DEVELOPER GUIDE

# Sections

1. Overview
2. Features
   1. Structure
   2. Pick a Card, Any Card
   3. The Play ()
3. Improvements

## 1. Overview

This program runs one of the most popular card games, Crazy Eights. This program was built using a series of classes, which helps to separate the different parts of the class. Through a GUI integrated with key and mouse listeners, the user is able to play with an AI computer, and is even able to access a Settings menu, without interrupting the flow of the game, that allows for unique customization.

## 2. Features

### a. Structure

The structure of the program is split into many classes for clarity purposes, the very basic unit is the Card class, which stores all the information on a card, then comes the Deck class, which is simply a collection of Cards with some extra methods to deal with each. After the Deck class comes the Hand class, which is an abstract class that inherits methods from the Deck class and is used to construct the Human and Computer classes. The Human and computer classes are used to determine whether the program needs a user input or simply a decision made by the program.

At the topmost level, which is the GUI level, are the classes CrazyEights, Settings, End, Instructions, and ChooseSuit. ChooseSuit is used for the user to choose a suit after playing an 8, End is simply a JLabel and a button to signify the end of the program, Settings is used to allow the user to select a back, Instructions is a TextBox with a button to allow the user to review the rules without opening up the guide, and CrazyEights is the core of the program and contains the logic used to play the game and the GUI components.

### b. Pick a Card, Any Card

When the user clicks anywhere on the board, a mouselistener event is triggered. This event checks if the user has been clicking on the cards, the question mark or gear on the top corner, or just randomly. The position of cards follows this formula: (each card is 73x97 pixels)

x = 500 – (hand\_length\*15+58)/2, y = 570

Starting at that position, each subsequent card is places 15 pixels to the right, overlapping most of the card. Math is used to determine which card has been clicked.

During the player’s turn, the Human class’s choose method is called, which in turn calls the CrazyEights instance’s getChoice method. That method will serve as a buffer to stop the program from moving forward without the user selecting a card. In that method, as long as it is still the user’s turn, the program will sleep for 10 milliseconds, this is very important as without the sleep the program will stop working after the user gets 2 or 3 turns. The mouselisteners have been created so that it will only respond to card clicks during the user’s turn, at which point the value of a global variable, position, is set and the current player is changed to exit the while loop and changed back in the getChoice method to allow the game to continue normally. After the player chooses a card, the position is returned to the choose method and the card at that position will be played and returned to the main CrazyEights instance to continue the game.

### c. Getting Back

The Settings class is interesting in the sense that is pretty much a standalone feature that does not interrupt the running of the game, a Settings object must be created with a CrazyEights object as its parameter, this is because the class needs to call the setBack() method of the CrazyEights class to do a instantaneous change in the card backs and it will need the reference to the current game.

### d. The Big Play ()

The play() method is the essence of the game, as long as the game has not ended, it will be up and running. At the beginning of each turn, the method checks how man card out of the current player’s hand can be played, if that number is 0, a card will be drawn. The hand will then be sorted, a check will be performed to see which cards can now be played, and graphics will be repainted. Then, if the player has 1 or more cards that they can play, the choose method is called and a card is selected. The current card will be put into the discard pile and the selected cad will become the current card. Then program will check if the current card is a special card and execute functions accordingly.

If the deck from which to draw cards ever reaches 0, a new deck will be formed using the discard pile and shuffled and the discard pile will be reset to nothing.

After a card has been played, the program checks if the current player’s hand’s length has reached 0, if so, they have won the game. If not, a 2 second delay is applied so the user can see the actions of the computer players and the current player is incremented depending on the direction of play and made sure to be inside the boundaries [0,3] to ensure a valid player is playing.

## 3. Improvements

Due to time restrictions, many features that could have been developed have not, and below are some that could perhaps become a guideline on what to do next.

One improvement to increasing use experience would be to develop multiplayer and animations, multiplayer would allow multiple users to enjoy the game with each other and animations make it clearer what is happening on the board.

Another improvement is figuring out how to make the Enter keybinding work after exporting to .jar, currently, everything goes flawlessly when run in the IDE, but after exporting, for some mysterious reason the Enter keybinding des not work anymore.

A Third improvement would be to create more intelligent AIs, as of now, the AIs play the first card available and the chooses the suit that is most dominant in the hand. A slight improvement would be great.