Final Project

Hanoi Tower AI Game (40%)

DUE DATE

Session 15

OBJECTIVE

- Interpret specifications and analysis performed
- Design a solution based on the requirements and specifications
- Design the logic required for functional solution
- Translate design documents and algorithms into source code
- Use debugging tools and error-handling techniques
- Validate the solution with test data
- Integrate the knowledge acquired thus far
- Use the features of Eclipse IDE
- Demonstrate the use of the Java programming language
- Apply the various program flow constructs
- Apply advanced Java programming techniques
- Implement graphic objects in the GUI
- Employ JavaFX GUI to implement a game
- Use advanced Java concepts in designing a graphical game

DESCRIPTION

Your knowledge on Java, Algorithms, and DB are good enough to create a game with GUI, Login, and a simple AI. This game will simulate a Hanoi Tower explained in "18.8: Towers of Hanoi" in Chapter 18: "Recursion" in the *Java How to Program, Early Objects* (11th Ed.) textbook.

This game involves moving a tower of disks on one peg to another without putting the bigger disks of the tower on top of the smaller disks. When the user logs in, they can configure the game. (The number of pegs, blocks, etc.)

After configuring the game, the user should be able to see the visual representation of the tower. This can be implemented using JavaFX. Once the user starts the game (this can happen from the "Start" button), the timer

goes off until the user finishes the game. The user should be able to interact with the GUI to move the disks around and finish the game.

Once the user finishes the game, the application stores the record of the game with the username and time so that anyone accessing the application can see the records.

The user can give up on the game and see the solution. AI with the algorithm to solve Hanoi Tower will show the animation of how the game could have been finished. The user will also need to be able to save and load the game in progress.

INSTRUCTIONS

This project is all about architecture and functionality. Plan the architecture in the beginning to avoid modifications at the end. It is important to fully understand the rules of the *Tower of Hanoi* game; make sure to fully understand the information in "18.8: Towers of Hanoi." It is good to watch video tutorials of this game to learn how it can be solved programmatically.

This application requires you to combine different concepts and technologies covered in the course.

- When the application starts, a JavaFX window will pop up and prompt the user to either log in or create an account.
- If the user wants to create an account, the application needs to make sure all the usernames in the DB are unique. This has to be done by connecting to JDBC.
- The application checks the credentials by matching the usernames and the passwords that are stored in the JDBC database. Once logged in, the user can choose to either start the game or load the saved game from the file system.
- If the user chooses to start the game, they can specify the number of pegs/disks, which pegs the disks have to be moved between, etc. Minimum three pegs and three disks; maximum limit is up to the programmer.
- The application needs to provide an easy UI involving mouse clicks and entering the number. For example, they should be able to move the disks from one column to the other by mouse click or drag.
- The game follows the rules of *Tower of Hanoi*. For example, a disk cannot be put on top of a disk that is smaller than itself, etc.
- The application should be able to determine when the game is finished (Solved). When the game is finished, it shows the time record and asks the user whether to save the record or not. Then it prompts the user to configure another game of *Tower of Hanoi*.
- The timer keeps track of the time from the beginning of the game until the user finishes or gives up. The timer should display the lively updated time to the user. Once the user finishes the game, the application stores the record in the JDBC database.
- Any user can view the game record of any other user. The application can show the "leaderboard," which displays the top player in each game configuration.
- The user can save the game without finishing it; thus, the application needs to be able to serialize the

object that stores info of the on-going game and deserialize it if the user wants to load the game and continue. **Note: objects containing JavaFX objects cannot be serialized.**

- If the user wants to give up on the game, AI on the application shows the animated solution to the game. Refer to "18.8: Towers of Hanoi" for how to make the application to solve the Tower of Hanoi game.

Key Functions

The main functions of the simulator may be summarized as follows:

- Login credentials involving JDBC
- User score record involving JDBC
- Save/Load the on-going game using object serialization
- Game GUI with JavaFX
- Algorithm to solve any Tower of Hanoi game
- Animation to show how the algorithm solved the game.

Guideline

The following is a guideline to implement the Tower of Hanoi Game. Fully understand the game rules and requirements before you start coding. Make sure to plan thoroughly according to the requirements.

- It is better to separate both GUI and AI algorithms from the logic side.
- It is better to integrate all DB queries within a single class.
- It is better to implement AI algorithms in separate classes.
- Don't worry about making the GUI pretty until the logic side is complete.

DBHelper Class

This is the class to connect to the JDBC database, PUT data on the database, and GET data on the database.

Key methods:

checkUserCredential:

Take the username and password as strings, check if the user exists, and check if the password is correct. Passwords are good to be hashed for security reasons, but it is not required to implement the security side of the application for this project.

createNewUser:

Take the username and password as string to create a new user in the database.

DisplayLeaderboard:

Sort the players by shortest time to finish the game and send the list of sorted records.

DisplayRecordOfUser:

Get records of a certain user in the DB.

SaveGame Class

This object gets serialized to save the on-going game. It can be stored inside the Java project folder. The application needs to know if the saved game is opened by the right user. It needs to stop one user from opening the saved game of another user.

GUI Class

This class is specialized for displaying animation and taking user input for the game. It is a good idea to separate game logic from GUI.

AI Class

As mentioned in the requirements, the application needs to be able to solve the game by itself and display the process in animation. Solving the *Tower of Hanoi* game involves an algorithm covered in the course. It is a good idea to create a separate class to solve the game.

GameLogic Class

Game logic can be implemented in many different ways. Here are the suggestions:

- The peg object gets populated with ID in the game logic class. Each disk object stores the ID of the peg they belong to. Whenever the user moves the disk, the event handler accesses the GameLogic class to set the position value inside the certain disk object.
- GameLogic class stores codified information on which disks are in which pegs. This data changes every time there is an input from the user.
- GameLogics can be integrated into the GUI event handlers (many students choose this way and regret it).

Note

This guideline is not a requirement. Unlike the beginner's OOP course where the guideline is the requirement, students can choose to implement the application however they want. However, it is a good idea to implement the classes mentioned above to avoid any extra debugging work.

Part 1 – The Architecture Design

The key part of this course is to understand Object-Oriented Design and other Java technologies. After reading the specification and requirements, draw a class diagram to fit the requirements. It will work as a blueprint for the project. Write a pseudo code to fit the class diagram. You are free to write pseudo code for certain methods, but this is not required.

Part 2 – Interpret the Design to Write Source Code

Refer to your analysis and design work from Part 1 and create the classes accordingly, wiring them as designed. Your code must show the same flow as that of which you designed in Part 1. For example, if you design an

architecture to use a while loop to cycle through an array to extract a specific data element, your code should include the while loop as well with the same logic.

Error Management: Test the Client Prototype application

1. You should test as you build to avoid unnecessary delay.
2. Make sure you add appropriate data input validation and error-handling (**hint**: every time you capture a piece of data, make sure it is valid and matches the type of data expected).

Schedule

Note, this schedule is a guideline only. You may require more or less time during each session.

Session 11: Part 1 Analysis and Design of the work to be done

- Analyze the project specifications and the feature requirements.
- Create your application design. You must use at least two types of documents for the design component. (Suggestion: You may want to use a UML class diagram to design your program's overall architecture, and then use flow chart for logics inside the methods and/or pseudo code for each individual feature, depending how you decide to create the program).
- Validate your logic and architecture with your instructor. Make sure you are able to explain the logic to your instructor and that he or she approves it before moving on to the coding phase.
- Typically, you should be able to complete this phase in one session. Otherwise, you should complete it as homework and show it to the instructor at the beginning of the next session.

Session 12: Logic Required by the Processes

- Upon approval of your initial design, you can begin detailing each process. At this point, each process should have sufficient detail to match the requirements. You can review the details of each and determine if there are any elements that you need to address based on your instructor feedback.
- Once everything is finalized, you may begin coding your program. Create your project structure in Visual Studio and ensure that all of your required files are properly setup.
- Begin coding your application according to your logic documentation. It is essential that you follow your plan.

Sessions 13 to 15: Apply Analysis and Design to Coding the Application

- Apply your problem-solving skills.
- Code the various processes of your program according to specs.
- Add appropriate data input validation and error-handling.
- When coding your application, you may need to revise the logic (iterative development). Make sure you update your design documentation as required.

- Document any changes to your logic and design explaining why you decided to modify your approach. This must be submitted as part of your project submission
- Validate the solution using test cases.
- Add any missing input validation or error-handling.
- Update the documentation.
- Submit the project.

SUBMISSION INSTRUCTIONS

Your project must include the following:

- All source code, as well as all other files required for the proper functioning of your application.
- A project report containing:
  - A title page with your name, the submission date, and your instructor's name
  - A table of contents
  - The project specifications
  - All relevant documentation including diagrams and pseudo codes
  - The user manual (instruction guide for your instructor; this must include any test data that you use to test your application)
  - Explanations of any additions that you made (where applicable)

It is strongly recommended that you verify, using anti-virus software, that your submission contains no viruses.

Work must be submitted in the correct file type and be properly labelled as per the College naming convention: NAME_COURSE_ASSIGNMENT. E.g. XuXiaLing_FM50D_A01.

GRADING CRITERIA

Assignment Value: **40%**

Grading Criteria	Grading
Analysis and Design Part 1 with complete documentation showing appropriate architecture design required by the processes/methods (pseudo code and class diagrams) approved by instructor	/30
Design requirements specified in the Detailed Guideline/Requirements	/25
Functional requirements specified in the Detailed Guideline/Requirements	/20

Documentation preparation and update (if you modify your architecture or logic, it must be documented)	/5
Input validation and error handling using controls and code	/15
Error-free and bug-free application (properly debugged)	/5
TOTAL	**/100**

Penalties

- A penalty of 5% will be deducted from your mark for each day your project is late.
- Any project submitted more than three calendar days late will receive a maximum mark of 60%.
- Any project containing a virus must be resubmitted and will receive a maximum mark of 60%.