Object Oriented Programming

2022 final report

Deadline:2022/8/2 20:00

1 Tower of Hanoi

Tower of Hanoi is a simple game (see Fig. 1). There are three pillars and some disks with difference radius. You are allowed move disks one by one. And you are allowed to put a disk into a pillar if the top disk of the destination pillar is larger than the disk you want to put or the destination pillar has no disk. Initially some disks are put in the leftmost pillar. The task is to move all disks to the rightmost pillar.

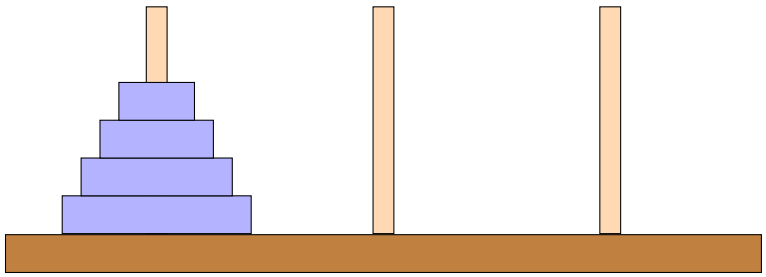

Fig. 1: Tower of Hanoi with 4 disks

The algorithm for solving the task is recursively described. For moving n disks from a pillar f to another t, you move $n-1$ disks from a pillar f to the remaining pillar o, move one disk from f to t, and finally move $n-1$ disks from o to t. The procedure which actually move a disk is restricted for the $n = 1$ case.

The algorithm is shown in Algorithm 1. For a game with n disks, the game starts with calling `moveDisk(0, 2, n)`, where the leftmost and the rightmost pillars are indexed as 0 and 2.

Algorithm 1 Move disks

```
procedure moveDisks(f, t, n)
    if n = 1 then
        Move one disk from f to t
        return
    end if
    o is a pillar o! = f and o! = t
    moveDisks(f, o, n − 1)
    Move one disk from f to t
    moveDisks(o, t, n − 1)
end procedure
```

2 Class Planning

Let us start to plan classes for solving Tower of Hanoi. The game needs to control motions of disks between pillars. The `Disk` class is for disks with different radius. The `Pillar` class is for pillars, which have a stack of disks. The `Pillar` class needs to prevent a larger disk from being put above a smaller one. The main class, `Hanoi`, has three pillars.

The template for this project can be get through the following URL.
`https://github.com/oop-mc-saga/Hanoi`

2.1 `Disk` class

The `Disk` class in `parts` package simply keeps the radius of a disk. The class implements `Comparable` interface.

Exercise 2.1 Complete the method `compareTo()`.

Exercise 2.2 All classes in Java are extension of `Object` class. The `Object` class has `toString()` method, so all classes have. Explain the purpose of this method. And define the function for the case of `Disk` class.

2.2 `Pillar` class

The `Pillar` class in `parts` package stores disks in a stack. The `java.util` package has `Stack` class. However, the API reference recommends to user `Deque` interface and its implementations instead. The `Pillar` class uses `Deque` for storing disks.

Exercise 2.3 Read the API reference, and explain the following methods of `Deque` interface.

- `isEmpty()`
- `addFirst()`
- `removeFirst()`
- `getFirst()`

Exercise 2.4 The `canPush()` method investigates whether the disk specified as an argument can be put to this pillar. If the pillar is not empty, you have to decide true or false by investigating the size of the top disk. Implement the `canPush()` method.

2.3 `Hanoi` class

The `Hanoi` class is contained in `model` package. This class is the main part of Tower of Hanoi. The class has an array of three `Pillar` instances constructed in the Constructor. The pillars are indexed as 0, 1, and 2 from the leftmost to the rightmost. The number of disks are passed through the Constructor and `Disk` instances are placed in the leftmost pillar.

By calling `start()` method, the task is started by calling `moveDisk()` method.

Exercise 2.5 In this game, you are allowed to move disks one by one. So we implement `moveSingleDisk()` method for moving only one disk from f to t, where f and t are indexes of the pillar array.

Exercise 2.6 The `Hanoi` class has a boolean variable `debug`. If true, the class prints the current state to the standard output at every time calling the `moveSingleDisk()` method. Add this function (print state) in the `moveSingleDisk()` method.

Exercise 2.7 Implement the `moveDisks()` method according to Algorithm 1.

3 Running the project

The `Main` class in `default package` has `main()` method only and is used for running the `Hanoi` class.

Exercise 3.1 Implement `main()` method for cases with $n = 3$ and $n = 4$. And confirm that the task is correctly completed.

4 Theoretical Analysis

Let us discuss the number of disk motions necessary for completing the task.

Exercise 4.1 Let C_n be the number of disk motions necessary for moving n disks from one pillar to another. Derive the recursive relation for C_n based on the Algorithm 1. And show C_n as a function of n by solving the recursive relation.

Exercise 4.2 The `Hanoi` class has a variable `numberOfMove`. Modify `moveSingleDisk()` method for incrementing `numberOfMove`. And confirm the theoretical result with the value of `numberOfMove`.

5 How to submit your report

Your report should be prepared as a PDF file and submitted through *Teams*.

- Prepare your report digitally with such as `Word` or LaTeX.
- The file name should be `studentID.pdf`.
- Contain description of understanding and solutions of tasks, programs, and program outputs.
- Show multiple examples.
- Improve your program's readability with suitable naming of classes, variables, and methods. Also add suitable comments in programs.
- Write your document neatly with correct Japanese or English.
- Cite suitable references.

6 Scoring

C: Requested programs are coded, but not explained or not suitably constructed as OOP.

B: Classes are correctly defined and codes are well organized.

A: Class planning and workflows are well discussed in the report.

S: Over the level A, some notable points are found.