Task 6:
Progress Report

GROUP 57:
Jun, Akansha, Chris, Koji, Iris, Wei

Specification

Jun

CRC Cards

Iris and Koji

Entity Classes

- User
- Board (abstract/interface)
- TileBoard
- Leaderboard
- gameSettings

Use Cases

- UserCreator, UserManager, HistoryUpdater
- promptLogIn, InfoChecker
- BoardGenerator, BoardUpdater
- LeaderboardManager
- gameSettingsManager

Controllers

- endGame

Database

- UserDatabase

Command Line/UI

- UserInputManager
- runGame

User	
username: username of User password: password of User gameMode: difficulty of User matchHistory: collection of each game this User has played (Mode, numMoves, time) getUsername: gets username of user setPassword: if user wants to change password getGameMode: current game mode user is in setGameMode: changes difficulty of gamemode	UserDatabase UserAccount

Entity –> Enterprise Business Rules

UserManager	
changeGameMode: changes the current game mode of User changePassword: changes password of User (requires getPassword and setPassword from User)	User

Use Case–> Application Business Rules

UserCreator	
createUser : creates a new User object with username, password, empty matchHistory for each of the modes checkUsername : helper for createUser that checks if username exists in userDataBase	User promptLogin UserDataBase

Use Case–> Application Business Rules

promptLogin	
User login (userLogin) User sign up (createNewUser)	UserCreator userLogin

Use Case–> Application Business Rules

InfoChecker	
password entered valid/invalid (from userDatabase) username exists/doesn’t exists	UserDatabase UserLogin

Use Case–> Application Business Rules

UserLogin	
Prompts login/signup Takes user input for username and password and calls infoChecker methods to see if information is valid	InfoChecker promptLogin

Command Line/User Interface –> Frameworks and Drivers

UserDatabase	
Stores a collection of Users and user data in a local file	User createNewUser

Database –> Frameworks and Drivers

HistoryUpdater	
updateHistory: adds the last match with data (gamemode, moves, time) played to User	User endGame

Use Case–> Application Business Rules

UserInputManager	
getInput : gets the user's input for moves	runGame

Command Line/User Interface –> Frameworks and Drivers

Board (abstract)	
Shape: number of rows and column Size: rows x columns (number of tiles that are put on) numPairs: size/2	BoardGenerator TileBoard

Entity –> Enterprise Business Rules

Tile	
key : integer value getKey : gets key value of Tile Theme : theme of tile	TileBoard

Entity –> Enterprise Business Rules

TileBoard Board	
revealedTiles: keeps track of which Tiles are always flipped because they are matched boardState: revealed and unrevealed cards for the current state (even during a players move) numMoves: the number of moves for this board (2 cards flipped) getBoardState Time: time for this board game	Tile Board runGame User BoardGenerator

Entity –> Enterprise Business Rules

BoardGenerator	
Creates TileBoard with size based on User.GameMode, (num moves, time) = 0, 0	User TileBoard

Use Case–> Application Business Rules

BoardUpdater	
updateBoardState: updates board based on user inputs (eg. keeps matched cards flipped all the time) updateNumMoves: updates number of moves for this board after user's second input per turn timeUpdater: constantly running to keep track of time	getUserInput runGame TileBoard

Controller –> Interface Adapters

runGame	
generateBoard : create board based on user's input difficulty (randomly place tiles) getUserInput: constantly await for user input until all tiles are matched showTile : userinput reveals tile on TileBoard checkMatch : helper for changeGameSettings, check if 2 consecutive tiles are a match changeGameSettings: changes settings updateBoard: updates Board whenever User moves, calls checkMatch to see if second move matches first move endGame: i	gameSettingsManager BoardGenerator BoardUpdater UserInputManager TileBoard

UI

Leaderboard	
highScores: ordered list for the highest scores for each game mode getLeaderboard: Contains/displays top 10 scores and username out of all users for a given difficulty setLeaderboard: changes list of high scores	LeaderboardManager

Entity –> Enterprise Business Rules

LeaderboardManager	
addToLeaderboard: for a gamemode, adds (User, gamemode, num moves, time) if it has higher num moves than one on the leaderboard. If num moves is tied with another, check time showLeaderboard: shows leaderboard	Leaderboard endGame

Use Case –> Application Business Rules

endGame	
checkLeaderboard promptNewGame: asks user if they want to play another game Add in match history (gamemode, num moves, time) to User’s match history	BoardGenerator historyUpdater LeaderBoardManager

Controller –> Interface Adapters

gameSettings	
isOpen isMuted Volume setVolume wantsToExit	gameSettingsManager

Entity –> Enterprise Business Rules

gameSettingsManager	
Mutes the sound effects Can adjust volume Resets game for user Exits game for user	gameSettings runGame

Use Case–> Application Business Rules

Scenario

Akansha and Chris

Scenario

1. User is prompted to log in to their account
2. User logs into their account with username and password
3. A 3x4 board (easy mode) is generated
4. 12 tiles are generated, each with a numerical key. Each tile in a pair should have the same key as one other tile in the pair
5. The tile order is randomized and sent into the board method, where it is displayed in the 3x4 board.

Scenario

6. When the player clicks on the first tile, the timer begins to count the time it takes the player to complete the game.
7. The user clicks on a tile, the front side with the picture is displayed.
8. The user clicks on another tile, and the front side is displayed. Move counter increases by 1.
9. After the second click, check if the two tiles the user clicks on are the same.
    a. If they are the same, increase the score counter by 1.
    b. If they are not the same, hide both tiles again.

Scenario

10. Repeat step 7-9 until all the tiles are matched/all the tiles are showing.
11. Update user's game history with the game they just played (total moves from moveCounter, time). Update leaderboard if the score is a-top the leaderboard.
12. User is sent to view the leaderboard
13. The user can choose whether to start a new game or exit

Skeleton Code

Jun and Akansha

Open Questions

Chris

Open Questions

- What are some interfaces that we could implement if we need to at all?
- How do we deal with unexpected user inputs or activity? What would be the best way to make sure that the program doesn't break?

Design

wei

What has worked well so far with your design?

- Tile Class
- User Class
- Numerical Prototype of the game

Individual

Akansha

- Choosing domain (*Task 1*)
- Editing specification (*Task 2*)
- Making and editing CRC cards (*Task 3*)
- Writing scenario walkthrough (*Task 4*)
- Made and set up the repository (*Task 5*)
- Created/designed all classes and test classes for skeleton program
- Worked on Tile class for skeleton program (constructor, get, set, createTileList, setUpDashBoard, setUpKeyBoard, printBoard, runGame, main, login, setUpBoard, move counter and leaderboard) (*Task 5*)
- Wrote JUnit test cases for Tile class (*Task 5*)
- Progress report (*Task 6*)
- PLAN TO WORK ON: User class (accounts, database), Tile

Jun

- Choosing domain (*Task 1*)
- Writing specification *(Task 2)*
- Making CRC cards *(Task 3)*
- Editing scenario walk-through *(Task 4)*
- Progress report *(Task 6)*
- Made and set up the repository *(Task 5)*
- Created/designed all classes and test classes for skeleton program *(Task 5)*
- Worked on Tile class for skeleton program (constructor, get, set, createTileList, flipped, setUpDashBoard, setUpKeyBoard, printBoard, runGame, main, login, setUpBoard, move counter and leaderboard) *(Task 5)*
- Wrote JUnit test cases for Tile class *(Task 5)*
- **PLAN TO WORK ON:** Leaderboard, LogInPage, TileBoard

Chris

- Making CRC cards
- Making scenario walk-through
- Worked on Tile class for scenario walkthrough (login, runGame)
- Open questions

Koji

- Task 1
  - Choosing domain
- Task 2
  - Writing specification
- Task 3
  - Developed CRC Card model final drafts
- Task 5
  - Test Cases for Skeleton code
- PLAN TO WORK ON: Tile, TileBoard, BoardGenerator, BoardUpdater

Iris

- Task 1
  - Picked the domain for our project
- Task 3
  - Combined the 2 CRC models we had, revised, and finished it up
- Task 5
  - Worked on Tile class for skeleton program (fixed print method, moved login method outside of rungame, moved setUpBoard outside of rungame, added move counter, added leaderboard message, fixed main method)
  - Wrote JUnit test cases for Tile class
- Task 6
  - Presentation for the CRC model and wrote down individual parts
- **PLAN TO WORK ON** : LeaderBoard, LogInPage, User

Wei

Task 1

- Choosing Domain

Task 6

- Design