Kelas : 01
Nomor Kelompok : 05
Nama Kelompok : Doraemosantuy

1. 13520022 / Primanda Adyatma Hafiz
2. 13520046 / Hansel Valentino Tanoto
3. 13520073 / Lyora Felicya
4. 13520076 / Claudia
5. 13520121 / Nicholas Budiono

Asisten Pembimbing : Morgen Sudyanto

1. Diagram Kelas

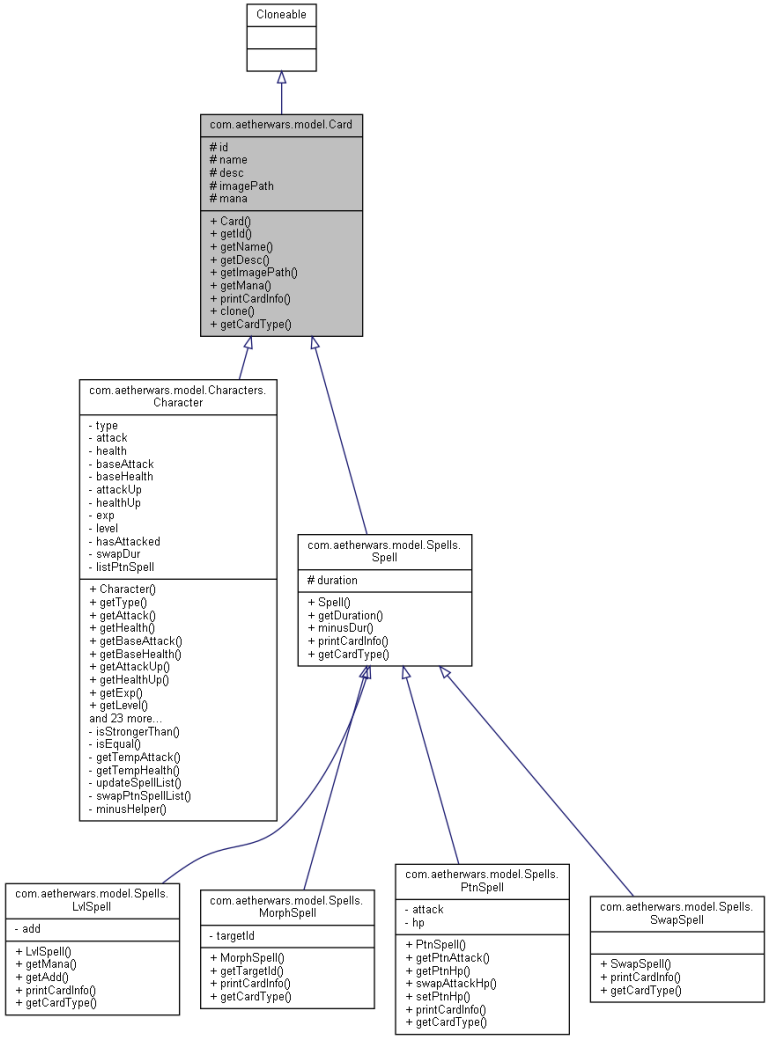


Diagram Kelas Card, Character, Spell, PtnSpell, SwapSpell, LvlSpell, MorphSpell

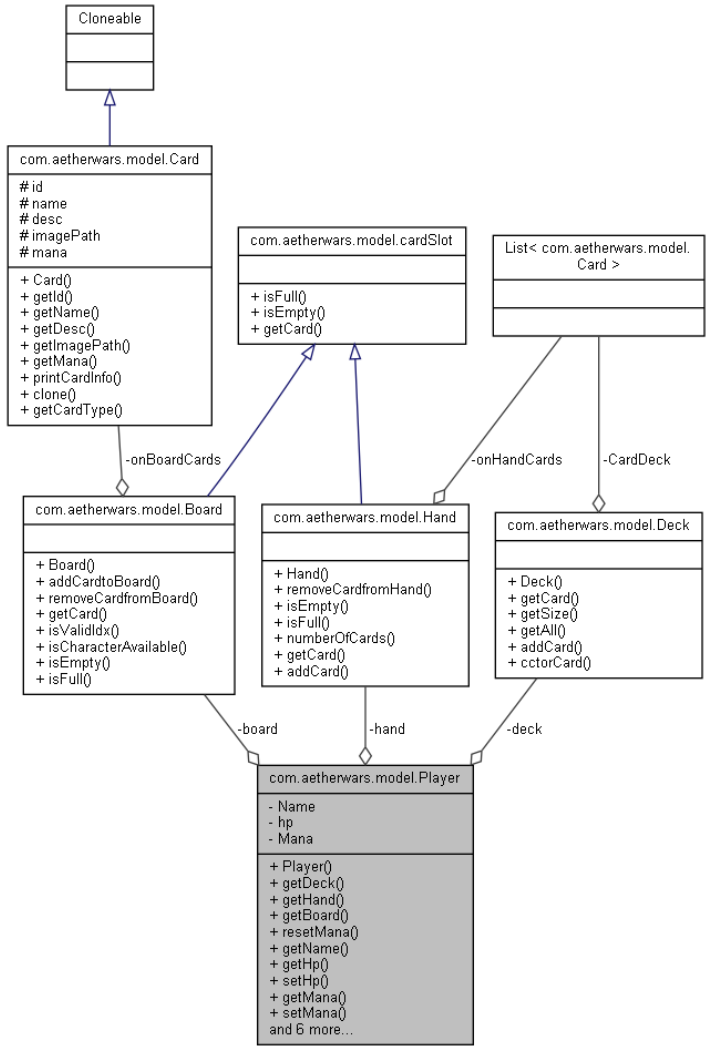


Diagram Kelas Player

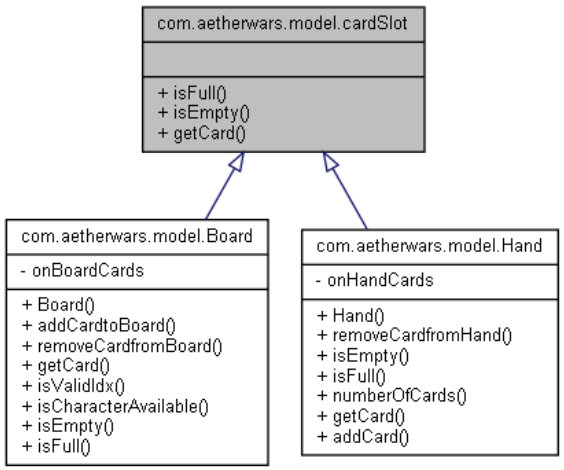


Diagram Kelas Board, Hand, cardSlot

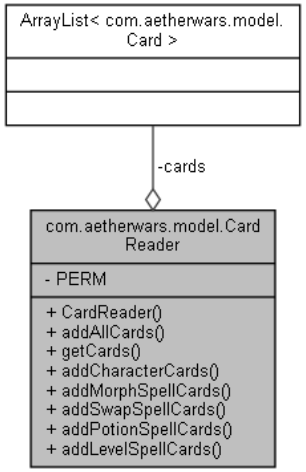


Diagram Kelas CardReader

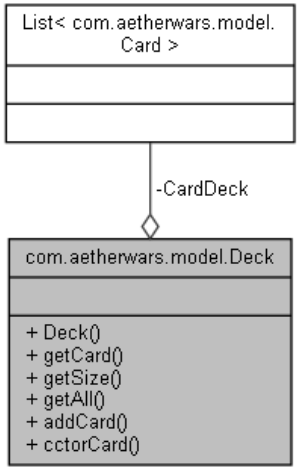


Diagram Kelas Deck

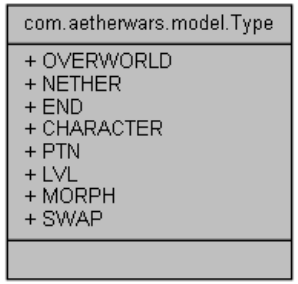


Diagram Kelas Type

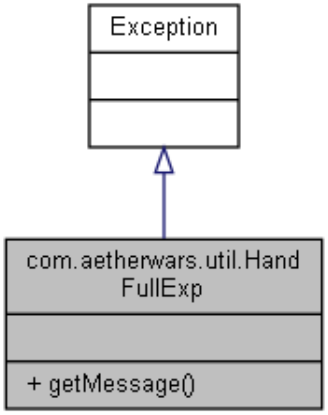


Diagram Kelas HandFullExp

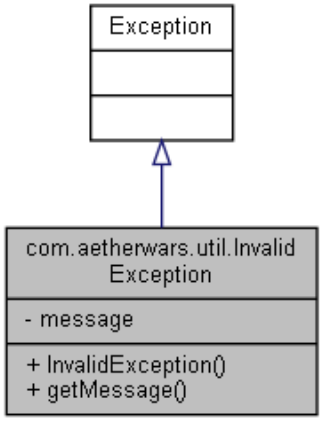
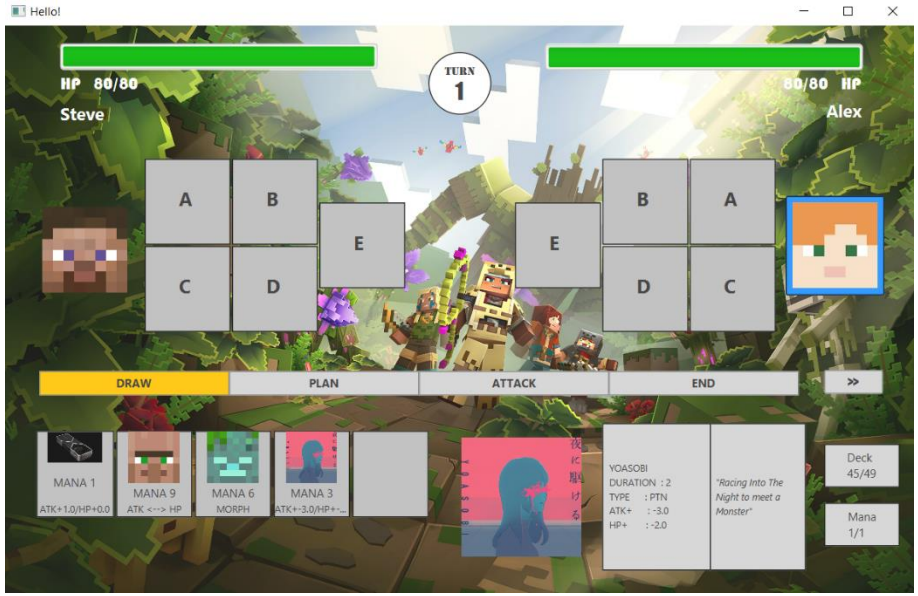


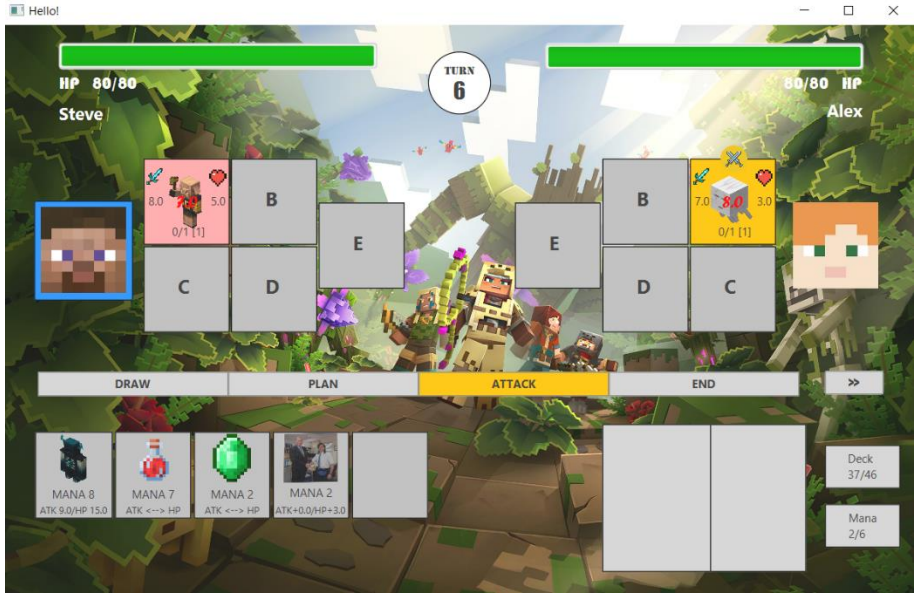
Diagram Kelas InvalidException

2. Deskripsi Umum Aplikasi

Aplikasi yang dibuat merupakan sebuah permainan kartu dengan sistem *turn based* yang dapat dimainkan oleh 2 pemain. Pemain akan bermain secara bergantian pada 1 layar yang sama. Tujuan dari *game* ini adalah menghabiskan *health points* musuh. Pada awal permainan, setiap pemain akan diberikan *deck* yang merupakan *list of cards*, merupakan campuran dari kartu *spell* dan kartu karakter. Karakter digunakan untuk menyerang lawan, sedangkan *spell* digunakan untuk melakukan modifikasi terhadap atribut dari karakter. Setiap pemain juga memiliki mana yang dibutuhkan untuk mengeluarkan kartu. Pemain akan secara bergiliran melakukan aksinya hingga memenangkan permainan. Setiap giliran terbagi ke dalam 4 *phase* yang terdiri atas *Draw Phase*, *Planning Phase*, *Attack Phase*, dan *End Phase*. Seorang pemain dinyatakan menang jika HP lawan telah mencapai 0 atau *deck* lawan kosong pada saat *Phase Draw* lawan.

Permainan ini dibuat dengan menggunakan bahasa pemrograman Java, dan implementasi *user interface*-nya dengan memanfaatkan JavaFX.





3. Konsep OOP

3.1. Inheritance

Beberapa penggunaan konsep *Inheritance* di aplikasi ini, yaitu:

- Kelas Character (src/main/java/com/aetherwars/model/Characters/Character.java) merupakan anak dari kelas Card (src/main/java/com/aetherwars/model/Card.java).
- Kelas LvlSpell (src/main/java/com/aetherwars/model/Spells/LvlSpell.java) merupakan anak dari kelas Spell (src/main/java/com/aetherwars/model/Spells/Spell.java), kelas Spell merupakan anak dari kelas Card (src/main/java/com/aetherwars/model/Card.java).
- Kelas PtnSpell (src/main/java/com/aetherwars/model/Spells/PtnSpell.java) merupakan anak dari kelas Spell (src/main/java/com/aetherwars/model/Spells/Spell.java), kelas Spell merupakan anak dari kelas Card (src/main/java/com/aetherwars/model/Card.java).
- Kelas SwapSpell (src/main/java/com/aetherwars/model/Spells/SwapSpell.java) merupakan anak dari kelas Spell (src/main/java/com/aetherwars/model/Spells/Spell.java), kelas Spell merupakan anak dari kelas Card (src/main/java/com/aetherwars/model/Card.java).
- Kelas MorphSpell (src/main/java/com/aetherwars/model/MorphSpell.java) merupakan anak dari kelas Spell (src/main/java/com/aetherwars/model/Spells/Spell.java), kelas Spell merupakan anak dari kelas Card (src/main/java/com/aetherwars/model/Card.java).

Alasan penggunaan konsep *inheritance* dalam implementasi kelas Character adalah karena Card terdiri atas 2 jenis yaitu Character dan Spell. Kedua jenis tersebut memiliki atribut dan *method* yang juga terdapat pada kelas Card, dengan beberapa atribut dan *method* tambahan yang hanya dimiliki oleh masing-masing tipe kartu. Dengan alasan yang serupa, konsep *inheritance* juga diimplementasikan pada kelas LvlSpell, PtnSpell, SwapSpell, dan MorphSpell yang merupakan *subclass* dari kelas Spell. Alasannya adalah karena terdapat 4 jenis *spell* yang masing-masing memiliki efek yang berbeda, sehingga terdapat atribut yang hanya dimiliki oleh *spell* tertentu. Namun keempat jenis kartu *spell* tersebut pasti memiliki atribut dari kelas Spell. *Inheritance* yang diimplementasikan termasuk ke dalam jenis *multilevel inheritance*, karena kelas Spell meng-*inherit* kelas Card. Keuntungan dari penggunaan konsep *inheritance* disini adalah *reusability* dari kelas Spell.

3.2. Composition

Konsep *Composition* yang digunakan dalam aplikasi ini, yaitu:

- Kelas Deck (src/main/java/com/aetherwars/model/Deck.java) memiliki banyak kelas Card (src/main/java/com/aetherwars/model/Card.java), yaitu dalam bentuk List of Card.
- Kelas Hand (src/main/java/com/aetherwars/model/Hand.java) memiliki banyak kelas Card (src/main/java/com/aetherwars/model/Card.java), yaitu dalam bentuk List of Card.
- Kelas Board (src/main/java/com/aetherwars/model/Board.java) memiliki banyak kelas Card (src/main/java/com/aetherwars/model/Card.java), yaitu dalam bentuk Array of Card.
- Kelas Player (src/main/java/com/aetherwars/model/Player.java) memiliki tepat satu kelas Board (src/main/java/com/aetherwars/model/Board.java), kelas Hand (src/main/java/com/aetherwars/model/Hand.java), dan kelas Deck (src/main/java/com/aetherwars/model/Deck.java).
- Kelas Character (src/main/java/com/aetherwars/model/Characters/Character.java) memiliki banyak kelas PtnSpell (src/main/java/com/aetherwars/model/PtnSpell.java), yaitu dalam bentuk List of Spell.

Composition dilakukan pada kelas Deck, kelas Board, dan kelas Hand dimana atributnya merupakan *List of Cards*. Alasan penggunaannya adalah karena ketiga kelas tersebut merupakan kelas yang berperan sebagai slot untuk menempatkan kartu. Kartu dapat dipindahkan dari Deck ke Hand, lalu dari Hand ke Board untuk kemudian dimainkan, sehingga dibutuhkan kelas Card untuk membuat sebuah *list* kartu untuk memenuhi fungsionalitas dari kelas Deck, kelas Hand, dan kelas Board. Dengan adanya *composition* ini, pembuatan ketiga kelas tersebut menjadi lebih mudah.

Selanjutnya penggunaan konsep *Composition* juga dapat ditemukan pada kelas Player yang memiliki tepat satu kelas Board, kelas Deck, dan kelas Hand. Artinya, Deck, Hand, dan Board merupakan atribut dari kelas Player dan dikelola oleh Player. Alasan penggunaannya adalah karena ketiga kelas tersebut merupakan bagian dari Player.

3.3. Interface

Konsep *Interface* yang digunakan dalam aplikasi ini, yaitu:

- Interface cardSlot (src/main/java/com/aetherwars/model/cardSlot.java)

Interface cardSlot ini akan di-*implement* oleh kelas Board dan Hand karena keduanya memiliki struktur data yang serupa sehingga menggunakan beberapa *method* yang memiliki kegunaan yang sama yaitu method isEmpty(), isFull(), dan getCard(). Oleh karena itu dibuat *interface* cardSlot agar *method-method* tersebut dapat dideklarasikan hanya sekali di file cardSlot.java dan bisa diimplementasi oleh banyak kelas.

3.4. Method Overriding

Konsep *Method Overriding* yang digunakan dalam aplikasi ini, yaitu:

- Method toString pada kelas Hand dan kelas Board

Implementasi method ini dapat di override oleh kedua kelas

3.5. Java API

Konsep Java API yang digunakan dalam aplikasi ini, yaitu:

- Kelas Deck (src/main/java/com/aetherwars/model/Deck.java) memiliki List of Card.
- Kelas Hand (src/main/java/com/aetherwars/model/Hand.java) memiliki List of Card.
- Kelas Character (src/main/java/com/aetherwars/model/Characters/Character.java) memiliki List of PtnSpell.
- Kelas CardReader (src/main/java/com/aetherwars/model/CardReader.java) memiliki List of Card.

Alasan penggunaan Java API berupa *List of* Card pada kelas Deck dan kelas Hand adalah karena keduanya harus dapat menyimpan sekumpulan kartu, dimana Deck tersusun atas 40 hingga 60 kartu dan Hand maksimal 5 buah kartu. Implementasi *List of* Card pada kedua kelas ini membuat pengelolaan dan akses terhadap kartu menjadi lebih mudah.

Penggunaan Java API juga terdapat pada kelas CardReader. Kelas CardReader berfungsi untuk menerima kartu yang dibaca dari file CSV lalu mengidentifikasi setiap jenis kartu yang dibaca dan memasukkannya ke dalam sebuah *list* kartu. Kelas Character juga memiliki List of PtnSpell, alasan penggunaannya adalah untuk mendapatkan informasi apakah terdapat potion spell yang sedang aktif pada Character tersebut. Dengan demikian maka perhitungan pengurangan health pada permainan akan lebih mudah untuk diatur.

3.6. SOLID

Konsep SOLID pada dalam aplikasi ini, yaitu:

- S: Kelas Deck (src/main/java/com/aetherwars/model/Deck.java) hanya bertugas untuk menginisiasi Deck dengan List of Cards dan mengembalikannya ke Player, Efek dari potion spell diatur oleh kelas PtnSpell.java (src/main/java/com/aetherwars/model/Spells/PtnSpell.java), Efek dari Level Spell diatur oleh Kelas LvlSpell (src/main/java/com/aetherwars/model/Spells/LvlSpell.java), Efek dari Swap Spell diatur oleh Kelas SwapSpell (src/main/java/com/aetherwars/model/Spells/SwapSpell.java), dan Efek dari Morph Spell diatur oleh Kelas MorphSpell (src/main/java/com/aetherwars/model/Spells/PtnSpell.java).
- L: Kelas Character (src/main/java/com/aetherwars/model/Characters/Character.java) dan Kelas Spell (src/main/java/com/aetherwars/model/Spell.java) merupakan turunan dari Kelas Card (src/main/java/com/aetherwars/model/Card.java)

Single Responsibility Prinsiple (SRP) terdapat pada Kelas LvlSpell, kelas SwapSpell, kelas MorphSpell, dan kelas PtnSpell. Alasan penggunaannya adalah jika semuanya di gabungkan ke dalam satu buah kelas yang sama, maka kelas tersebut akan memiliki tanggung jawab yang cukup besar, yaitu bertanggung jawab terhadap efek yang ditimbulkan setiap tipe Spell pada kartu Character. Dengan memisahkan setiap jenis kartu *spell* ke dalam kelas yang berbeda, maka masing-masing kelas tersebut hanya perlu bertanggung jawab untuk mengatur efek dari Spell masing-masing pada kartu Character. Selain itu, kelas Deck hanya bertugas untuk menginisiasi sebuah *list* yang berisi kartu, oleh karena itu terdapat konsep SRP juga pada kelas Deck.

3.7. Design Pattern

Konsep *Design Pattern* yang digunakan dalam aplikasi ini, yaitu:

- Digunakan *design pattern* Singleton yaitu kelas Player (src/main/java/com/aetherwars/model/Deck.java) hanya diinstansiasi di awal permainan, yaitu pemain1 dan pemain2.
- Digunakan *design pattern* Composition pada salah satu atribut dari kelas Character yaitu this.listPtnSpell yang merupakan tipe List<Card> yang merupakan komposisi dari kelas Card (src/main/java/com/aetherwars/model/Card.java)

4. Bonus Yang dikerjakan

4.1. Bonus yang diusulkan oleh spek

4.1.1. Animasi

Pada GUI yang kami buat terdapat animasi *drag and drop* pada card yang berfungsi untuk memindahkan card dari Hand ke Board pada saat fase *planning*. *Drag and drop* juga dibuat dengan memastikan bahwa card yang berada pada Hand hanya bisa dipindahkan ke Board pemain yang sedang mendapat giliran. Selain itu, setelah fase *planning* fitur *drag and drop* ini akan diberhentikan dan dijalankan kembali pada fase *planning* selanjutnya. *Drag and drop* diimplementasikan dengan memanfaatkan penambahan *event listener* pada objek yang akan dilakukan *drag* maupun *drop*. Berikut adalah contoh implementasi fungsi pada saat card didrop

```
player1BoardA.setOnDragDropped(new EventHandler<DragEvent>() {
    public void handle(DragEvent event) {
        //System.out.println("Set mana, mana = "+pemain1.getMana()+" card = "+currentHar
        if(currentBoard.isCharacterAvailable( idx: 0) && currentHand.getCard(currentDragHa
            // tambah spell ke character
            try {
                pemain1.useSpell(currentDragHand,  charChardIdx: 0, cardReader.getCards());
            } catch (InvalidException e) {
                e.printStackTrace();
            }
        }else if(!currentBoard.isCharacterAvailable( idx: 0) && currentHand.getCard(curren
            try {
                currentBoard.addCardtoBoard((Card)currentHand.getCard(currentDragHand).c
            } catch (CloneNotSupportedException e) {
                e.printStackTrace();
```

Kemudian nantinya setelah stage planning selesai maka seluruh efek *drag and drop* akan diberhentikan dengan contoh implementasi kodenya adalah sebagai berikut:

```
public void endDragAndDrop1(){
    player1BoardA.setOnDragDetected(null);
    player1BoardA.setOnDragOver(null);
    player1BoardA.setOnDragDropped(null);

    player1BoardB.setOnDragDetected(null);
    player1BoardB.setOnDragOver(null);
    player1BoardB.setOnDragDropped(null);

    player1BoardC.setOnDragDetected(null);
    player1BoardC.setOnDragOver(null);
    player1BoardC.setOnDragDropped(null);

    player1BoardD.setOnDragDetected(null);
    player1BoardD.setOnDragOver(null);
    player1BoardD.setOnDragDropped(null);
```

5. Library yang Digunakan

Dalam pembuatan *User Interface* dari program kami memanfaatkan *library* JavaFX sebagai *framework* dalam pembuatan GUI Java. JavaFX sendiri adalah salah satu teknologi dari Java yang ditujukan untuk mempermudah perancangan aplikasi yang kaya dengan konten multimedia seperti grafis, efek grafis, dan video. JavaFX dapat digunakan untuk membangun *software* di berbagai macam perangkat antara lain yaitu aplikasi *desktop* di Windows serta *web browser* di Windows, Linux, dan Mac OS. Alasan kami menggunakan JavaFX yaitu karena pembuatan GUI dengan JavaFX dapat dipermudah dengan adanya *designer* seperti *Scene Builder*. Selain itu, JavaFX juga dilengkapi dengan fitur animasi *drag and drop* serta *event listener* yang simpel sehingga mempermudah pembuatan animasi pada *interface* GUI. Selain itu, jika dibandingkan dengan *framework* GUI lainnya seperti Java Swing, JavaFX cenderung lebih simpel untuk dipelajari karena memiliki jumlah komponen yang lebih sedikit dan juga memberikan tampilan yang lebih modern dibandingkan Java Swing.

6. Pembagian Tugas

- 13520022 / Primanda Adyatma Hafiz
    - Bertanggung jawab untuk tampilan dan penggunaan GUI JavaFX pada saat permainan dalam fase drawing dan planning
    - Melakukan penggabungan GUI dengan program

- 13520046 / Hansel Valentino Tanoto
    - Bertanggung jawab untuk tampilan dan penggunaan GUI JavaFX pada saat permainan dalam fase attack dan end
    - Melakukan penggabungan GUI dengan program

- 13520073 / Lyora Felicya
    - Bertanggung jawab untuk merancang class Player
    - Bertanggung jawab untuk merancang class Board

- 13520076 / Claudia
    - Bertanggung jawab untuk merancang class Card baik untuk Character maupun Spell
    - Bertanggung jawab untuk membuat program yang membaca data konfigurasi dari Card

- 13520121 / Nicholas Budiono
    - Bertanggung jawab untuk merancang class Hand
    - Bertanggung jawab untuk merancang class Deck

7. Form Asistensi

Kelas : 01
Nomor Kelompok : 05
Nama Kelompok : Doraemosantuy
6. 13520022 / Primanda Adyatma Hafiz
7. 13520046 / Hansel Valentino Tanoto
8. 13520073 / Lyora Felicia
9. 13520076 / Claudia
10. 13520121 / Nicholas Budiono

Asisten Pembimbing : Morgen Sudyanto

1. Konten Diskusi

- Apakah ada rekomendasi untuk pembuatan GUI?
  Direkomendasikan menggunakan Java Fx untuk GUI.

- Apakah Spell bertipe Level memang tidak disediakan file .csv-nya?
  Memang tidak disediakan, karena hanya ada 2 jenis yaitu naik level dan turun level.
  Jadi, apakah bisa di-*hardcode*?
  Spell bertipe Level boleh di-*hardcode*.

- Bagaimana jika implementasi duration untuk Spell yang bersifat permanen diisi dengan nilai 0?
  Boleh, tapi lebih baik nilai duration pada Spell yang permanen bisa dijadikan sebagai *constant* untuk mempermudah *readability*

- Apa saja hal penting yang perlu diperhatikan dalam mekanisme *attack*?
  Setiap membunuh kartu lawan, kartu akan mendapat exp

- Bagaimana sebaiknya implementasi penggunaan Spell?
  Penggunaan potion lebih baik diimplementasi di *class* Character (tambahkan atribut saja, misalnya *list* usedPotion).

SwapSpell secara sederhana dilakukan dengan menukarkan semua angka yang berkaitan dengan kartu tersebut (card bersama buffnya).

- Apa saja contoh *design pattern* yang umum digunakan?
  Composition: *object* yang pakai *object* lain, misal Player mempunyai *list of* Card
  Singleton Design Pattern: sebuah kelas hanya diinstansiasi 1 kali, misal Board

2. *Screenshot* Bukti