THE UNIVERSITY OF SYDNEY

INFO1113 Assignment 2 - Citadels

Due: 18 May 2025, 11:59PM AEST

This assignment is worth 16% of your final grade.

Task Description

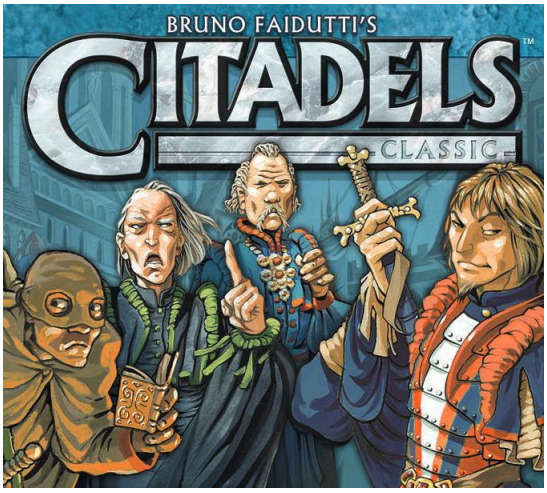
In this assignment, you will create a game in the Java programming language using gradle as a dependency manager. Your task is to implement a version of the Citadels card game (classic version). The game is played through the command line interface and may have between 4-7 players, one of which is the human player (player 1) and the others are computer-controlled players. You should refer to the following resources:

- Official game rules pdf
- 3 minute youtube video explaining the game and rules
- District cards reference file (src/main/resources/citadels/cards.tsv in the scaffold)

You are encouraged to ask questions on Ed under the assignments category if you are unsure of the specification – but staff members will not be able to do any coding or debugging in this assignment for you. As with any assignment, make sure that your work is your own, and do not share your code or solutions with other students.

Working on your assignment

You have been given a scaffold which will help you get started with this assignment. You can download the scaffold onto your own computer and invoke `gradle jar` to compile and resolve dependencies. After that, to run the game you can do `java -jar build/libs/citadels.jar`



Gameplay

Initial Setup

The game begins by prompting the user on how many players will be in the game. Then the initial setup of the game occurs, which involves shuffling the district card deck, randomly assigning the crown token, giving each player 2 gold tokens, and dealing 4 district cards to each player. Then rounds begin, and each round is divided into two phases: **Character Selection** phase, and **Turn** phase. Rounds will continue until the game ends (if one player builds 8 districts in their city, then the game will end at the end of the round) – then points are totalled, and a winner is determined.

Example output in the command line during this phase:

```
Enter how many players [4-7]:
> 6
Shuffling deck...
Adding characters...
Dealing cards...
Starting Citadels with 6 players...
You are player 1
```

 District Cards

The number of district cards is determined by the reference file `cards.tsv` (tab separated values format) which is provided in the scaffold. You need to read this file to set up the district card deck. Each row specifies the card name, colour and cost in gold, quantity of this card in the deck, and if it's purple, it also has a special ability which is described and should be displayed to the user if they type the "info" command.

 Character Cards

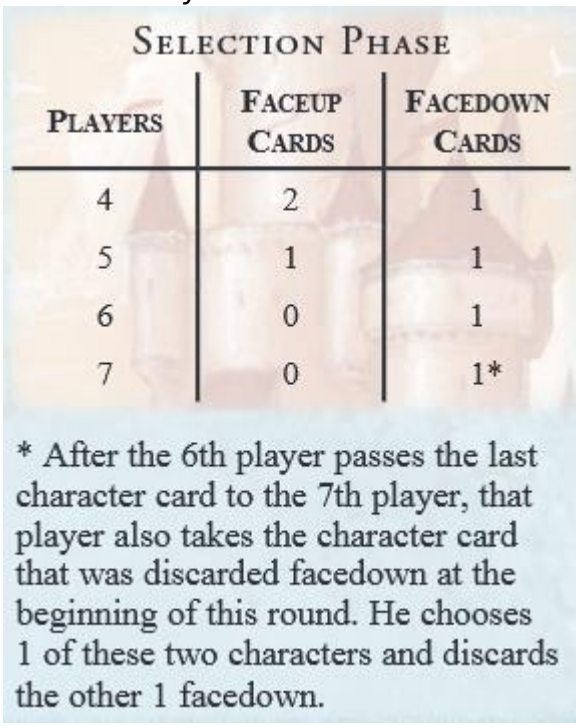
There are 8 character cards. Each has a special ability (refer to page 14 of the official rules). They perform their turns in the order provided:

1. Assassin - Select another character whom you wish to kill. The killed character loses their turn.
2. Thief - Select another character whom you wish to rob. When a player reveals that character to take his turn, you immediately take all of his gold. You cannot rob the Assassin or the killed character.
3. Magician – Can either exchange their hand with another player's, or discard any number of district cards face down to the bottom of the deck and draw an equal number of cards from the district deck (can only do this once per turn).
4. King – Gains one gold for each yellow (noble) district in their city. They receive the crown token and will be the first to choose characters on the next round.
5. Bishop – Gains one gold for each blue (religious) district in their city. Their buildings cannot be destroyed by the Warlord, unless they are killed by the Assassin.
6. Merchant – Gains one gold for each green (trade) district in their city. Gains one extra gold.
7. Architect – Gains two extra district cards. Can build up to 3 districts per turn.
8. Warlord – Gains one gold for each red (military) district in their city. You can destroy one district of your choice by paying one fewer gold than its building cost. You cannot destroy a district in a city with 8 or more districts.

Character Selection Phase

At the beginning of this phase, the 8 character cards are shuffled and game randomly discards 1 face down, and depending on the number of players, some also face-up (see Figure 1 below). The King cannot be discarded face-up, if this occurs, the game should try again with another random selection. For example:

```
Player 4 is the crowned player and goes first.
Press t to process turns
================================
SELECTION PHASE
================================
> t
A mystery character was removed.
King was removed.
The King cannot be visibly removed, trying again..
A mystery character was removed.
Bishop was removed.
Player 4 chose a character.
> sdhf
It is not your turn. Press t to continue with other player turns.
```



SELECTION PHASE

PLAYERS	FACEUP CARDS	FACEDOWN CARDS
4	2	1
5	1	1
6	0	1
7	0	1*

* After the 6th player passes the last character card to the 7th player, that player also takes the character card that was discarded facedown at the beginning of this round. He chooses 1 of these two characters and discards the other 1 facedown.

Figure 1 (left). Character discard rules for different numbers of players

Each player will take their turn choosing a character in order, from the remaining character cards handed to them by the previous player. The human player is always Player 1. So in the previous example above (5 player game), Player 5 would choose their character card next, and then Player 1 (human player) would choose, then Player 2, then Player 3.

Turn Phase

After all players have selected a character, the character numbers will be announced by the program in the order of their numbers (1-8 on page 2). If a character was discarded, it can be skipped. For example:

```
Character choosing is over, action round will now begin.
================================
TURN PHASE
================================

1: Assassin
No one is the Assassin
> t
2: Thief
```

```
Player 1 is the Thief
Your turn.
Who do you want to steal from? Choose a character from 3-8:
> 6
You chose to steal from the Merchant
Collect 2 gold or draw two cards and pick one [gold/cards]:
> gold
Player 1 received 2 gold.
> hand
You have 4 gold. Cards in hand:
1. Watchtower (red), cost: 1
2. Docks (green), cost: 3
3. Cathedral (blue), cost: 5
4. Market (green), cost: 2
> citadel
Player 1 has built:
> build 2
Built Docks [green3]
> citadel
Player 1 has built:
Docks (green), points: 3
> all
Player 1 (you): cards=3 gold=1 city=Docks [green3]

Player 2: cards=4 gold=2 city=

Player 3: cards=4 gold=2 city=

Player 4: cards=4 gold=2 city=

Player 5: cards=4 gold=2 city=

Player 6: cards=4 gold=2 city=
```

(see appendix for continuation of this example)

On each player's turn, unless their character has been killed, you must give that player the option of choosing to receive 2 gold, or drawing 2 cards and choosing 1. Then the player may build a district, spending their gold, as long as they can afford to do so with the gold they have. (Each character can only build 1 district per turn, except for the Architect which can build up to 3.) Players can also perform their character's special ability on their turn (they do not have to use the special ability). For the Assassin and Thief, they are prompted at the start of their turn for who they want to kill or steal from. Characters that receive additional gold for districts in their city will also have this happen automatically at the beginning of their turn, including the merchant who receives 1 additional gold. The King receives the crown token even if they are killed, and that player will go first in selecting characters on the next round. Other characters (Magician, Architect and Warlord) perform their special abilities later during their turn.

For computer-controlled players, you need to incorporate logic to determine the actions a computer-controlled player would take, including whether they decide to use a special ability. You may use random values to help with this decision-making, however try to give the computer controlled players some intelligence as well (for example, if they run out of cards or are running low, they might prefer selecting the Architect or Magician characters. And if they have enough gold to build a district, they will build the most expensive one they can. If they don't have any good districts, they might also prioritise choosing Magician or Architect to make use of those special abilities).

Commands

Your program must provide meaningful feedback to the user and not crash if invalid parameters are entered for any command. If an invalid command is entered, or `help`, display the help message which describes all commands and information about them. Any time the user is required to type something, you should describe the options `[option1/option2]` etc. If multiple commands are available, you can wait for the user to type with a prompt: ('>' sign).

The main general commands that should be available are listed below:

Command	Description
`t`	Process turns – proceed to the next sequence in the game, eg the next computer player makes their turn
`hand`	Display the cards in your hand and the amount of gold you have
`gold`	Display the amount of gold you have, eg: `You have 4 gold.`
`build <H>`	Builds a district from your hand. The parameter h is the position of the card in your hand. A player cannot have duplicate buildings in their city.
`citadel [p]`, `list [p]`, `city [p]`	Display the current districts in the player's city. If the parameter `p` is not provided, then display player 1's city. Example: `citadel 2` displays player 2's city.
`action`	Gives info about your character's special action. When you run this command, it should describe how you can perform your character's special ability. For example, the Magician may do `action swap <player number>` to swap their hand with the given player. Or they may do `action redraw <id1,id2,id3,…>` to choose to discard the given number of district cards from their hand and redraw them from the deck, where id1, id2 etc are the positions in the player's hand when doing the `hand` command.
`info <H>`	Gives information about the special ability of a purple building in your hand.
`info <name>`	Gives information about a character, as specified on page 2
`end`	Ends your turn. Display: `You ended your turn.`
`all`	Display info about all players, including number of cards in hand, gold and districts built in their city. See the example on the previous page
`save <file>`	Saves the current game state info using the JSON file format in the given file.
`load <file>`	Load the game state from the given file.

`help`	Display the help message, similar to this:
	`Available commands:`
	`info : show information about a character or building`
	`t : processes turns`
	
	`all : shows all current game info`
	`citadel/list/city : shows districts built by a player`
	`hand : shows cards in hand`
	`gold [p] : shows gold of a player`
	
	`build <place in hand> : Builds a building into your city`
	`action : Gives info about your special action and how to perform it`
	`end : Ends your turn`

`debug`	Toggles debug mode, where the hand of computer-controlled players will be visible on their turn in a debug message.

Any time a district building is displayed, you should output the name of the district, colour and gold cost value, like in the following format: `Docks [green3]`

Game End

The game ends after the round in which any player builds a completed city – that is, a city containing 8 districts. A city can have more than 8 districts. When the game ends, players score points as follows:

- Score points equal to the building cost of each of your districts.
- If your city has at least one district of each type, score 3 points.
- The player who first completed their city scores 4 points.
- Any other player who completed their city scores 2 points.
- Score any extra points from your unique districts.

The player with the most points wins. If there is a tie, the tied player who revealed the character with the highest-numbered rank in the last round wins.

Your program should display a detailed score calculation breakdown for all players and then congratulate the winning player.

Saving

Your program must have the ability to save the current game state to a file, when the user types the command `save <filename>.` You should use the JSON file format to do this, with the simple-json library. The program must also be able to load a saved game file from the format you created, with the command `load <filename>`.

Application

Your application will need to adhere to the following specifications:

- Jar file named `citadels.jar` must be able to be created when `gradle jar` is run, using Java 8. Then it must be able to be run with `java -jar build/libs/citadels.jar`
- Your program must not exhibit any memory leaks.
- You must use the JUnit testing framework for tests, which can be run with `gradle test` or `gradle test jacocoTestReport` to generate a code coverage report.

Marking Criteria (16%)

To submit, you must upload your build.gradle file and src folder to Ed. Please also include sample JSON save files that you have tested with your program. Do NOT submit the build folder (unless you only include the build/reports/ folder which contains the results of your testing and code coverage, and build/libs folder which contains your jar file). Ensure src is in the root directory with the other files, and not part of a zip, then press MARK. Submit your report and UML to canvas.

Demo / Viva (2%)

The demonstration is conducted during tutorials in week 12, where you will be asked to show the functionality to your tutor and they will ask you questions about your codebase and how you implemented the functionality. You may be asked at random to explain:

1. How you implemented the special ability of one or more of the purple district cards
2. How you implemented the special ability of a character card
3. How you implemented computer-controlled player decision logic

4. How you would extend the program to add an additional character from the Additional Character Set Expansion. For example, 9. Queen - Receives three gold during their turn if they are sitting next to the player possessing the No. 4 character (King, Emperor, etc.).

The demonstration will also involve you showing and explaining the codebase you wrote for Assignment 1 and may involve questions from your tutor about that.

Final Code Submission (8%)

You will need to have implemented and satisfied requirements listed in this assignment. Make sure you have addressed the following and any other requirements outlined previously.

- Program compiles successfully and sets up initial game state, initialising and shuffling district deck and character deck correctly, dealing 4 district cards to each player and 2 gold tokens
- Commands can be received from the user as described in the requirements
- Character selection phase progresses normally with adherence to the game rules
- Turn phase progress normally with adherence to the game rules.
- Implementation of character special abilities
    - Simple abilities: Assassin, Thief, King, Bishop, Merchant
    - Complex abilities: Magician, Architect, Warlord
- Implementation of district special abilities (purple districts)
- Human player can collect gold or cards at the start of their turn (draw 2 and discard 1)
- Human player can build districts. A city can only have unique districts (no duplicates)
- Computer-controlled players can make intelligent decisions.
    - Can choose characters
    - Can collect gold/cards
    - Can build districts
    - Can perform special abilities
- Game end, score calculation and winner is determined correctly
- Save / load game state to a JSON file format
- Game is bug free and does not crash with invalid input. Meaningful feedback is always shown to the user and the interface is clear and intuitive.

Testcases (3%)

During development of your code, add testcases to your project and test as much functionality as possible. You will need to construct unit test cases within the src/test folder using JUnit.

Ensure your test cases cover over 90% of execution paths (Use jacoco in your gradle build) – average of branches and instructions. Ensure your test cases cover common cases. Ensure your test cases cover edge cases. Each test case must contain a brief comment explaining what it is testing. To generate the testing code coverage report with gradle using jacoco, run "`gradle test jacocoTestReport`".

Design, Report, UML and Javadoc (3%)

You will need to submit a report that elaborates on your design. This will include an explanation of any object-oriented design decisions made (such as reasons for interfaces, class hierarchy, etc) and an

explanation of how the extension has been implemented. This should be no longer than 500 words. This report will be submitted through Canvas.

You will need to submit a UML diagram in PDF form to Canvas to provide a brief graphical overview of your code design and use of Object Oriented Principles such as inheritance and interfaces. Markers will use this to determine whether you have appropriately used those principles to aid you in your design, as well as figure out whether more should have been done. A general guideline is that markers will be looking for some use of inheritance or interfaces, how extensible the code is, and penalising repeated code. Note that you should not simply use a UML generator from an IDE such as Eclipse, as they typically do not produce diagrams that conform to the format required. We suggest using software such as LucidChart or draw.io for making your diagrams.

Your code should be clear, well commented and concise. Try to utilise OOP constructs within your application and limit repetitive code. The code should follow the conventions set out by the Google Java Style Guide. As part of your comments, you will need to create a Javadoc for your program. This will be properly covered in week 11 but the relevant Oracle documentation can be found here.

Report, UML and OO design:	2%
Javadoc, comments, style and readability:	1%

Suggested Timeline

Here is a suggested timeline for developing the project. Note that it is released on April 16 (week 8) and due May 18 (end of week 11).

Week 8: Familiarise yourself with gradle and the resources provided (game rules, youtube video and cards.tsv). Identify opportunities to utilise Object Oriented Design principles such as inheritance and interfaces and begin to plan a design for the codebase with regards to the classes that you will need to make. Make a rough UML diagram for your design that you can base your codebase from.

Weeks 8-9: Begin writing the actual code for the program. Start small, for example by initially creating the input loop, card deck, and player logic, then gradually add more like characters and other game elements. At the end of the week, you should have a basic version of the game with the character selection and turn phases. If confident, use Test Driven Development (writing test cases at same time as writing the code). Conduct a large amount of user testing to ensure the initial mechanics work as expected.

Weeks 9-10: Develop more gameplay features, such as the special abilities, and computer player decision logic. You should have a fairly high code coverage for your test cases at this stage. If you are noticing any questionable design decisions, such as God classes or classes that are doing things they logically should not be doing, this is the time to refactor your code.

Week 11: Finish developing the remaining features for your program, notably the save/load of game state to a json file and game end score calculation. Additionally, finish writing your testing suite. Create the UML and Javadoc for the program. Fix any remaining bugs that your code exhibits. Submit your code to Ed (by uploading the entire project and pressing MARK) and submit your UML to Canvas in PDF form.

Week 12: Demonstrate the completed program to your tutor during the week 12 lab. They will check each criteria item has successfully been completed, and may ask you questions about how you implemented it to test your understanding.

Academic Declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgment from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Computer Science, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of Computer Science for the purpose of future plagiarism checking.

Appendix – Further Examples

(Continuation of previous example)

```
> end
You ended your turn.
> list
Player 1 has built:
Docks (green), points: 3
> t
3: Magician
Player 4 is the Magician
Player 4 collected 2 gold
Player 4 built a Castle [yellow4] in their city.
> all
Player 1 (you): cards=3 gold=1 city=Docks [green3]

Player 2: cards=4 gold=2 city=

Player 3: cards=4 gold=2 city=

Player 4: cards=3 gold=0 city=Castle [yellow4]

Player 5: cards=4 gold=2 city=

Player 6: cards=4 gold=2 city=

> t
4: King
Player 6 is the King
Player 6 collected 2 gold
> t
5: Bishop
```

```
Player 3 is the Bishop
Player 3 collected 2 gold
Player 3 built a Castle [yellow4] in their city.
> t
6: Merchant
Player 2 is the Merchant
The Thief steals 2 gold from the Merchant (Player 2)
Player 2 collected 2 gold
Player 2 collected 1 gold from merchant action
Player 2 built a Tavern [green1] in their city.
> debug
Enabled debug mode. You can now see all player's hands.
> t
7: Architect
Player 5 is the Architect
Player 5 collected 2 gold
Player 5 drew 2 cards for architect action
Debug: Harbour [green4], Castle [yellow4], Monastery [blue3], Prison
[red2]
Player 5 built a Monastery [blue3] in their city.
> t
8: Warlord
No one is the Warlord
> t
Everyone is done, new round!
================================
SELECTION PHASE
================================
A mystery character was removed.
Player 6 chose a character.
> t
Choose your character. Available characters:
Assassin, Thief, Magician, Bishop, Warlord, Architect
> assassin
Player 1 chose a character.
> t
Player 2 chose a character.
> t
Player 3 chose a character.
> t
Player 4 chose a character.
> t
Player 5 chose a character.
Character choosing is over, action round will now begin.
================================
TURN PHASE
================================
1: Assassin
Player 1 is the Assassin
Your turn.
Who do you want to kill? Choose a character from 2-8:
> 2
You chose to kill the Thief
Collect 2 gold or draw two cards and pick one [gold/cards].
> cards
Player 1 chose cards.
Pick one of the following cards: 'collect card <option>'.
1. Market [green], cost: 2
2. Temple [blue], cost: 1
> collect card 1
You chose card Market [green2]
> t
```

```
Your turn.
> t
Your turn.
> hand
You have 3 gold. Cards in hand:
1. Watchtower (red), cost: 1
2. Cathedral (blue), cost: 5
3. Market (green), cost: 2
4. Market (green), cost: 2
> gold
You have 3 gold.
> build 2
You cannot afford to build this building.
> city
Player 1 has built:
Docks (green), points: 3
> build 3
Built Market [green2]
> all
Player 1 (you): cards=3 gold=1 city=Docks [green3], Market [green2]

Player 2: cards=3 gold=2 city=Tavern [green1]

Player 3: cards=3 gold=0 city=Castle [yellow4]

Player 4: cards=3 gold=0 city=Castle [yellow4]

Player 5: cards=5 gold=1 city=Monastery [blue3]

Player 6: cards=4 gold=4 city=
> end
You ended your turn.
> t
2: Thief
Player 2 is the Thief
Player 2 loses their turn because they were assassinated.
> debug
Disabled debug mode. You will no longer see all player's hands.
> t
3: Magician
Player 3 is the Magician
Player 3 collected 2 gold
Player 3 built a Temple [blue1] in their city.
```