Assessment Criteria Definitions (including the corresponding metrics how each of these criteria is to be assessed)

Completeness of the solution direction (i.e. to what extent are all key functionalities covered in the design) [Functional completeness; functional correctness]

- Functional correctness - Absolute
  - Based on the key functionalities stated in Sprint 2:
    - set up the initial game board (including randomised positioning for dragon cards)
    - flipping of dragon ("chit") cards
    - movement of dragon tokens based on their current position as well as the last flipped dragon 1 card
    - change of turn to the next player
    - winning the game.

    How many of these key functionalities are implemented correctly
- Functional completeness - Ordinal
  - Degree to which set of functions covers all specified tasks (How much functionality the defined functions can cover)
    - Covers none - The defined functions cannot be used to cover any required functionality
    - Covers some - The defined functions can be used to cover some of the required functionality
    - Covers most - The defined functions can be used to cover most of the required functionality
    - Covers all - The defined functions can be used to cover all of the required functionality

Rationale behind the chosen solution direction [Functional appropriateness]

- Ordinal scale
  - How well the functions facilitate the execution of user stories/game rules, e.g., design patterns.
    - Not appropriate - the design of the system is completely unable to facilitate the functionality needed for the system
    - Somewhat appropriate - the design of the system is able to facilitate some of the functionality needed for the system
    - Very appropriate - the design of the system is able to facilitate most of the functionality needed for the system
    - Extremely appropriate - the design of the system is able to facilitate all if not all of the functionality needed for the system

Understandability of the solution direction [Appropriateness recognizability]

- Ordinal scale
  - How understandable is the facilitation of the execution of key game features; is rationale overly complex, or easy to grasp and work with
    - Not understandable - the rationale behind the design of the system is unable to be understood, or would require a disproportionate amount of time to grasp knowledge of implementation

        - Somewhat understandable - the rationale behind the design of the system is somewhat able to be understood and requires slightly more than expected amount of time to grasp knowledge of implementation
        - Very understandable - the rationale behind the design of the system is able to be understood quite easily and requires the expected amount of time to grasp knowledge of implementation
        - Extremely understandable - the rationale behind the design of the system is able to be understood extremely easily and requires much less time than expected to grasp knowledge of implementation

Extensibility of the solution direction (in anticipation of extensions to the game for Sprint 4) [Modifiability];

- Ordinal scale
    - How modifiable and extendable the solution is with regards to potential extensions to the game
        - Not extensible - The solution is not extendable, would have to change overall structure
        - Somewhat extensible - The solution is somewhat extensible, would have to change some structure
        - Very extensible - The solution is mostly extensible, would have to make minor changes to structure
        - Extremely extensible - The solution is completely extensible, no structural changes required

Quality of the written source code (e.g., coding standards, reliance on case analysis and/or down-casts) [Maintainability]; [*https://google.github.io/styleguide/javaguide.html*](https://google.github.io/styleguide/javaguide.html)

- Absolute + Ordinal
    - With regards to reliance on case analysis and or downcasts, it can be counted
    - With regards to coding standards, an ordinal scale is used for measure
        - Doesn’t conform - the coding standards used does not conform to Google Java Coding Guidelines
        - Somewhat conforms - the coding standards used somewhat conform to Google Java Coding Guidelines
        - Mostly conforms - the coding standards used mostly conforms to guidelines as specified by the Google Java Style Guide
        - Conforms - the coding standards used completely conforms to guidelines as specified by the Google Java Style Guide

Aesthetics of the user interface [User engagement];

- Ordinal
    - How likely is the user interface to encourage continued user interaction (considering given functions and information)
        - Not likely - User interface is not visually appealing, user would not want to continue playing
        - Somewhat likely - User interface is somewhat visually appealing, user would continue playing
        - Very likely - User interface is visually appealing, user would like to continue playing

Usability [Operability, self descriptiveness, user assistance]:

- Ordinal
    - How easy the system is to operate and control, and how understandable and accommodating the system is
        - Not understandable - User does not know how to or cannot navigate the game
        - Somewhat understandable - The user can navigate the game with difficulty
        - Extremely understandable - The user can navigate the game intuitively

Review of Prototypes

Group Member 1 Krishna:

Completeness of the solution direction (i.e. to what extent are all key functionalities covered in the design) [Functional completeness; functional correctness]

- Functional Correctness: 4 - does not account for edge cases when moving the player
- Functional Completeness: 5

Rationale behind the chosen solution direction [Functional appropriateness]

- Very appropriate - covers all functionality, and each class has its own purpose without obvious god classes. Deck class currently provides no value, however prepares for later extension

Understandability of the solution direction [Appropriateness recognizability

- Somewhat understandable //

Extensibility of the solution direction (in anticipation of extensions to the game for Sprint 4) [Modifiability];

- Very extensible

Quality of the written source code (e.g., coding standards, reliance on case analysis and/or down-casts) [Maintainability];

-

Aesthetics of the user interface [User engagement];

-

Usability [Operability, self descriptiveness, user assistance]:

-

Summary of key findings:

-

Group Member 2: Jeffrey KRISHNA DO

Completeness of the solution direction (i.e. to what extent are all key functionalities covered in the design) [Functional completeness; functional correctness]

- Functional Correctness: 5
- Functional Completeness: 5

Rationale behind the chosen solution direction [Functional appropriateness]

- Very appropriate - good logic behind functions, everything has purpose and no unnecessary classes. Deck class currently provides no value, however prepares for later extension. Turn is a static class, good use of static

Understandability of the solution direction [Appropriateness recognizability

- Very understandable

Extensibility of the solution direction (in anticipation of extensions to the game for Sprint 4) [Modifiability];

- Very extensible

Quality of the written source code (e.g., coding standards, reliance on case analysis and/or down-casts) [Maintainability];

-

Aesthetics of the user interface [User engagement];

-

Usability [Operability, self descriptiveness, user assistance]:

-

Summary of key findings:

-

Group Member 3: Zilei

Completeness of the solution direction (i.e. to what extent are all key functionalities covered in the design) [Functional completeness; functional correctness]

- Functional Correctness: 5
- Functional Completeness: 5

Rationale behind the chosen solution direction [Functional appropriateness]

- Very appropriate

Understandability of the solution direction [Appropriateness recognizability

- Very understandable

Extensibility of the solution direction (in anticipation of extensions to the game for Sprint 4) [Modifiability];

- Very extensible, turns are extremely extensible, cards and mappieces might need some slight modification

Quality of the written source code (e.g., coding standards, reliance on case analysis and/or down-casts) [Maintainability];

-

Aesthetics of the user interface [User engagement];

-

Usability [Operability, self descriptiveness, user assistance]:

-

Summary of key findings:

-