Assignment 3

Deadline: March 20, 2022, 11:59 pm

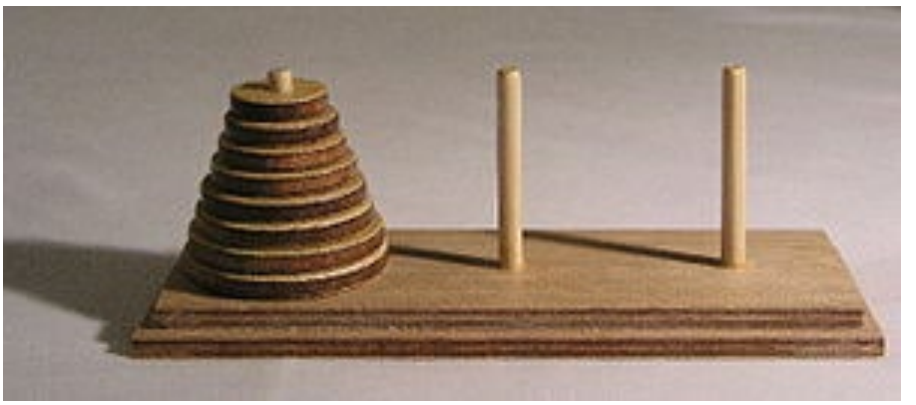
Learning objectives

- Using Interfaces
- Using Stacks

Important note: Your assignment will not be graded and will get zero if:
- **Names are missing at the beginning of all java files (see Assignment files).**
- **Your solution files are not submitted in the correct format (see the instruction page).**

Introduction

In this assignment, we are working on a new game called Hanoi Tower. In this game, the Player is asked to move *n* disks from tower 1 to tower 3 using tower 2 as temporary storage. The disks vary in size, initially arranged from the largest at the bottom to the smallest at the top at tower 1. The rule is rather simple; no large disk can be placed on the top of a smaller disk while making the moves.



Hanoi Tower with 8 disks [1]

In this game, we will assume two human players will play the game. The first Player is given a chance to finish the game within a certain upper bound for the number of moves; if he can do it, he gets one score. Then the game resets to the initial state, and the second Player will start his turn and try to achieve the same goal. Once the second Player has finished, the scores for both players will display, and they will have a chance to play again, preserving their scores. However, the number of disks will be increased by 1 in each following game.

If you don't know the game, read about it [1]. Try it at [2].

The following are sample runs expected from your code:

Sample run 1:

```
>java HanoiTower 2
************************************************************
*      This is a test                                      *
*                                                          *
*                                                          *
*                                                          *
************************************************************


Player 1 turn
Tower 1
-
--
Tower 2


Tower 3



Your goal is to move 2 disks from tower 1 to 3
Only one simple rule: no large disk on the top of a smaller one!
Enter the source and the destination towers each on a single line:
1
2
Tower 1

--
Tower 2

-
Tower 3


Moves played 1  Max 6
Enter the source and the destination towers each on a single line:
1
3
Tower 1


Tower 2

-
Tower 3

--

Moves played 2  Max 6
```

```
Enter the source and the destination towers each on a single line:
2
3
Tower 1


Tower 2


Tower 3
-
--

Moves played 3  Max 6
You did it within the allowed number of moves!
Your score is 1
=================

Player 2 turn
Tower 1
-
--
Tower 2


Tower 3




Your goal is to move 2 disks from tower 1 to 3
Only one simple rule: no large disk on the top of a smaller one!
Enter the source and the destination towers each on a single line:
1
3
Tower 1

--
Tower 2


Tower 3

-

Moves played 1  Max 6
Enter the source and the destination towers each on a single line:
3
2
Tower 1

--
```

```
Tower 2

-
Tower 3



Moves played 2  Max 6
Enter the source and the destination towers each on a single line:
1
3
Tower 1


Tower 2

-
Tower 3

--

Moves played 3  Max 6
Enter the source and the destination towers each on a single line:
2
3
Tower 1


Tower 2


Tower 3
-
--

Moves played 4  Max 6
You did it within the allowed number of moves!
Your score is 1
================
Player 1 score 1
Player 2 score 1
Do you want to play again?! enter y or Y
Number of disks will be increased by 1!!
Y

Player 1 turn
Tower 1
-
--
---
Tower 2
```

```
Tower 3



Your goal is to move 3 disks from tower 1 to 3
Only one simple rule: no large disk on the top of a smaller one!
Enter the source and the destination towers each on a single line:
```

….

The game continues

Sample run 2:

```
java HanoiTower 1
************************************************************
*      This is a test                                      *
*                                                          *
*                                                          *
*                                                          *
************************************************************


Player 1 turn
Tower 1
-
Tower 2

Tower 3


Your goal is to move 1 disks from tower 1 to 3
Only one simple rule: no large disk on the top of a smaller one!
Enter the source and the destination towers each on a single line:
1
2
Tower 1

Tower 2
-
Tower 3

Moves played 1  Max 2
Enter the source and the destination towers each on a single line:
2
3
Tower 1

Tower 2

Tower 3
-
Moves played 2  Max 2
You did it within the allowed number of moves!
Your score is 1
=================

Player 2 turn
Tower 1
-
Tower 2

Tower 3


Your goal is to move 1 disks from tower 1 to 3
Only one simple rule: no large disk on the top of a smaller one!
```

```
Enter the source and the destination towers each on a single line:
1
2
Tower 1

Tower 2
-
Tower 3

Moves played 1  Max 2
Enter the source and the destination towers each on a single line:
2
1
Tower 1
-
Tower 2

Tower 3

Moves played 2  Max 2
You finished the allowed number of moves!
Your score is 0
=================
Player 1 score 1
Player 2 score 0
Do you want to play again?! enter y or Y
Number of disks will be increased by 1!!
N
```

Sample run 3:

```
java HanoiTower 5
************************************************************
*      This is a test                                      *
*                                                          *
*                                                          *
*                                                          *
************************************************************


Player 1 turn
Tower 1
-
--
---
----
-----
Tower 2




Tower 3




Your goal is to move 5 disks from tower 1 to 3
Only one simple rule: no large disk on the top of a smaller one!
Enter the source and the destination towers each on a single line:
1
2
Tower 1

--
---
----
-----
Tower 2




-
Tower 3




Moves played 1  Max 62
```

```
Enter the source and the destination towers each on a single line:
1
2
Invalid move!!
Tower 1

--
---
----
-----
Tower 2




-
Tower 3




Moves played 1  Max 62
Enter the source and the destination towers each on a single line:
3
2
There is/are no disk/'s at tower 3!!
Tower 1

--
---
----
-----
Tower 2




-
Tower 3




Moves played 1  Max 62
Enter the source and the destination towers each on a single line:
```

….

The game continues

Implementation

We are now ready to program our solution. The following is a description of each class/interface. Please notice these important notes:

- ***All classes/interfaces should include a full header discussing the details of the class and how it is used within the assignment (At least 40 words).***
- ***All methods should be fully documented(At least 30 words each).***

  // all methods should be documented with: purpose of the method, the input and the output of the method, and <u>where it is used in the assignment</u>
- ***Do not copy from the text in this document into your documentation, you should write in your own words.***

<u>Classes/Interfaces of the assignment are:</u>

- Stake

Stack is an interface that contains the following methods:

```
boolean isEmpty( );
void push( Object o );
Object pop( );
Object peek( );
int size( );
```

- LinkedStake

LinkedStack is an implementation of the interface Stack. It will have two instance variables; a reference to the linked structure and the top/size. (You need an extra class for the Linked structure, the class Elem)

Make sure to add all the necessary conditions within each method as needed!

- Player

Player is an interface. It defines two methods, the method play, and the method getScore. Play is void and has one input parameter, a reference to a HanoiTowerGame. getScore will return an int value but take no input parameters.

- HumanPlayer

HumanPlayer is a class that implements the interface *Player*. An instance variable should be used to track the score of the Player; it initially has a value of zero.

In its parameter, the method play receives a reference (say *game*) to a HanoiTowerGame. It first prints out the game board (use toString for the *game*) and some simple instructions (see sample runs).

Then in a loop, the method will ask the users to input the source and destination tower values which will be passed to the *game* play method using the parameter reference (e.g. game.play(s,d)). Please note the numbers entered should be between 1 to 3, however when passed to the method play it should be decremented by 1 as they will be treated as array indices there.

This will continue while the game state is PLAYING. The board is printed within this loop and the current number of moves and the max number of allowed moves.

Once the game state changes, the player score is incremented by 1, and an appropriate message is printed if the state is WINNER. If the state is LOSER, it means the user finished his moves, and an appropriate message is printed. (Note: game state changes at the play method inside HanoiTowerGame class)

The score of the Player is printed, and the method finishes.

- GameState

This is an enumeration class that has three values: PLAYING, LOSER, and WINNER.

- HanoiTower

This class implements playing the game. Only one main method is there.

Display the student info using StudentInfo.java class

Check the arguments and use only the first one for the number of disks which can be any integer from 1 and up. The default value is 3 if no arguments are there.

Define two human players and call the play method for each Player within a loop that asks if the players want to play another round where the number of disks is increased by 1.

This code is given to you.

- HanoiTowerGame

This method should define an array of Stacks of size 3 that will represent the three towers. Other instance variables are needed to track the current moves (*level*) and the max number of allowed moves (*maxLevels*). The max number of allowed moves is calculated based on the number of disks as follows:

maxLevels =2*((int)Math.pow(2,disks) -1);

In addition, we will have the following instance variables: gameState (holds the game state) and disks (the number of disks in the game being played).

Methods/constructors:

- Constructor that takes one int parameter (number of disks to play): this will initialize all the instance variables and the three stacks (towers).
- getDisks, getLevel, maxLevels, getGameState return the corresponding value of the member variables.
  - ***Note that all instance variables are private unless otherwise specified.***
- getDisksAtTower(int i) will return the number of disks at stack[i].
- play: the play method takes two integer parameters that represent the source tower and the destination tower. This method will use the peek stack method to see if valid moves are to be conducted (size condition and non-null source). Invalid moves are not counted in the moves instance variable. If a valid move is detected, pop and push should be used appropriately!

checkWinner will be called from this method.

- checkWinner: will see if all disks are placed at the third disk or not. If so, the game state is changed to WINNER. Otherwise, if the max number of moves are consumed, the state is changed to LOSER.

The player can continue playing after winning or losing and the level can keep couting the moves, but the state will never change! Winning flage is UP!

Note: HanoiTower class logic does not allow this as two players are playing, but this is possible if a single player is playing!

- toString: should print the board of the game (see sample test cases). Example for 5 disks (4 at tower 1, 1 at tower 2, none at tower 3 (note empty lines!):

```
Tower 1

--
---
----
-----
Tower 2




-
Tower 3
```

- StudentInof.java and Utils.java are the same as in the previous assignment.

How to test/check my implementation:

Please note that there are two ways to check the correctness of your code:

- Run HanoiTower with the number of disks and manually play the game between two players. You can try the three sample runs stated in this document.
- Run the JUnit test files will test your implementation at HanoiTowerGame class. Instructions and the test cases for this run are available at the class HanoiTowerGameTest.

Academic Integrity

This part of the assignment is meant to raise awareness concerning plagiarism andacademic integrity. Please read the following documents.

- https://www.uottawa.ca/vice-president-academic/academic-integrity
- https://www2.uottawa.ca/about-us/policies-regulations/academic-regulation-i-14-academic-fraud

Cases of plagiarism will be dealt with according to the university regulations. **By submitting this assignment, you acknowledge:**

1. I have read the academic regulations regarding academic fraud.
2. I understand the consequences of plagiarism.
3. With the exception of the source code provided by the instructors for this course, all the source code that will be submitted is mine.
4. I did not collaborate with any other person, with the exception of my partner in the case of teamwork.

• If you did collaborate with others or obtained source code from the Web, please list the names of your collaborators or the source of the information and the nature of the collaboration. Put this information in the submitted README.txt file. Marks will be deducted proportionally to the level of help provided (from 0 to 100%).

Rules and regulation

- Submit your assignment through the online submission system at brightspace
- You must preferably do the assignment in teams of two, but you can also do the assignment individually.
- If you do not follow the instructions, your program will make the automatedtests fail, and consequently, your assignment will not be graded.
- We will be using an automated tool to compare all the assignments against each other (this includes both the French and English sections). Submissions that are flagged by this tool will receive a grade of 0.
- It is your responsibility to make sure that BrightSpace has received your assignment. **Late submissions will not be graded.**

Files

Place all the assignment files (see below) in a folder and name it as

a3_3000000_3000001

where 3000000 and 3000001 are the student numbers of the team members submitting the assignment (***simply repeat the same number if your team has one member***).

Zip this folder using any zipping tool; the resulted file should be a3_3000000_3000001.zip

Your submission must contain thefollowing files.

- Stack.java
- LinkedStack.java
- Player.java
- HumanPlayer.java
- GameState.java
- HanoiTower.java
- HanoiTowerGame.java
- StudentInfo.java
- Utils.java
- README.txt
  - A text file that contains the names of the two partners for the assignments, their student ids, section, anda short description of the assignment (one or two lines).

References:
[1] https://en.wikipedia.org/wiki/Tower_of_Hanoi

[2] https://www.mathsisfun.com/games/towerofhanoi.html