2018-11-20

Joseph Allen

Java / Android Documentation

OPL Casino

Contents

[Log: 2](#_Toc530471731)

[October 26th: 2](#_Toc530471732)

[October 28th: 2](#_Toc530471733)

[October 29th: 2](#_Toc530471734)

[October 30th: 2](#_Toc530471735)

[October 31st: 2](#_Toc530471736)

[November 1st: 3](#_Toc530471737)

[November 2nd: 3](#_Toc530471738)

[November 3rd: 3](#_Toc530471739)

[November 4th 3](#_Toc530471740)

[November 5th 3](#_Toc530471741)

[November 6th: 4](#_Toc530471742)

[November 7th: 4](#_Toc530471743)

[November 8th: 4](#_Toc530471744)

[November 9th: 4](#_Toc530471745)

[November 10th: 5](#_Toc530471746)

[November 11th: 5](#_Toc530471747)

[November 12th: 5](#_Toc530471748)

[November 13th: 5](#_Toc530471749)

[November 14th 6](#_Toc530471750)

[November 15th: 6](#_Toc530471751)

[November 16th: 7](#_Toc530471752)

[November 17th: 7](#_Toc530471753)

# Bug Report:

No bugs that I am aware of.

# Feature Report:

## Non-Implemented:

* Cannot load in a seeded deck by itself
* AI cannot capture multiple sets
* Ai extending builds is implemented, however they are not tested in edge cases to a point where I feel confident, they are flawless. There are no known bugs with them.

## Implemented:

* All other required features
* Added the full action log which keeps tracks of all moves
* Decks and piles show number of cards which are remaining for them
* Only text input from user is to get the name of a save file
* Program auto selects required options for selected card, and unselects when card is unselected
* Program shows build value and build string representation next to the build
* Trail button is disabled when required to capture

# Description of Classes & Data Structures:

## Model:

### Action Log:

Static class which is responsible for maintaining the log, which is a Vector<String>. Able to add elements to the log and display. Once added to the log, all data is immutable and can only be read.

### Build:

Extends BuildType.

This class is used to represent a singular build of cards. Once a build is created, the cards are immutable, however the owner is mutable.

### BuildType:

Abstract class which extends Card Type

Guarantees the implementation of functions to get cards in a build as a flattened array and vector.

### Card

Extends CardType

Contains multiple constructors to create a card from different instances. Immutable once created.

### CardType

Abstract class at the top of the Card Hierarchy.

Says all children will have a suit, a value and contains helper functions to convert from Enums to values and vice versa.

### Computer

Extends Player.

This class contains all the logic to figure out which move to make for the Ai. The implementation of the function doMove has a precedence chart of:

1. Build (Extend / Create are equal precedence)
2. Capture Set
3. Capture Build / Identical Symbols
4. Trail

As the class looks for moves in that order, it returns the first occurrence it finds, as everything else will be an equal or lower precedence. Therefore, all moves of equal priority are executed based on the left most hand card of the AI going to the right.

In addition, the advisor which does move recommendation comes from this class. It works by creating a new computer object and copies the hand from the passed in object. Therefore, when the class runs its algorithm, it thinks it is making a normal move. One top of that, the name of the advisor changes the log string to be formatted as “Move Recommendation:” instead of as an action being performed.

### Deck

Maintains a vector of cards, and is able to load in from a saved state, or generate a new deck of 52 cards. It is responsible for maintaining the vector in a valid state at all times.

Once created, no cards can be added to the deck, only removed. The top of the deck is located as the left most side of the serialized text.

Deck is able to deal cards directly into a hand, or to put them into an array which is then up to the responsibility of whoever called it. Make sure the deal is valid as there is no way to return the cards once drawn.

### Hand

Is used to maintain a set of cards which need to be added and removed from. The hand is used as: Player hands, player piles, and table cards. As the hand object is used for table cards, that therefore means the object is able to handle having builds.

In order to keep builds out of areas where builds should not be, it is recommended to use overloaded functions which only accept the Card child class, or to use the mutator methods of the containing object to the hand, which will know the appropriate function to call.

### Human

Extends Player.

This class is used to format the raw data which is sent by the controller as the user taps the screen. Class does no validation of its own. Move validation is primarily done in round, or the controller in cases where it is wrong in every possible case.

### MultiBuild

Extends BuildType.

This class is responsible for maintaining the a vector of builds which contain it. Once the object is created, it is immutable except for the owner string. This class overloads the accessors functions as it needs to flatten the data before it is returned.

### Player

Abstract base class of the Player Hierarchy.

This class has an abstract entry point doMove which is the round can communicate to the player that a move needs to be performed. In addition, this class contains the implementation of printing out a move to the log.

In addition, contains all the accessor and mutator functions which are public. This includes helper wrapper functions for the scoring – Count aces and spade which call the appropriate hand function parameters. Other functions are entry points which are to receive data originally passed from the controller.

Lastly, contains wrapper functions which ensure static typing to the hands that only Cards will be able to reach them.

### PlayerMove

A container which contain the action preformed, the index of the selected card, and a vector of selected table indices. The table indices are stored in descending order to allow deletion iteration in normal iteration.

### PlayerSaveData

A container for all data related to a single player. Contains name, score, hand, and pile in their serialized form. This object is meant to be passed from Serializer to Player or Player to Serializer then discarded.

### Round

Main model entry point for the controller during the GameLoop. This class either contains or contains an ancestor of all the other model classes which are used in the game. Therefore, this class contains many wrapper functions to make code in the controller look much cleaner.

On top of that, the round is responsible for knowing which player is next to player, if the round is over or ongoing, and to validate a move that a player makes is valid. If the move is valid, the round then executes the move.

### Serializer

Static class responsible for loading and saving of files in the correct format. The init function is called in the starting activity, and the clear function should be called once all objects have read their data.

#### To Save:

Before a file is saved, ensure all fields have been set to the current value. The round class contains a wrapper function, serializeObjects() which does this task. However each object can be called individually. The Serializer then writes the data to the save file to the current active filename. If no filename has been specified before it is called, it will save to “save.csav”

#### To Load:

Serializer will load the file into all the appropriate member fields, and create all the accessor objects. It is up to each individual object to query the Serializer to get its save data.

### Tournament

This class is responsible for scoring the last played round per the scoring rules of the game, and checking if anyone won. It contains an overloaded toString function which can do a raw dump of the scoring calculation.

## View:

### BuildView

Extends CardView.

This class is responsible for drawing a build to a linear layout which is passed to the object via the controller. The 0th child gets the value of a build, the string representation, and who owns it, and then the rest of the children get the cards which are in the build. The card’s graphic is drawn by the appropriate card view.

Overloads the function requiredButtons to be equal to the number of cards in the build it is bound too.

### CardView

Responsible for drawing a card to an image button. Responsible for finding the correct Card image based the suit and value of the model.

### DeckView

Keeps track of all the card views for the cards in the deck. More specialized version of HandView

### HandView

Keeps track of all card views which match the cards in the hand. In addition, able to be polymorphic with build views as the table hands needs to use them.

### PlayerView

Container object for the Hand views of the player object it is bound to.

### RoundView

Container object of all the views that the Round model is responsible for keeping track of.

## Activities ( Controllers):

### ActionLogPopUp

The is the screen which displays the Action Log Graphically. On create, it goes through all the entries, starting with the newest, and displays them top down in a scroll view.

Goes to: Nowhere, only back

### CoinFlip

Asks the user to call the coin toss, gets the result, and passes this data onto the gameLoop.

Goes to: Game Loop

### GameLoop

This is the main game screen where a round is played. This screen shows the current deck, table, and players’ hands & piles. Also contains the menu to access the log and shows the most recent action that was taken directly on the table.

This is also the screen which is the main controller to the model and passes all information to the required parts within.

Goes To: Score Screen

Opens: ActionLogPopup, Turn Menu

### LoadScreen

This screen finds all the save files in the target directory, loads them into a scroll view, and when the user taps one, it has the Serializer load that file and continue the game.

Goes to: Game Loop

### ScoreScreen

After a round finishes, this is the screen which tallies up the score, and then uses that score information to decide if a new round is needed or a player won and to go to the main menu.

Goes to: Welcome, Game Loop

### TurnMenu

Submenu to the game loop screen, and this is the pop up menu which allows for the user to save the game, or quit without saving.

Goes to: Nowhere, only back or exit

### Welcome

Landing activity for the application. This calls the init functions for all the static classes, and has two options: New game which goes to coin toss, an Load save, which goes to LoadScreen.

Goes to: LoadScreen , Coin Flip

## Enums:

### CardSuit

Represents the suits of cards

Values: Invalid, Build, Club, Spade, heart Diamound

### PlayerActions

Action the player performed

Values: Invalid, Build, Capture, Trail

### PlayerID

Ids for initialization and Serialization

Values: humanPlayer CompPlayer. [Player 1, Player 2]

## TourScoreCode

Codes for how a tournament scored

Values: NoWinner, HumanWon, CompWon, Tie

# Log:

## October 26th:

* Prepared the repo with file structure, updating .gitignore (.1 hours)
* Ensured android studio works, emulator runs, and can deploy to device with example applications (1 hour)

## October 28th:

* Did android studios tutorial on how create an app and prepared environment to start work on the application (1.5 hours)
* Added a new activity for the coin flip
  + Design time (.2 hours)
  + Implementation time (1 hour)
* Designed a failed data structure to pass data (.5 hours)
* Added Skeleton for the cards ( 1 hour)
  + Card Model object
  + Card View
  + Card Enums for Suits

## October 29th:

* Previous activities end once new ones start to prevent user from going back (.5 hour)
* Buttons gray out after being clicked on coin toss (.5 hours)
* Added basic card functions, this includes values, constructors, and getters and setters (1 hour)
* Deck created and drawing card is implemented (.5 hours)
* Created skeleton for the hands, which include the underlying data structure, getters and setters, removal and adding cards. (1 hour)
* Created ability to draw four cards to a hand (.1 hour)
* Added views for all the created classes, and have them initialize based off the current state of the model (2 hours)

## October 30th:

* Created image buttons and displayed a test card (4 hours)
  + Many failed attempts were included in this
* Added images for all cards (.5 hours)
* Worked on xml for the human hand (.5 hours)
* Locked game orientation to portrait (.2 hours)
* Created help functions in view, allowing for all cards to be displayed to the image buttons ( 1 hour)

## October 31st:

* HandView is capable of drawing to all buttons, but not correct in all cases (1 hour)
* Improved initialization order to guarantee correct views are loaded before displaying (1 hour)
* Added PlayerMoves and PlayerID enum classes (.2 hours)
* Added skeleton for the player abstract class (1.5 hour)

## November 1st:

* Added computer hand to the XML file (.5 hour)
* Able to display the computer cards to the hand (.1 hour)
* Made the background table green (.1 hour)
* Round can initialize itself (.1 hour)
* Created PlayerView skeleton (.5 hour)
* Created basic player moves which always trails ( .5 hours)
* Converted the controller from using test variables to using the actual data vars (2 hours)
  + This needed more accessors and some code restructuring
* Cards start face-up (.1 hour)
* Fixed debug code to display cardBacks instead of testCards(.1 hours)

## November 2nd:

* Able to display N cards on the table section (1.5 hours)
* Change crop type of the cards so they display card with less whitespace (.2 hours)
* Added helper functions in controller to change color and set up pixels in density (.5 hours)
* Able to select and deselect cards (Graphically and in the model) (1.5 hours)
* Added vectors of ids for buttons declared programmatically so they can be accessed in loops (.2 hours)
* HandViews and hands can limit selection to one card at a time if needed (.2 hours)

## November 3rd:

* Able to set the submit and confirm buttons to correct state (1 hour)
* Able to correctly use string localization on the button (.2 hour)
* Able to trail cards graphically (3.5 hours)
* Updates cards in [graphic] hand after trailing (1 hour)

## November 4th

* Changes clickable of player hands and tables so the only clickable hands are the ones that are valid to be clicked at that point in time (1 hour)
* Computer trails correctly appear graphically (.1 hour)
* Program will now always correct unselect card after it is played (.5 hours)

## November 5th

* Hands and tables can now have their clickabilty toggled to match what is able to be clicked based on the current turn (.5 hours)
* Computer Trails update graphically (.2 hours)
* Human can now unselect cards they were tapped in their hand and reset the menu button (.5 hours)

## November 6th:

* Worked on manual and code formatting for mile stone submission (1 hour)
* Controller can now check for when the round ends (.2 hours)
* Internal logic updated so true means the same thing for all things in the main call stack (Contoller -> model) (.5 hours)
* Program is able to find all matching cards on the table, based on selected card (.5 hour)
* Table is able to clear all selected cards automatically, and forcibly select all cards which need to be captured (1 hour)
* Required cards can no longer be unselected (.2 hours)
* Table releases selected cards when the user is no longer required to capture them (.3 hours)

## November 7th:

* Fixed player view incorrectly initializing the pile observer (.5 hours to debug, .1 to fix)
* Rewrote Hand and HandView to use observers instead of being told when to update through polling / controller (3 hours)
* Hands properly display new cards in all cases when new cards are dealt from the deck (.5 hours)
* Created the Action Log which logs all moves that happen in the game, and displays thems in a separate activity, and as a button on the main screen (1 hour)
* Added more helper functions to Action Log to make it easier to integrate into the program (.2 hour)
* Added the correct hook-ups through-out the program to log what moves were taken (.5 hour)
* Controller can now removed buttons from the table once they are captured (.7 hours)
* Human is able to capture cards, with no rule validation (.5 hours)
* Made sure the removed card indices in Playermove is always sorted in descending order (.1 hour)
* Created the Human Pile in XML in preparation for captures being finished. (.5 hour)

## November 8th:

* PlayerMove can be told to mark itself invalid (.1 hour)
* Optimized the XML to make it more cleaner and so prevent it from having unintended complications whenever small changes are made (.5 hours)

## November 9th:

* User interface now properly updates when a move is rejected by the model (1 hour)
* Added radio buttons to the XML and hooked them up so the human can choose which move they want to make (1.2 hours)
* Fixed a bug in human logic where it would drop cards that it selected in data structure copying (.2 hours)
* Added accessors to get the piles from the views to allow them to be drawn (.1 hour)
* Captured cards are now added to the human pile graphically (.75 hours)
* Added a RoundView class with the round to maintain properly Model View Controller semantics (1.5 hours)
* Added a Computer Pile and the deck to the XML in preparation for them being added (.2 hours)

## November 10th:

* The deck is displayed on the main game screen (.5 hour)
* Tracked down a null pointer exception which only happened when the Computer goes first (.5 hours to find, .1 to fix)
* At end of rounds, cards go to the pile of who captured last (not graphically) (.5 hours)
* Cards now render graphically for both players (.5 hours)
* Radio buttons now disable trailing as an option when a player must capture cards on the table (.2 hours)
* Worked on how to display a Build on the table (1.5 hours prototyping different designs)
* Builds can now be created in the model (1 hour)

## November 11th:

* Builds can draw to the screen (.2 hours)
* Updated many functions in Controllers and views which were needlessly specific in the datatypes they accepted which prevented the Builds from displaying polymorphically. (2 hours)
* Cards in a captured build get added to the pile, however build graphic remains on the table (1.2 hours)
* Builds format correctly in the log (.1 hour)
* Builds act like normal image buttons through program logic, but not through user interaction ( 1 hour)

## November 12th:

* Builds can be selected or deselected like normal cards (.2 hours)
* Able to get to the load file screen (1 hour)
  + This includes design time for the screen
* Failed to implement file loading (3 hours)
  + This was do to a failure in understanding documentation and how android deals with file permissions
* Required cards unselect when not on the capture actions (.2 hours)
* Computer is now able (and required) to capture identical symbols on the table(.5 hours)
* Started to implement reserved values, which are used to track how many and what values are being used in a build (1 hour)

## November 13th:

* Prevent cards being selected in a duplicated manner for captures (this is due to a required capture stacking on top of a use selected capture) (.5 hours)
* Load file is able to find and display all save files in the directory (1 hour)
* Added the turn menu, which is able to quit the game (1 hour)
* Computer is now able to offer help to the player (1.5 hours)
* Ask for Help button is only visible while the human is making their move (.5 hour)
* Able to show the current player graphically (.2 hour)
* Added a switch which allows the turn menu to be accessed when the computer is the current player (.2 hour)
* PlayerSaveData class was added to store all player save in a single object (.25 hours)
* Added the framework to write data to a save file (.75 hours)
* Increased minimum api version to 26 from 21 (.1 hour)
* Fully able to write (what data is available) to a save file (1.5 hours)

November 14th:

* All new activities are now locked to portrait as the others are (.1 hours)
* Fixed bug where the application would close before finished writing data – by not closing the app. (.5 hour)
* Able to open a file and read in all its data (.5 hour)
* Able to remove blank lines (.1 hours)
* Able to clear headers from lines (.2 hours)

## November 15th:

* Human is able to capture sets (.5 hours)
* Fixed starting logic for the computer playing first on a save (.2 hour)
* Able to fully load in a file minus for build owners (1 hour)
* Able to now move to the Game Screen after reading in a file (.2 hour)
* Added more getter and setters to serializer to clean up code (.1 hour)
* Core Gameloop can now tell itself if its loading in a file or starting a new game (.5 hours)
* Able to create a card from a string to load aide in Serialization (.5 hour)
* Able to load in a deck from a string (.2 hour)
* Able to load in a save file in the round, which passes the data to the rest of the model (.5 hour)
* All other model files can parse their save file data (1 hour)
* Restructured model so builds and multi builds inherit from an abstract class BuildType (.5 hour)
* Fixed logic error in the remove blank line function which accidentally marked lines containing only whitespaces as non-blank (.5 hour)
* Overloaded toString in multi-builds (.1 hour)
* Added a new constructor in Hand for the table save which can parse builds (1 hour)
* Went over multiple functions to use build types instead of builds so both can be used (.5 hours)
* Added draw call on load for piles for save games (.1 hours)
* Multi builds display to the screen correctly (.3 hours)
* Added helper counting functions in preparation for scoring (.5 hours)
* Added round over logic to detect when round is over in Controller(.1 hour)
* Added skeleton to the tournament class for scoring and round continuation (.5 hours)
* Designed and added skeleton for the Score screen (1 hour)

## November 16th:

* Piles display on score screen (.5 hours)
* Added Build Strings to build buttons for readability (.1 hour)
* Computer move checkbox starts checked (.1 hour)
* Added move reason to the computer moves (.5 hours)
* Ai can capture sets up to size 3 (1.5 hour)
* Updated parsed lines in file loading to allow valid blank lines – such as empty piles (.75 hours)
* Table no longer makes all builds a multi build on loading in a file (.5 hour)

## November 17th:

* Fixed a bug which caused a desync between the controller, view, and model when a build was extended (4.5 hours to debug and fix)
  + The above fix also had a rewrite of the controller button logic to be more consistent, which was another (1 hour)
* Fixed cards being falsely added to a pile when multi builds were made (.5 hour)
* Fixed computer buttons being clickable in save files (.1 hour)
* Removed the BaseView superclass from all custom views as they were not used (.5 hour)
* Added rejection reasons in the action log from the model for when a move gets rejected (.2 hours)
* Found and fixed and off-by-one error in code rewrite which prevented builds from drawing correctly (.5 hour)
* Launcher icon is now the Ace of Spades (.1 hour)
* Did a passthrough of comments in preparation for submission (.75 hours)

## November 18th:

* Added catch for null pointer (.2 hours)
* After a build is made, controller will ask model to condense builds, which will turn builds into multi builds (1.5 hours)
* Fixed an error in above function where one if case used an index value instead of the value at the index (.2 hour)

## November 19th:

* Able to save the game with a custom file name (1 hour)
* Able to load in all build owners and assign them (1.5 hour)
* Able to write Build Owners tag to save files (.75 hours)
* Removed unused member variables in the Player Class (.2 hours)
* Tracked down a bug which would cause players to swap turn orders in a game (.75 hours)
  + Result was loading in a save file when players had a different amount of cards
* Added one second delay after saving a game before closing to ensure data is written to the disk (.2 hours)
* Scores now persist through all rounds of a tournament (.5 hours)
* Computer no longer tries to build with a reserved value (.2 hours)
* Added sum value of builds to the action log print out (.2 hours)
* Styled screens to have a consistent theme:
  + Created a round rectangle background (.2 hour)
  + Learned how to add shadows to text (.5 hour)
  + Styled the main menu (.75 hour)
  + Styled the load screen (.75 hour)
  + Styled the coin flip screen (.25 hours)
  + Styled the score screen (.25 hours)
  + Styled the turn menu (.25 hours)
  + Fixed labels being swapped in score screen (.1 hour)
  + Decorated main game screen (.5 hour)
* Added size of piles to pile label on game screen (.25 hours)

## November 19th:

* Styled and centered the score game button dynamically (.25 hour)
* Made starting screen buttons wider (.1 hour)
* Added proper comments to about half of the classes ( 3 hours)
  + Some files restructured to match required order
* Styled the action log sheet (.5 hours)
* Added styling to build buttons (.1 hour)

## November 20th:

* Worked on manual (in progress)

# Screenshots: