Pişti

SE 115 Project Description

Introduction

Pişti is a card game played by 2 or 4 people. It is very common in Turkey and one of the easiest games to learn and play. Read this description carefully, it is written to answer all of your questions.

This project is about the Pişti game played by **two players**: the computer versus the human player.

The Game

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, 10, J (Jack), Q (Queen), K (King). Therefore the deck has 52 cards.

In the game one of the players is the dealer - the dealer shuffles the cards and asks the other player to cut them. Cutting cards is done by splitting the deck into two parts, and placing the bottom part on the top part.

Now that the cards are ready, the dealer starts dealing the cards: one to the player, one to self, until each has four cards. Then, the dealer places four cards on the board. These cards are stacked - that is, they are not side by side, but placed on top of each other.

The game is played by taking turns. The player starts the game and is expected to place one of his cards on top of the board. While doing so, he is allowed to take all the cards from the board if the cards have the same value. 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.

The name “pişti” comes from a case where there is a single card on the board, and one of the players has the same card. For example, there is a single card, ♥4 on the board, and one of the players has a four from any suit, then the player can make a “pişti” and score 10 points.

Talking about “points,” two cards have additional points: ♦10 is 3 points and ♣2 is 2 points. All other cards are 1 point each. Once all cards are played, the player who has more cards gets an additional 3 points.

The “Jack” of any suit has a special power. This card can take all the cards from the board without taking the value of the top card into account.

If there are cards left on the board once all the cards are played, the last player who got the cards from the board also gets the remaining ones.

Basically, the game is to get all the cards on the board either by using a Jack or by using a card that has the same value as the top card on the board.

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 "pişti"
3	♦3	♣A ♦A	Pişti! 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	Now both players are out

		♦3 ♠7	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 is over, the values of the cards are calculated, the player who has more cards receives an additional 3 points. Each Pişti is 10 points (the value of the card does not matter). 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. 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 Pişti 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.

Rationale: Since we are learning programming using Java, this is expected. You are expected to use Java, but refrain from using additional libraries, such as GUI. You will not get any additional points if the project is in 3D.

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

Rationale: 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 you.

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.

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. The suits are available in Unicode, so you can easily copy and paste the following symbols ♠, ♣, ♥, ♦ and use them as characters.

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

Rationale: Although this is only required at the very beginning it is a very good challenge for SE 115. You must **not** use a shuffle method from the JDK, but rather **implement your own**. Find a shuffling algorithm, learn how it works, and implement it.

Functional Requirement 3: The program must be able to cut the deck.

Rationale: Once again, you should come up with how the program will cut the deck, and how you will realize it.

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 the player's hand, or the board, or when the player takes all the cards, where will the cards go? What about cards that were made "Pişti"?

Functional Requirement 5: The program must be able to calculate the player score.

Rationale: This requires the software to keep track of the cards each player has taken from the board, including the Pişti cards.

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

Rationale: You will learn how to read and write to a file in the upcoming weeks. 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. For example, once the game finishes the player collects 110 points. Since there is no one else in the list, then the game will ask the player for their name, and store this score in a file. The next game, if the player gets a score of 120, the game should again ask for their name and store the 120 as number 1, 110 as number 2, with their respective player names.

Functional Requirement 7: The program must be able to play for the computer.

Rationale: The program must be able to choose the appropriate card for the computer. For example, if the board has 3 on the top, and if the opponent has a 3, then it is the card that must be played. You are welcome to improve the strategy, but make sure you discuss it in the report.

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.

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 10 points each; this is 70 points.

Report is 20 points.

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: 5 points.

Exceptions:

- If the program is not running, then you cannot get any points.
- If you do not submit your JAR file, then we cannot run your program, and you cannot get any points.
- If the program crashes on an input, you will lose a large amount of points. Test your code.
- If you cannot answer the questions in the oral exam, you can lose more than 5 points. For example, if we ask you a question about “shuffling” and you cannot answer it, then you will not get any points from the shuffling requirement.

Submitting Your Project

This project is not a team project. It is an individual project. You have to 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 30.12.2022, 23:59. Late submissions will NOT be accepted.