Mişti

SE 116 Project Description

Introduction

This project will be conducted with teams of size three. Students are free to form their own teams or let the instructors randomly assign teammates.

The Game

Mişti is the new version of the card game Pişti. It is played by 2 to 4 players. Read this description carefully, it is written to answer all of your questions.

A game card deck has four suits:

- Spades: ♠
- Clubs: ♣
- Hearts: ♥
- Diamonds: ♦

Each suit has 13 cards: A (Ace), 2, 3, 4, 5, 6, 7, 8, 9, T (Ten), J (Jack), Q (Queen), K (King). Therefore the deck has 52 cards.

Each card is worth certain number of points, as recorded in a multi-line text file with the following format for each line:

<suit><cardface> <points>

Suit can be one of S, C, H, D, or *, signifying Spades, Clubs, Hearts, Diamonds, or any suit respectively. Cardface can be one of A, 2,3,..,9, T, J, Q, K, or *. Here the star matches any face value.

Points are integers; negative and zero values are allowed. In order to determine the value of a card, the first matching line is used. If no lines match the card, the default point value is one point. For example, if the first line says S* 5 and the second line says *3 -2, the points for Spade 3 is 5, as indicated in the earlier line, but the points for Diamond 3 is -2, the first match occurring in the second line, whereas Heart 4 gets the default value 1, due to no match.

In the beginning the computer must read (and validate of course) from the command line the name of the points file, the number of players, number of rounds to play, name of each player and whether the player is a human, a novice bot, a regular bot, or an expert bot. There can be at most one human player. Bots are automated, whereas a human player must interact through the console, viewing their hand of cards, what each player plays, and typing to the console their choice card when it is their turn.

Note that the computer acts as the dealer - at the beginning of each round the dealer shuffles the deck and cuts the cards. Cutting cards is done by randomly splitting the deck into two parts, and placing the bottom part on the top part. Now that the cards are ready, the dealer places four cards on top of each other on the board. Next the dealer starts dealing a hand: one card to each player until each has four cards.

The game is played by taking turns. Each player, when it is their turn, places one of their cards on top of the board. While doing so, they are allowed to take all the cards from the board if their card and the top card have the same value and thus earn the summation of their respective points per the pointValue file. For example, if the top card is ♣7 and the player has a 7 from any suit, then the player can play the 7, and take all the cards on the board. The "Jack" of any suit has a special power. This card can take all the cards from the board regardless of the top card value. When players have played all their cards, the dealer deals another hand. If there are cards left on the board after all the hands are played, the last player who got the cards from the board also gets the remaining ones.

The name "Mişti" comes from the case when a player takes a single card from the board. For example, when there is a single card, ♥4 on the board, and the next player whose turn it is has a ♣4, then the player can make a "Mişti". In that case the score for these cards are multiplied by five. For example, if ♣4 is worth 2 points and ♥4 is worth 3 points, the player earns (2+3)*5 = 25 points.

A Sample Game

The cards are shuffled and then cut. The table shows the board by placing the top card on the first line, and the remaining cards on the second line. We can never see the opponent cards.

Turn	**My Hand**	**Board**	**Comment**
1	♠10 ♣A ♥3 ♦3	♣4 ♣3 ♣2 ♥A	Since I don't have a four, I play one of the 3s.
	♠10 ♣A ♦3	♥3 ♣4 ♣3 ♣2 ♥A	Now the opponent will play. Hint: Since there are four 3s in the deck, it is very likely that the opponent will be playing a three.
	♠10 ♣A ♦3	♣10 ♥3 ♣4 ♣3 ♣2 ♥A	Now that the player has played, and cannot take any cards, it is my turn again. I'll be playing my 10, and I will get all the cards. One of the cards is ♣2 - so I'm happy.
2	♣A ♦3	♠10 ♣10 ♥3 ♣4 ♣3 ♣2 ♥A	Since I was able to play a 10, I'll get all the cards. The board will be empty.
	♣A ♦3		Now it is time for the opponent to play.
	♣A ♦3	♦A	It must be my lucky day. An ace is played, and

			now I can make a “Mişti”
3	♦3	♣A ♦A	Mişti! The board is once again empty.
	♦3		Now it is time for the opponent to play.
	♦3	♠7	Nothing to do, I will play my only remaining card.
4		♦3 ♠7	Card played, it is the opponent’s turn.
		♠8 ♦3 ♠7	Now both players are out of cards, the dealer will give us new cards.
5	♠5 ♣Q ♦J ♥3	♠8 ♦3 ♠7	I can play the J and take all the cards…
			… this goes on until all cards are played.

Once the game round is over, the player with the highest score wins.

Version Control System: Git

We want you to learn the common “version control system” software that is used everywhere: Git. A version control system helps us to keep track of our changes in time. It is also used as a collaboration tool, which means it also helps you to develop software with other developers. Git is one of many version control systems, but currently the most popular one. Therefore, it is vital for you to learn what Git is, what it does, how it works, and how to use it properly. This project is a great starting point because you will be using Git on your own, without the complication of other users, and you will get to learn the basics at your own pace.

Git software is available at https://git-scm.com/ and it contains documentation, along with videos, at https://git-scm.com/doc or at https://skills.github.com/. However, the documentation also talks about some more advanced things you can do with it. We have already uploaded an offline video to Panopto about Git. Please watch it and start using Git in your project from the very first day.

While Git can be used on your own local system, it is mostly used with a remote (online) server. The most common one is the GitHub, at https://github.com/. Registration is free, so make sure you create a profile if you haven’t done so already. You can create a repository for your project, and keep your source code online. Make sure you keep your Mişti project private, so that no one else can access it.

Requirements

In any software project, we write down a list of requirements. These requirements have to be implemented because it is what the software should do in a given environment. Some of them are “functional” - which are about the functions the software should have, and some of them are “non-functional” - which are about the conditions that the software will be running. You are expected to implement all of them.

Non-functional Requirement 1: The program must be implemented in the Java programming language. Source code should be delivered as a zip file, no other format is permitted.

Rationale: Since we are learning programming using Java, this is expected.

Non-functional Requirement 2: The development must be done on a private Git repository.

Rationale: Version control is important, especially in a team project. While you are free to use any Git server, we strongly suggest that you create a GitHub account. Every time you improve your project, commit your changes to the repository, so that we can see your progress at the end of the semester. Install the “GitHub Desktop” program and you will be ready to go. **The repository must be private**. No one must be able to see it, but your team. We expect at least weekly commits from each team member. Each commit must include the description of the work included in the commit. **The deliverables must include your git log screenshots.**

Non-functional Requirement 3: The program must be delivered as a JAR file.

Rationale: Instead of several class files, pack the program into a JAR file, and submit it as a Jar file.

Non-functional Requirement 4: The program must be accompanied by a project report in pdf format, detailed later in this document..

Rationale: The design and implementation decisions must be justified, and operation of the code must be demonstrated.

Non-functional Requirement 5: Each team member must be able to demo the program execution, and answer questions about the design, implementation, or the code to demonstrate their participation in the project.

Rationale: This will be a big factor in grading. Being absent during the presentation session is not acceptable.

Functional Requirement 1: The program must be able to create a deck of cards.

Rationale: This is a basic requirement: without the cards the game would be unplayable.

Functional Requirement 2: The program must be able to shuffle and cut the deck.

Rationale: You must use a shuffle method from the JDK in order to demonstrate your mastery of collection hierarchy, do not implement your own collection classes needlessly.

Functional Requirement 3: The program must be able to read game parameters from the command line and display appropriate error messages for invalid input. Parameters include number of players, point file name, name and expertise level of each player, and verboseness level.

Rationale: This is a console application, so the configuration is via command line parameters (arguments).

Functional Requirement 4: The program must be able to move cards from the deck to the players and the boards.

Rationale: There must be only one copy of a card - this requires careful planning: how will the cards be moved from the deck into each player's hand, or to the board, or when the player takes all the cards, where will the cards go? What about cards that were made "Mişti"?

Functional Requirement 5: The program must print to console a log of hands dealt to each player, and all the moves for each round and each hand in verbose mode, and just the score for each round in succinct mode. Below is a sample log fragment in verbose mode

….

Hand 4: Player1: {H2, D3, D4, D5} Score 12; Player2: {S2, S8, S9, DK} Score 13

1. D3 S2
2. H2! S8
3. D4 S9
4. D5 DK

Hand 5: …

Functional Requirement 5: The program must be able to calculate each player's score.

Rationale: This requires the software to keep track of the cards each player has taken from the board, including the Mişti cards. It must also be able to read the point value file and determine the value of each card accordingly.

Functional Requirement 6: The program must be able to store a "high score list" on a file which stores the top 10 scores and the names, expertise levels of the players who scored them.

Rationale: The high score list must be updated if one of the players gets a score higher than those on the list. Once there are more than 10, the last ones should be removed and only 10 must stay in the list.

Functional Requirement 7: The program must include a novice player. This player should obey all the rules of the game, but have a very rudimentary strategy, and just randomly choose one of the available cards to play at their turns.

Functional Requirement 8: The program must include a regular player. This player should obey all the rules of the game, but also choose a good card to play, based on which card is on the board, and what cards are in the hand, taking into consideration the point value of each card. For example, consider when the board has a 3 on the top, and the next player also has a 3: In this situation, the next player must play the 3 if the total value of cards to be taken is positive, otherwise a different card must be chosen.

Functional Requirement 9: The program must include an expert player. This player should obey all the rules of the game, but also choose a good card to play, based on what cards are on the board, and what cards are in their hand, and tracking what cards have already been played, taking into consideration the point value of each card.

Rationale: The program must be able to support games with players of varying expertise, as specified in the command line parameters, and choose the appropriate card for each such player. No restrictions on the mixture of levels; all or some can be from any level. Make sure

you discuss player game strategy in your report, and what aspects of players you are able to reuse, using which OO techniques across different expertise levels.

Functional Requirement 10: The program must include a human player.

Rational: At most one human player is allowed in any game round; i.e., zero or one human player is allowed but not more. The human user must view the game and indicate their card choice over the console.

Report

A report should accompany your source code. The report must be written formally. In this report, discuss how you have implemented all the functional requirements, discuss your design choices, the challenges you have faced and how you have solved them. **It should also include a sample run of your program for one round with 3 bot players, one expert, one regular, and one novice.**

This can only be done if you keep taking notes during the development. When we read the report, we must understand how you have developed the program. So, make sure you do your best to talk about what you have done in detail. The more detailed it is, the better.

The report must be in PDF format. If you submit in another format, such as “docx” or “pages” or anything else, we will not accept it, and you will not get any points.

Grading

All functional requirements are 60 points.

Report is 20 points. Evaluation criteria includes the narration quality, and the software design choices.

Source code is 15 points. Evaluation criteria includes adherence to basic style guidelines, proper comments, and implementation choices.

Git logs will be examined, it is 5 points. This is a separate PDF file that contains screenshots. However, we can ask you to show us your Git repository during the oral exam, or we can email you earlier to give access to us as a contributor.

There will be an oral exam at the end of the semester.

Exceptions:

- Failing any nonfunctional requirement results in a grade of zero.
- If the program is not running, then you cannot get any points.
- If the program crashes on **any** input, you will lose a large amount of points. Test your code.
- If you cannot sufficiently answer the questions about any aspect of the project (code, design, report, etc.) during the oral exam, you will lose substantial points overall.

Submitting Your Project

This project is a team project. One member from each team must submit the following files

- A ZIP file that contains your source code.

- A JAR file that runs your program.
- A PDF file that contains your project report.
- A PDF file that contains screenshots of your Git commits.

The due date is May 14, 23:59. Late submissions will NOT be accepted.