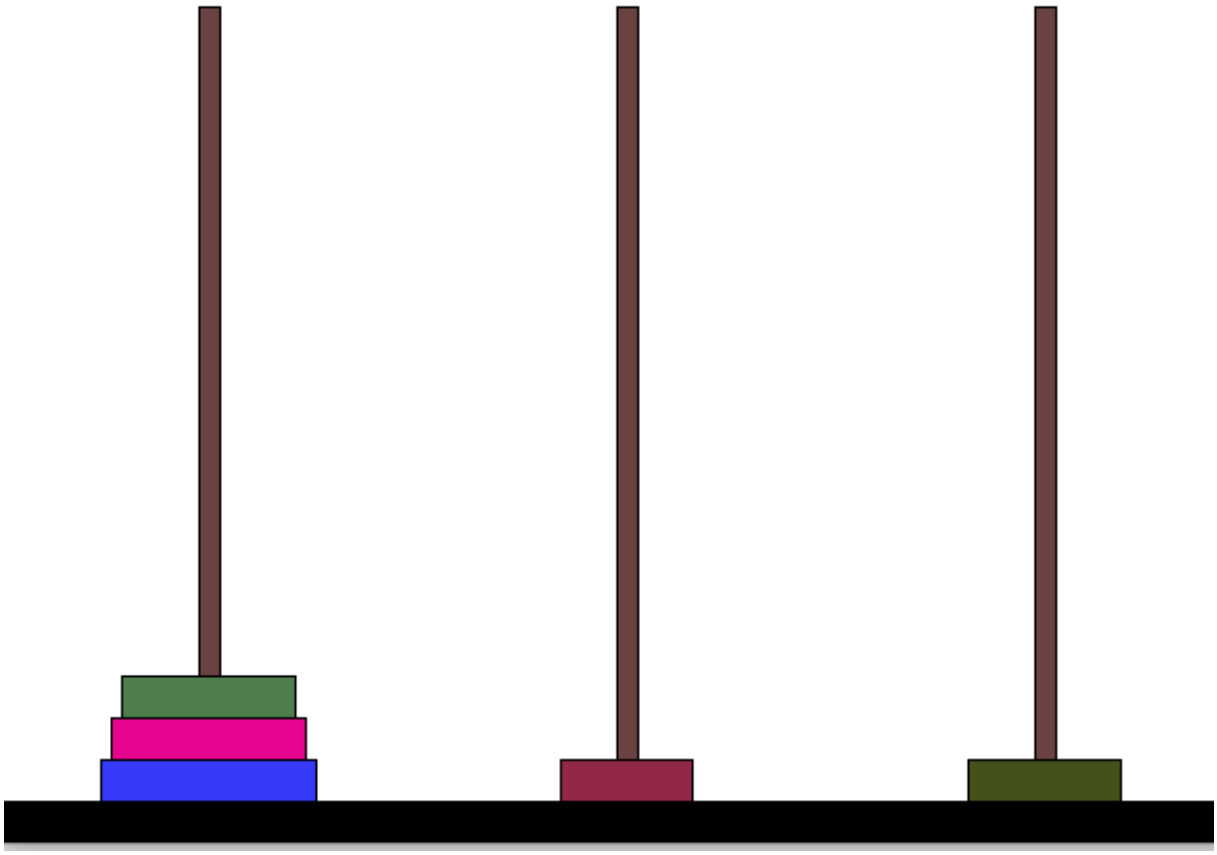

Figura 8 –

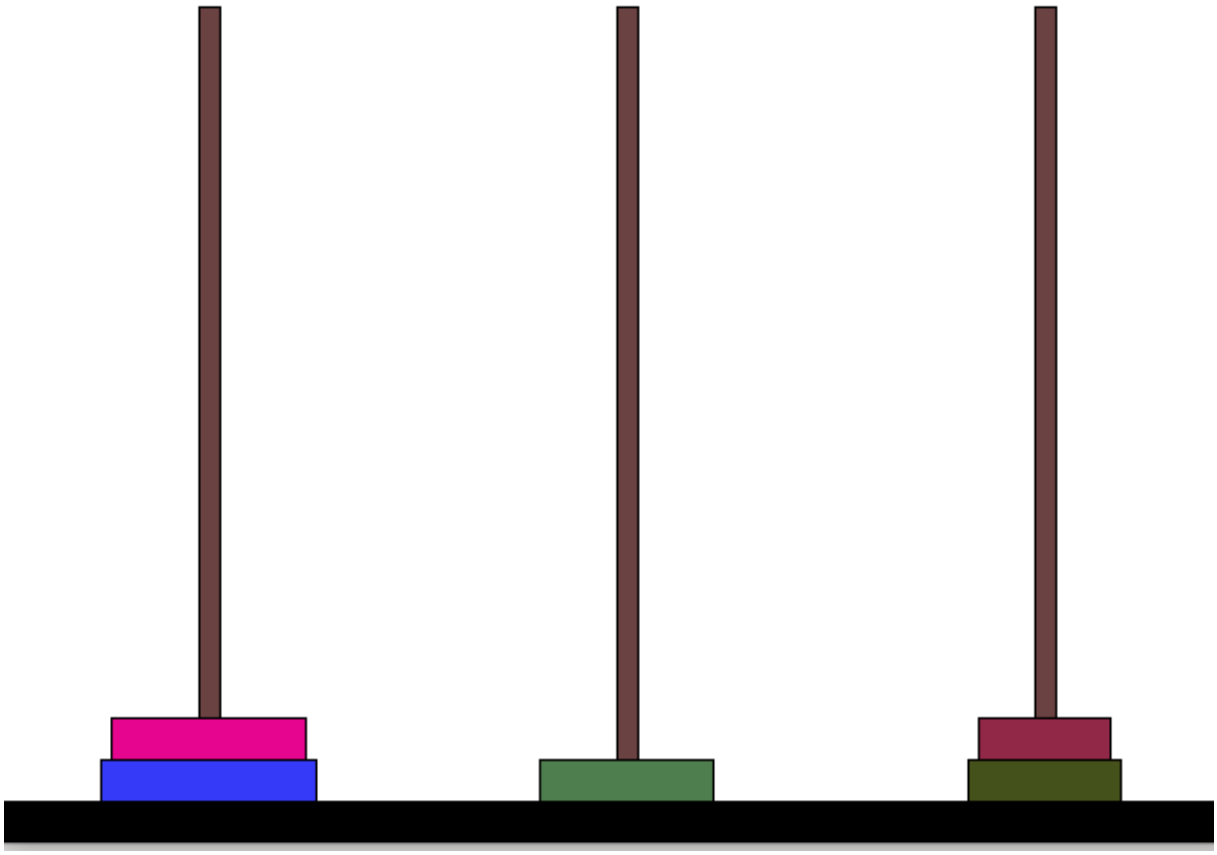

Figura 9 –

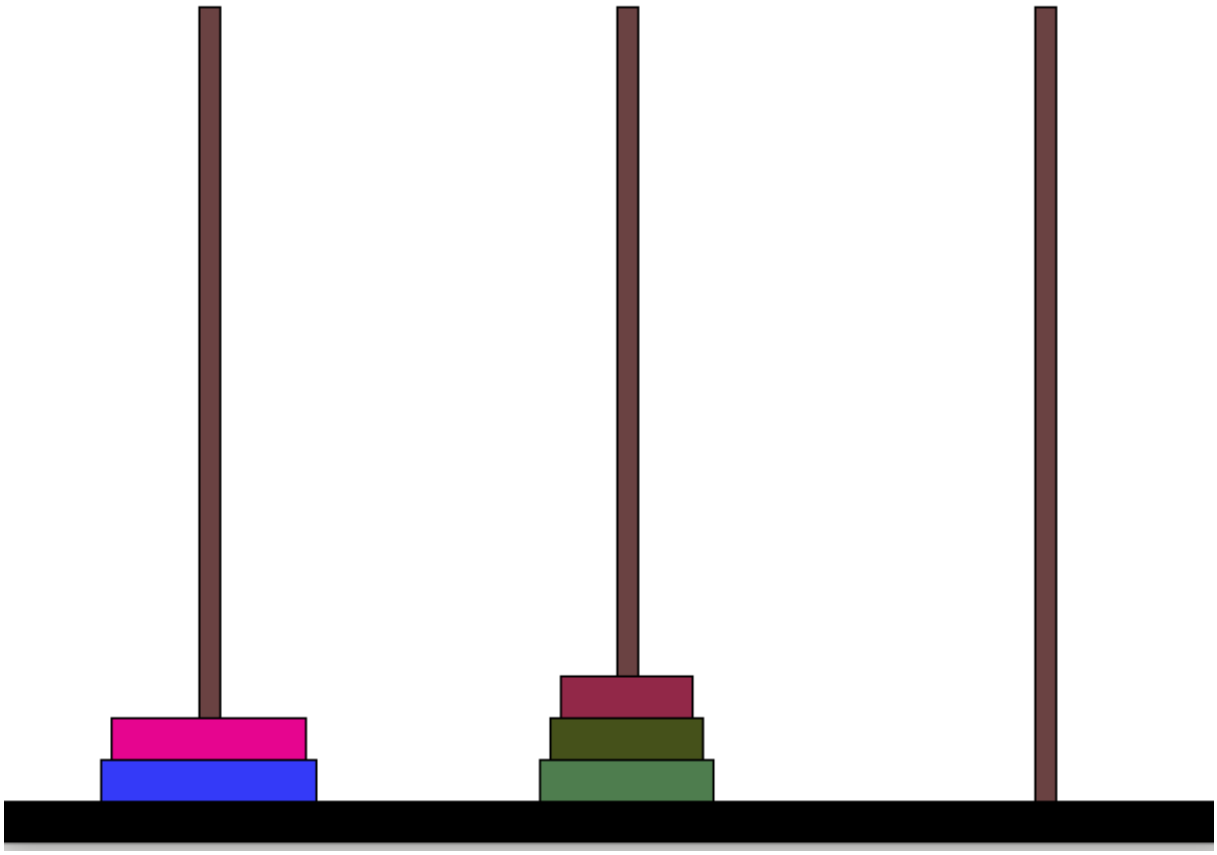

Figura 10 –

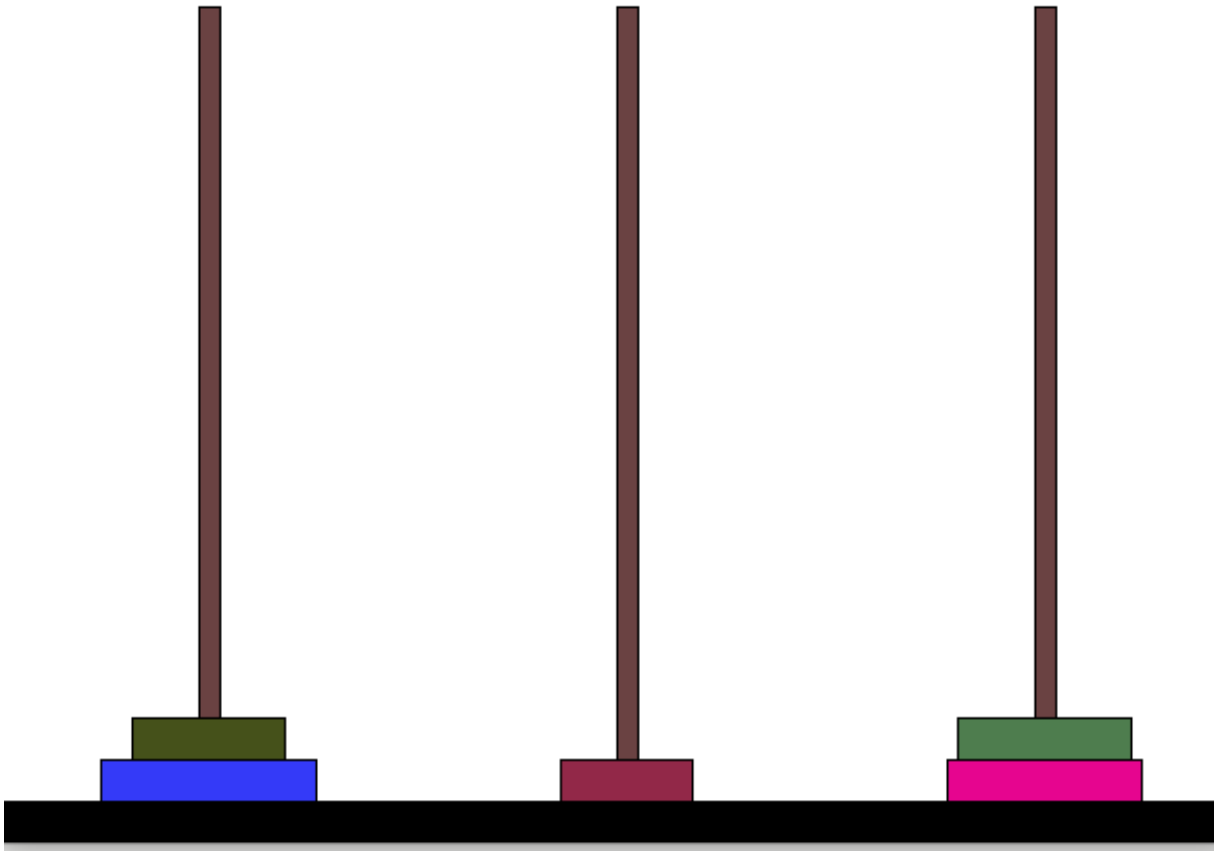

Figura 11 –

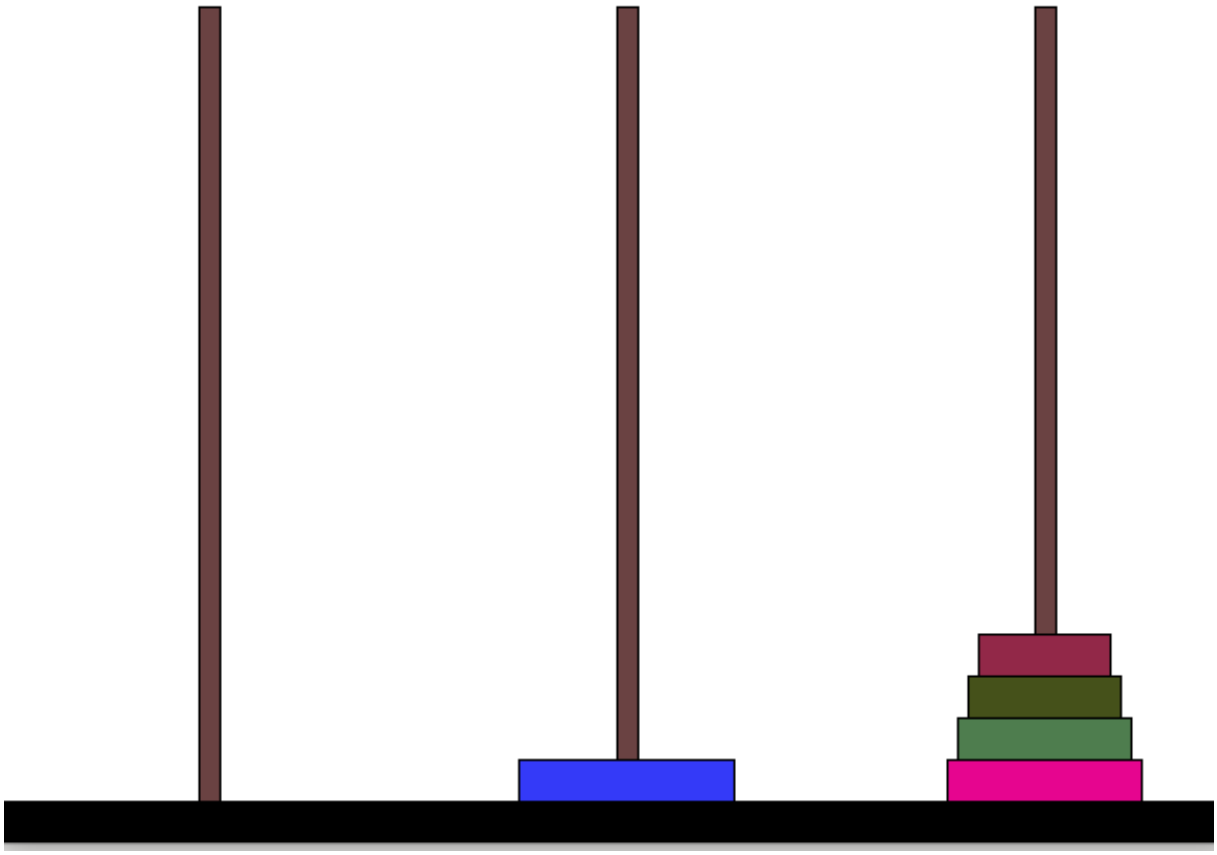

Figura 12 –

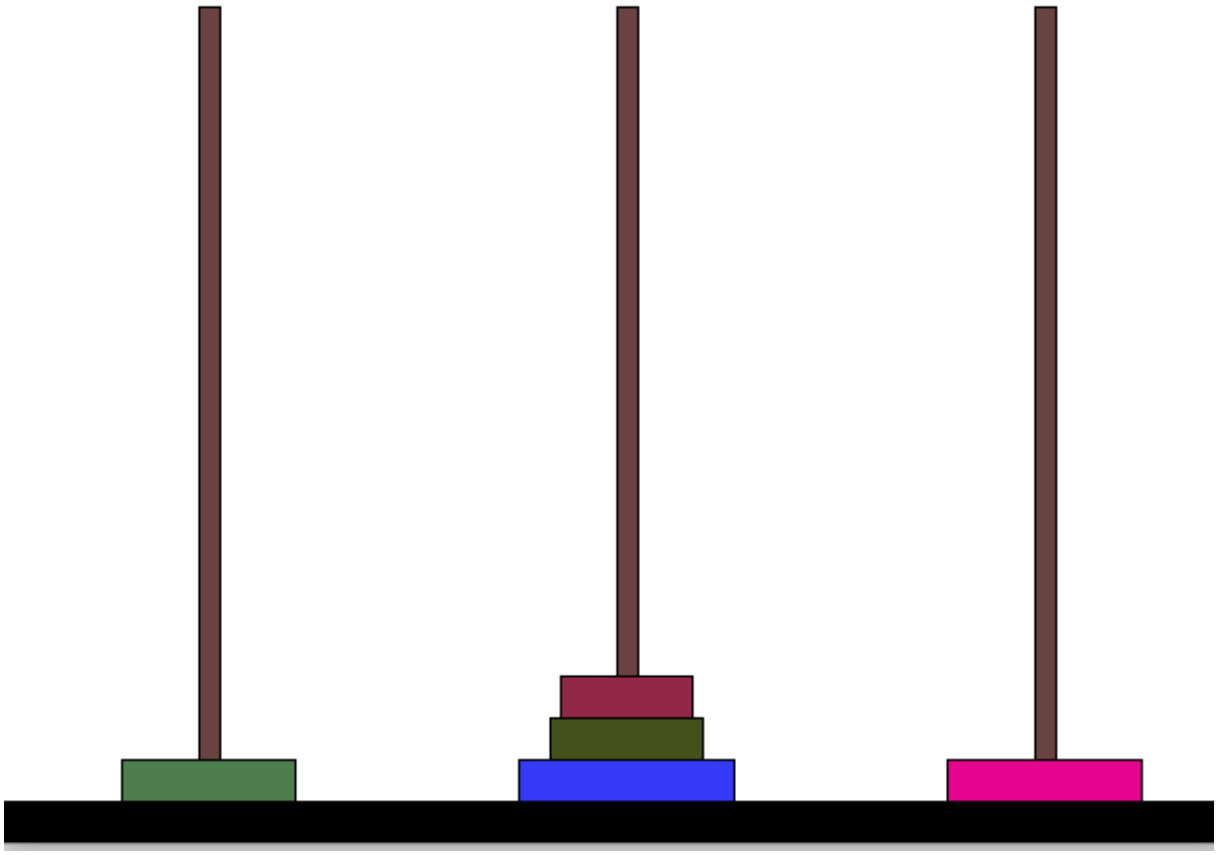

Figura 13 –

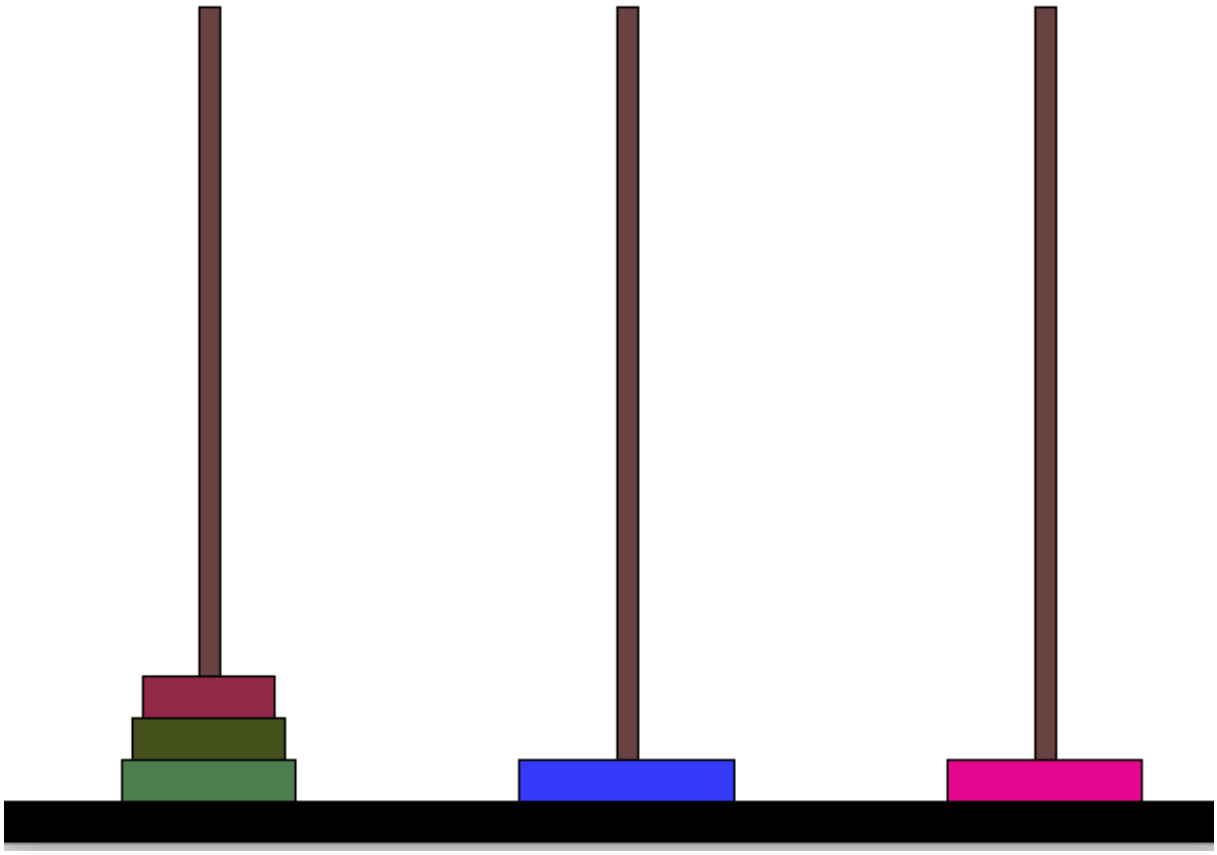

Figura 14 –

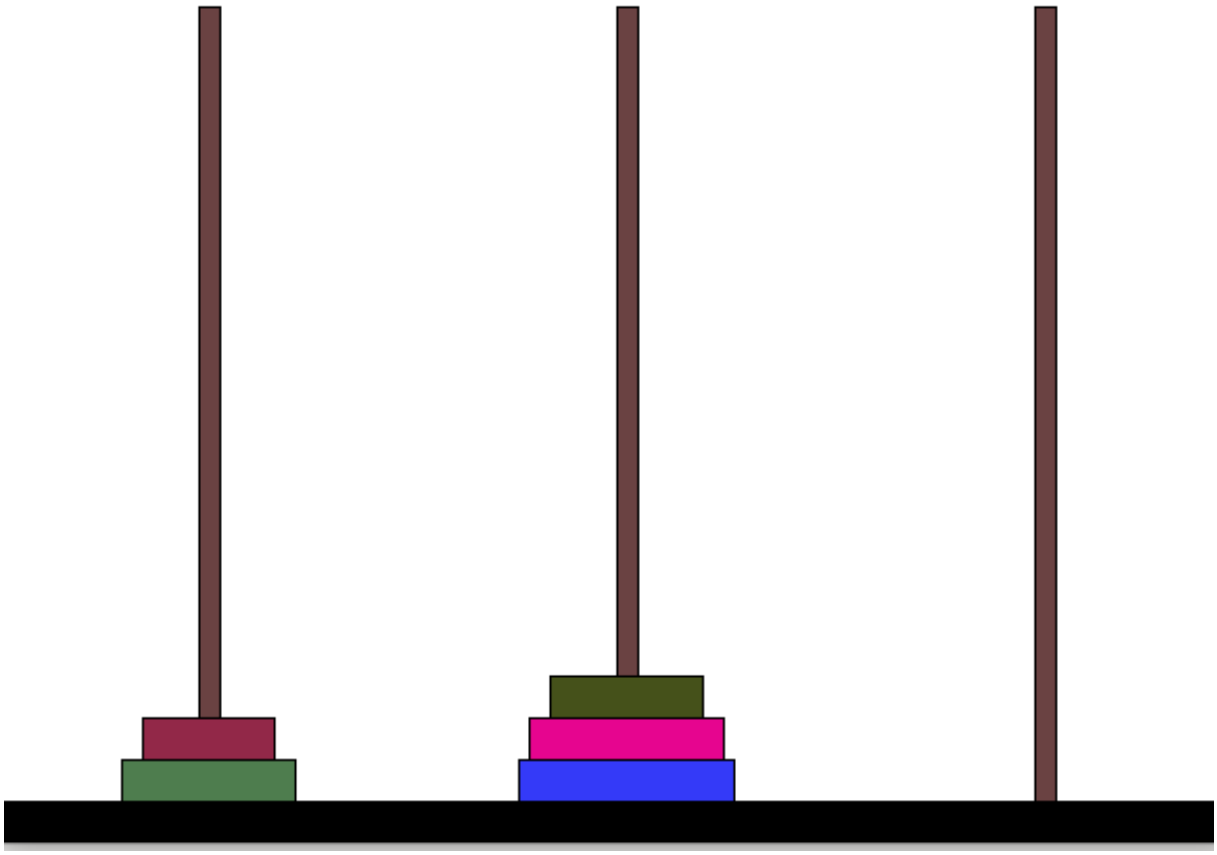

Figura 15 –

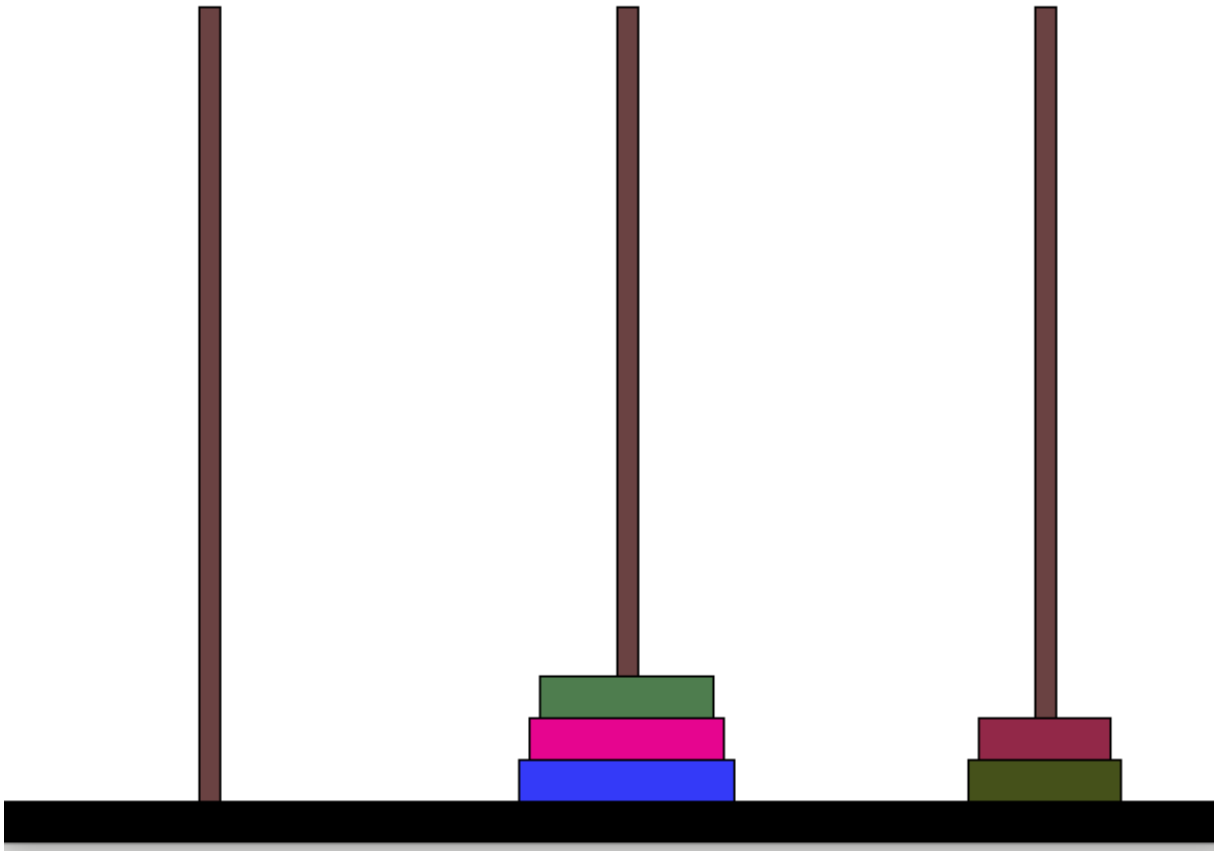

Figura 16 –

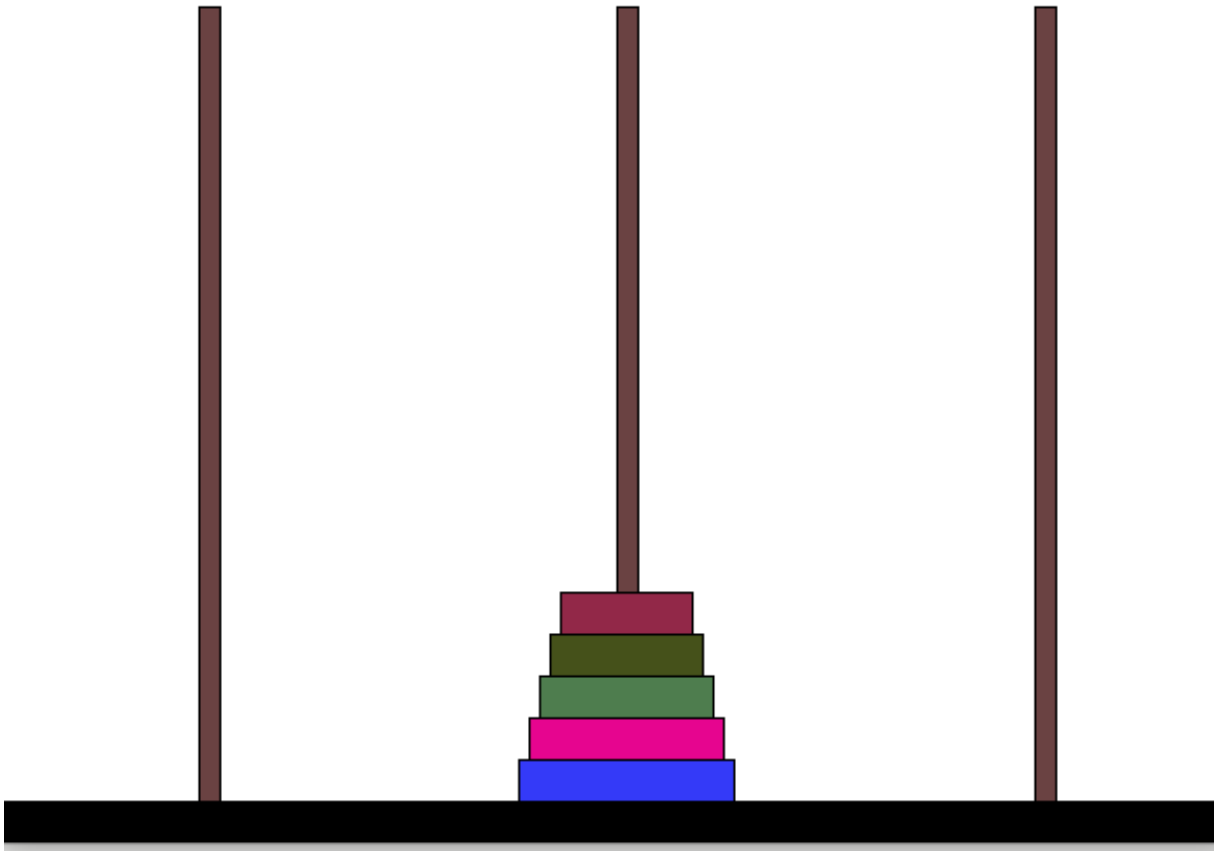

Figura 17 –

4 Conclusão

- Felizmente todos os resultados foram obtidos com êxito.
- A princpal dificuldade encontrada foi entender que eu teria que fazer o construtor fake para que quando eu tentasse passar um disco para uma torre vazia não desse o erro: NullPointerException e também determinar qual seria a nova posição y do disco.
- Aprendi bastante sobre as pilhas e seu funcionamento, também aprendi uma lição de antes de escrever qualquer coisa no código devo pensar ou pelo menos ter uma noção

minima de todas as partes do código e os possíveis erros obvio que ocorreriam como esse do NullPointerException.