Report of Skyjo Enhanced: UTBM Edition

By ALLAINÉ Hugo, ANGONNET Léo, BALEZEAU Quentin

Summary

Conception

The project

The concept of the project is to make a Skyjo card game adapted to UTBM's world.

The main functionalities and features of the game.

After launching the program, we can choose to make a new game, check the rules, or exit the application. When we start a new game, we have the choice between 3 different modes: DEUTEC, Engineering degree, UTBM Enjoyer. The first player who reaches 120, 180 or 300 points loses depending on game mode. Then we choose the number of players, between 2 and 6 and their respective names. If everything is correct, the game starts.

Our design decisions

For our game, we have chosen to base the point system on the ECTS. Cards give you credits according to the letter. The A card gives you 10 credits, the B card 8 credits, the C card 6 credits, the D card 4 credits, the E card 2 credits, the FX card takes away 1 credit and the F card 2 credits.

The first round, each player reveals 2 of his cards. The player who has the most credits by summing his 2 revealed cards starts.

Each turn, each player can choose to draw from the deck or the discard pile. In the first case the player can look at the card and either keep it or discard it. In the second case, the player can only trade the card.

When a player reaches 120, 180 or 300 credits, the game stops and the player with the least points wins the game.

Implementations choices

- The maximum number of players is limited to six, as having more would make the interface unclear.

- Customized cards are used.

- The total number of cards is reduced to 115 instead of 150.

- In cases where two players have the same score after the selection phase, the last player in both will be the first in the order of play.

- Instead of a dialog box, we use a "spinner" object, because the "spinner" is much more convenient to manage an integer input

- The card list is designed as an abstract class. It allows to create classes Deck Discard and Draw that extend CardList.

Architectural patterns of classes

- Public class Card
- Public Abstract Class CardList
- Public class CardMouseListener implements MouseListener
- Public class ConfigLoader
- Public class Deck extends CardList
- Public class Discard extends CardList
- Public class Draw extends CardList
- Public class Game
- Public class GameMenu extends Window
- Public enum GameModeEnum
- Public class Main
- Public MyButton extends JButton
- Public class Player
- Public class Window

Externals libraries used

In our card game project, we utilized the Swing and AWT libraries to handle the interface components. We employed JFrame, JPanel, Spinner, MouseListener, and ActionListener to create and manage various elements of the user interface. These libraries were chosen for their simplicity to use.

Additionally, we made use of the util library, which provided us with valuable tools. One notable feature was the Resource Bundle, enabling us to load parameters from properties files. This allowed for greater flexibility in customizing aspects of the game settings. We also leveraged Array and ArrayList data structures to store and manipulate cards, as well as JPanels to organize and display game elements. Furthermore, the File library was employed to enable the opening of the game rules in PDF format, enhancing the user's understanding and providing a convenient reference.

To ensure randomness and variation in the gameplay, we utilized the Collection library to shuffle the card list. This allowed for fair distribution and unpredictable card selections.

Our limitations or trade-offs that we encountered during the implementation and how we addressed them

During the implementation of the Skyjo game in Java, limitations and trade-offs were encountered, which required careful consideration and resolution. The following are the limitations or trade-offs that were faced, and the approaches taken to address them:

UI Component Alignment:

Limitation/Trade-off: Difficulty in correctly aligning the different UI components.

Solution: Extensive testing and adjustments were conducted to ensure proper alignment and visual coherence. The use of layout managers provided flexibility in managing the placement of UI elements.

We did not succeed in setting a background for the game.

Code Structure and Function Placement:

Limitation/Trade-off: Confusion and challenges in structuring the code and determining the placement of functions.

Solution: Refactoring and modularization were performed to improve the code structure and functions were grouped logically. We use abstract to reduce duplicated code and to enhance code organization and readability.