1)

O objetivo do trabalho é desenvolver um projeto em Java que permita o usuário jogar o jogo conhecido como Torre de Hanói. Para tal, na classe que modela o comportamento das 3 torres e implementa as regras do jogo, eu usei 3 pilhas para servirem como torres. Pilhas são o melhor tipo abstrato de dado para esse jogo, já que funciona segundo o princípio LIFO (last in first out), assim como os discos nas torres do jogo.

2)

O método de movimento de discos que eu fiz recebe duas Strings, que podem ser apenas “a”, “b” ou “c”. Ele checa as entradas e cria referências que apontam para as torres envolvidas no movimento em questão. Em seguida ele checa se o movimento é permitido, ou seja, se a torre da qual se deseja remover um disco não está vazia, e se o disco no topo da outra torre é maior do que o disco movido, ou se a outra torre está vazia. Se essas condições forem atendidas, ele remove um disco de cima da primeira torre e adiciona no topo da segunda.

Para fazer o método toString, a solução que eu pensei foi criar um arranjo de Strings para cada pilha como variáveis de instância. Esses arranjos possuem tamanho igual a quantidade de discos, e armazenam o valor de suas respectivas pilhas como String (e se houver um espaço vazio, armazenam “|”). Eu achei essa abordagem mais fácil, uma vez que nas pilhas que modelam as torres não é possível acessar os elementos abaixo do topo sem remover discos. Então, a cada vez que um movimento é realizado, o programa atualiza esses arranjos de Strings de forma a mantê-los condizentes com as pilhas. Tendo esses arranjos que estão sempre se atualizando a cada movimento como variáveis de instância, fazer o toString é só uma questão de organizar os arranjos em uma String só e retorná-la.

3)

```
C:\Users\USUARIO\OneDrive - PUCRS - BR\Programas\Java\VSCode\TADS\Torre hanoi>java -jar TorreHanoiLuisMarquetti.jar
Pra comecar, digite a quantidade de discos que desejas no jogo. Os valores permitidos sao de 3 a 7 discos: 4

1   |   |
2   |   |
3   |   |
4   |   |
=========
A   B   C
Movimentos ate agora: 0
Indique de que torre tirar o disco. Digite 'a' para torre da esquerda, 'b' para a do meio e 'c' para a da direita: Digite uma entrada aceitavel: a

Agora indique para qual torre deseja mover o disco:
b

|   |   |
2   |   |
3   |   |
4   1   |
=========
A   B   C
Movimentos ate agora: 1
Indique de que torre tirar o disco. Digite 'a' para torre da esquerda, 'b' para a do meio e 'c' para a da direita: a

Agora indique para qual torre deseja mover o disco:
c

|   |   |
|   |   |
3   |   |
4   1   2
=========
A   B   C
Movimentos ate agora: 2
Indique de que torre tirar o disco. Digite 'a' para torre da esquerda, 'b' para a do meio e 'c' para a da direita: b

Agora indique para qual torre deseja mover o disco:
c

|   |   |
|   |   |
3   |   1
4   |   2
=========
A   B   C
Movimentos ate agora: 3
Indique de que torre tirar o disco. Digite 'a' para torre da esquerda, 'b' para a do meio e 'c' para a da direita:
```

```
Indique de que torre tirar o disco. Digite 'a' para torre da esquerda, 'b' para a do meio e 'c' para a da direita: b

Agora indique para qual torre deseja mover o disco:
c

|   |   |
|   |   2
|   |   3
1   |   4
=========
A   B   C
Movimentos ate agora: 26
Indique de que torre tirar o disco. Digite 'a' para torre da esquerda, 'b' para a do meio e 'c' para a da direita: a

Agora indique para qual torre deseja mover o disco:
c
Parabens, voce conseguiu!!!

|   |   1
|   |   2
|   |   3
|   |   4
=========
A   B   C
Total de movimentos: 27
Desejas jogar de novo? Digite 1 pra sim e 0 pra nao.
```

4)

Houveram algumas dificuldades encontradas no desenvolvimento do programa. Uma delas foi pensar em todos os testes que devem ser feitos no método movimento, que é

responsável por remover o disco do topo de uma pilha e adicionar em outra, de modo a garantir a obediência às regras do jogo. Mas a maior dificuldade encontrada foi achar um jeito de imprimir a configuração do jogo. Foram várias tentativas até eu perceber que eu tinha que usar alguma variável auxiliar para guardar as informações de cada torre.

A complexidade da solução em notação O seria O(n), com n sendo a quantidade de discos escolhida. Isso pois todas as estruturas de repetição do algoritmo são executadas uma quantidade de vezes proporcional a quantidade de discos, e não há nenhum laço dentro de outro.