Vanier College

Dominion in Java Report

Deck Building Game

David Hernandez

Yi Wang

Data Structures and OOP

11 May 2025

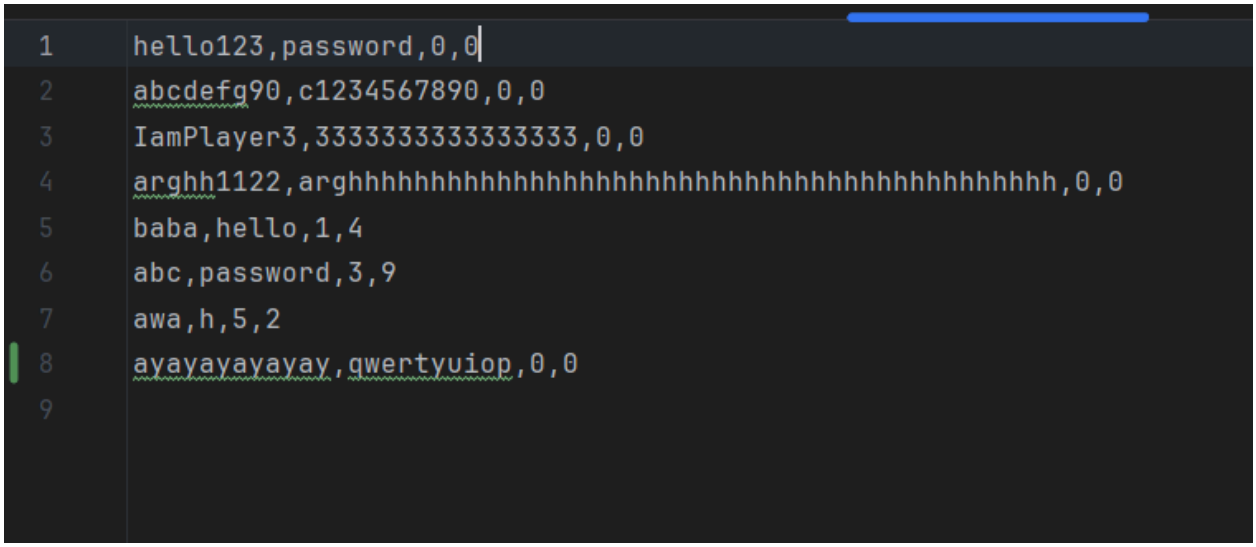
Table Of Contents

Project Description

This project aims to create a virtual space where 2-6 users can sign in to an account or play as guests, and play the beloved deck-building game Dominion.

Program Features

Users are able to create accounts and sign in to previous ones easily, which was part of the goals of the project:



```
C:\Users\User\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Users\User\AppData\Local\Programs\IntelliJ IDEA Ultimate\lib\
how many players? (1-6)
3
player 1, do you want to sign in, create an account, or play as a guest? (a/b/c)
a
player 1's name:
player 1
enter username:
hello123
enter your password:
sasldfhas
incorrect password, try again:
password
(player 1's discard has been shuffled into their deck)
player 2, do you want to sign in, create an account, or play as a guest? (a/b/c)
b
player 2's name:
player2
enter your new username: (must contain only letters and numbers, no spaces, at least 8 characters)
hello123
username already exists, try again:
ayayayayayay
enter your new password: (must contain only letters and numbers, no spaces, at least 8 characters)
qwertyuiop
(player2's discard has been shuffled into their deck)
player 3, do you want to sign in, create an account, or play as a guest? (a/b/c)
c
player 3's name:
player 3
```

Also, the game runs smoothly and is a perfect virtual replica of the real one. There are four interfaces (Action, Attack, Treasure, Victory. Deliverable 1 only predicted Attack), which are necessary to the project and work well. CardComparator is used often when listing cards in the terminal output and Player is comparable when listing the winners at the end of the game.

```
actionCards.sort(new Card.CardComparator( type: ""); // name alphabetically
```

```
treasureCards.sort(new Card.CardComparator( type: "cost"));
```

```
Collections.sort(players);
```

Challenges

The original plan was to have all kingdom cards from the original game in the code, yet this proved too long to code, so only seven simple kingdom cards were coded and one

complicated card (Bandit). Another difficulty is that my project structure changed over time. At the start, I wanted Action, Treasure, Victory to be abstract classes, but later I remembered that some cards in the original game can be both Treasure, Victory, and Action at the same time so abstract classes would not work. Thus, they became interfaces, which solved my problem. For a while, I tried to make the Card class implement Cloneable, but it didn't work for reasons I do not understand, so instead I made an abstract method in Card called copy() which would return a new card that is the same as `this`. Also, at some point, I wanted to make test cases for methods that used the Collections.shuffle method, which wouldn't work because the order of the elements would be different every time. So, I decided not to do testing for those methods which contained randomness or those containing terminal input.

Learning Outcomes

When doing this project, I learnt about the importance of class hierarchies. At first, I thought this concept useless because abstract methods don't even have a body so there's no point. Yet, for the structure, it is very useful to organize classes and everything is neater and makes more sense. I tried to make Dominion a year or two ago in Python, and there were a lot of bugs and it didn't work very well, also that I didn't implement OOP and all the code was in the same file. But this time, it worked much better. This is my first game in Java and I didn't start with something easy, but because of this, I have a lot more experience and familiarity with the language and game development in general than I used to. This is an interesting project and I will probably program all kingdom cards at some point and even create cards of my own.