Lauren Robison  
Canfield aka DEMON (I saw it had that nickname on the internet and thought it was fitting for this project lol)

Summary of the Project:

Problem: The project aimed to be able to play the Canfield solitaire game and to ensure that data validation was being done throughout to ensure cards and stacks could move to where the user was choosing accurately.

Approach: This Canfield project was approached in an object-oriented design manner to model the game components and how they interchanged. It uses a combination of conditional statements, loops, and functions to handle user input, validate moves between multiple classes of data, and can check the game status along the way.

Algorithms: The project incorporates various algorithms to determine the validity of moves, such as checking suit colors, values, and stack conditions. There was checks between cardType and cardType, cardType and stackType, cardType and foundation Type as a few examples. These algorithms include functions like checkIfValidMove() that would check if a cardType could be allowed to go on the stack the user wanted by comparing the suit and face elements. tableauType and foundationType got their own variants due to how the stacks need to have a specific order to be able to allow a cardType to be played on them. Another algorithm that tested me was the isStandStill() function that allowed the user to check at any time if there was still moves that could be played throughout the game before quitting.

Data Structures: Parent and child classes was a focus for this project. I had a parent class of deckType with cardType as a child. I had a parent class of stackType with reserveType, stockType, foundationType, tableauType all as it’s child classes.

Inputs and Results: The inputs to the project are provided by the user during the gameplay. The program prompts the user to select whether to peel 3 cards from stock, move a card, move a tableau stack, check game status, and to quit. There are menus following that have the user only input numbers that are labeled out after every decision to avoid issues. Users get to pick specific source and destination stacks, as well as other relevant choices. There’s a chance to win, have a stand still, or to quit.

