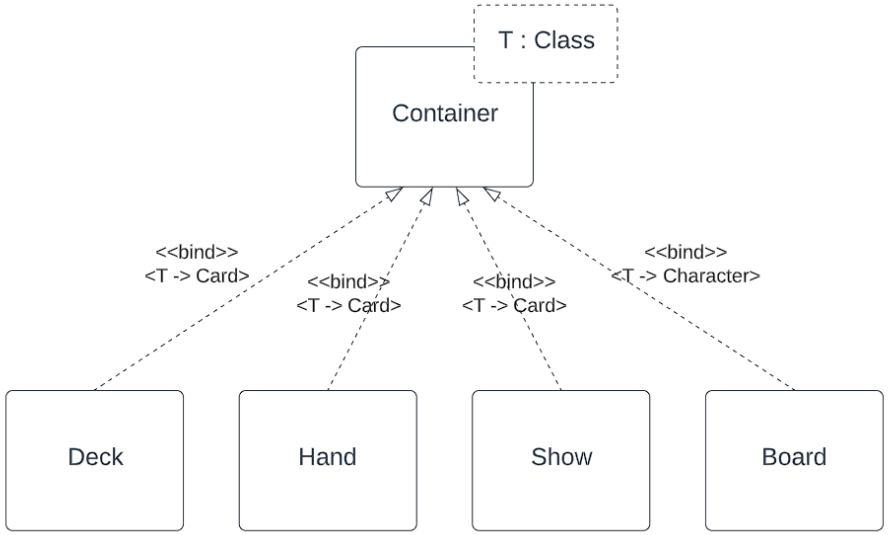
Kelas : 02
Kelompok : 01
1. 13520059 / Suryanto
2. 13520071 / Wesly Giovano
3. 13520092 / Vieri Mansyl
4. 13520113 / Brianaldo Phandiarta
5. 13520131 / Steven
6. 13520167 / Aldwin Hardi Swastia
Asisten Pembimbing : Morgen Sudyanto

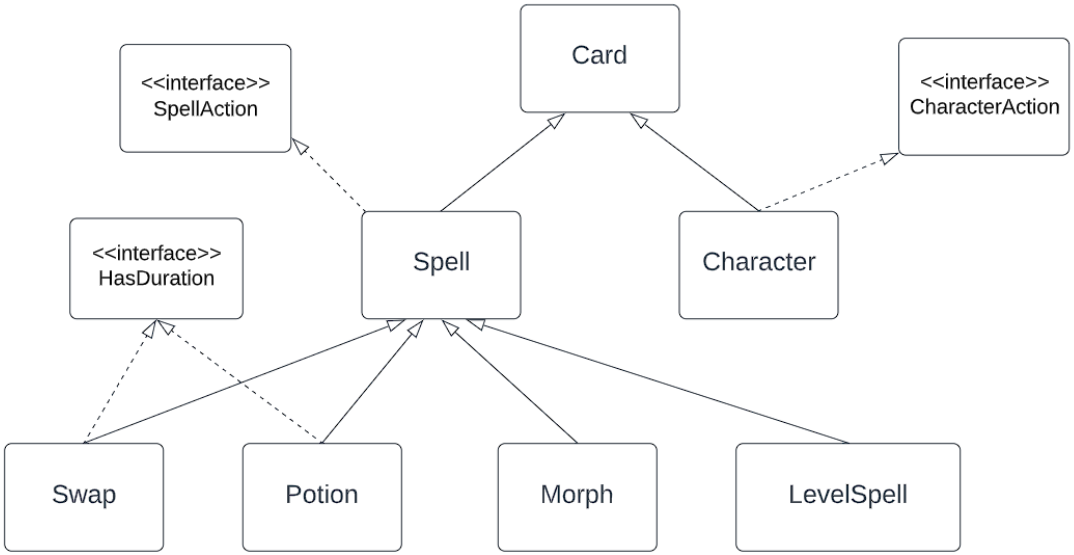
1. Deskripsi Umum Aplikasi

Aplikasi AetherWars merupakan permainan video berbasis desktop yang dikembangkan pada pada JavaFX menggunakan bahasa Java. Permainan AetherWars adalah permainan *kartu turn based* yang dimainkan oleh dua orang pada layar yang sama secara bergantian. Permainan memiliki 3 bagian, yaitu *draw*, *plan*, dan *attack*. Pada bagian draw, pemain akan mengambil tiga kartu dari deck kemudian memilih satu kartu yang nantinya 2 kartu lain akan dikembalikan ke deck. Kemudian pada bagian plan, pemain dapat menaruh kartu character dan spell pada board. Yang terakhir pada bagian attack, pemain dapat menyerang menggunakan kartu character yang terdapat pada board bagian pemain.

2. Konsep OOP

Berikut adalah kelas diagram utama dari program yang dibuat.


T : Class
Container
<<bind>>
<T -> Card>
<<bind>>
<T -> Card>
<<bind>>
<T -> Card>
<<bind>>
<T -> Character>
Deck
Hand
Show
Board



2.1. Inheritance

Berikut adalah 5 contoh inheritance yang diimplementasikan dalam program yang dibuat.

- Kelas Board (src/main/java/com/AetherWars/model/Board.java) merupakan anak dari kelas generik Container<T> (src/main/java/com/AetherWars/model/Container.java) dengan T adalah kelas Character.
- Kelas Deck (src/main/java/com/AetherWars/model/Deck.java) merupakan anak dari kelas generik Container<T> (src/main/java/com/AetherWars/model/Container.java) dengan T adalah kelas Card.
- Kelas Character (src/main/java/com/AetherWars/model/Character.java) merupakan anak dari kelas Card (src/main/java/com/AetherWars/model/Card.java).
- Kelas Spell (src/main/java/com/AetherWars/model/Character.java) merupakan anak dari kelas Card (src/main/java/com/AetherWars/model/Card.java).

- Kelas Potion (src/main/java/com/AetherWars/model/Potion.java) merupakan anak dari kelas Spell (src/main/java/com/AetherWars/model/Spell.java).

2.2. Composition

Berikut adalah 4 contoh composition yang diimplementasikan dalam program yang dibuat.

- Kelas Player (src/main/java/com/AetherWars/model/Player.java) memiliki sebuah Deck (src/main/java/com/AetherWars/model/Deck.java), sebuah Hand (src/main/java/com/AetherWars/model/Hand.java), dan sebuah Board (src/main/java/com/AetherWars/model/Board.java).
- Kelas BoardController (src/main/java/com/AetherWars/controller/BoardController.java) memiliki sebuah SplitPane, 11 buah Pane (10 untuk Pane tempat kartu dan 1 untuk deskripsi kartu), 2 buah ImageView, 4 buah FlowPane, 4 buah Button, 4 buah Label, sebuah Text, dan 3 buah Player (src/main/java/com/AetherWars/model/Player.java) dengan 2 buah Player yang bermain dan sebuah Player yang menunjukkan Player yang sedang bermain.
- Kelas Board (src/main/java/com/AetherWars/model/Hand.java) memiliki sebuah List<Character> yang sebenarnya mengandung 5 buah Character (src/main/java/com/AetherWars/model/Character.java).
- Kelas HandCharacterCardController (src/main/java/com/AetherWars/controller/HandCharacterCardController.java) memiliki sebuah Pane, sebuah ImageView, 3 buah Label, dan sebuah Character (src/main/java/com/AetherWars/model/Character.java).

2.3. Interface

Berikut adalah 3 buah interface yang diimplementasikan dalam program yang dibuat.

- Interface CharacterAction (src/main/java/com/AetherWars/model/CharacterAction.java)
- Interface HasDuration (src/main/java/com/AetherWars/HasDuration.java)
- Interface SpellAction (src/main/java/com/AetherWars/SpellAction.java)

2.4. Method Overriding

Berikut adalah 5 contoh method overriding yang diimplementasikan dalam program yang dibuat.

- Method addCard pada kelas Hand (src/main/java/com/AetherWars/model/Hand.java).
- Method getInfo pada semua subkelas dari Spell (src/main/java/com/AetherWars/model/Spell.java).
- Method effect pada semua subkelas dari Spell (src/main/java/com/AetherWars/model/Spell.java).
- Method getDuration pada kelas Swap yang mengimplementasikan interface HasDuration (src/main/java/com/AetherWars/model/Swap.java).
- Method addDuration pada kelas Swap yang mengimplementasikan interface HasDuration (src/main/java/com/AetherWars/model/Swap.java).

2.5. Polymorphism

Berikut adalah 5 buah polymorphism yang diimplementasikan dalam program yang dibuat.

- Pada kelas Character (src/main/java/com/AetherWars/model/Character.java) memiliki atribut kelas Spell, tapi pada method spellEffect di kelas Character (src/main/java/com/AetherWars/model/Character.java) memanggil spell.effect yang merupakan berbagai effect spell (LevelSpell, Morph, Potion).
- Pada kelas Spell (src/main/java/com/AetherWars/model/Spell.java), dapat dipanggil getInfo yang merupakan berbagai informasi spell (LevelSpell, Morph, Potion).
- Pada kelas HasDuration (src/main/java/com/AetherWars/model/HasDuration.java), terdapat getDuration yang merupakan durasi sebuah spell (Swap, Potion).
- Pada kelas HasDuration (src/main/java/com/AetherWars/model/HasDuration.java), terdapat addDuration yang merupakan penambahan durasi sebuah spell (Swap, Potion).
- Pada kelas Container (src/main/java/com/AetherWars/model/Container.java), terdapat addCard yang merupakan penambahan kartu pada sebuah container (Hand, Show).

2.6. Java API

Berikut adalah 5 buah contoh Java API yang digunakan dalam program yang dibuat.

- Kelas Container (src/main/java/com/AetherWars/model/Container.java) menggunakan Java Collection API yaitu pada atribut bertipe List <T extends Card> yang memanfaatkan ArrayList pada konstruktornya.
- Kelas Character (src/main/java/com/AetherWars/model/Container.java) menggunakan Java Collection API yaitu pada atribut bertipe List<Spell>.

- Kelas Player (src/main/java/com/AetherWars/model/Player.java) memanfaatkan Java Stream API pada pemanggilan method newTurn dari cards di Board.
- Kelas Character (src/main/java/com/AetherWars/model/Character.java) memanfaatkan Java Stream API pada pemanggilan method spellEffect untuk memanggil method effect dari attachedSpells.
- Kelas Deck (src/main/java/com/AetherWars/model/Deck.java) memanfaatkan Java Collection API pada bagian algoritmanya yaitu shuffle pada pemanggilan method shuffleCards.

2.7. SOLID

Berikut adalah konsep SOLID yang digunakan dalam program yang dibuat.

1. S - Single-responsiblity Principle
   (a) Kelas Hand (src/main/java/com/AetherWars/model/Hand.java) hanya bertugas untuk menangani kartu yang memang sudah di tangan player, sedangkan tiga buah kartu yang akan dipilih dari Deck dan dimasukkan ke Hand ditangani oleh kelas Show (src/main/java/com/AetherWars/model/Show.java).
   (b) Kelas Character (src/main/java/com/AetherWars/model/Character.java) salah satu tugasnya yaitu mendapatkan efek dari spell, bagaimana spell tersebut akan mempengaruhi Character sepenuhnya ditangani oleh subkelas dari Spell (src/main/java/com/AetherWars/model/Spell.java).
   (c) Kelas Board (src/main/java/com/AetherWars/model/Board.java) hanya bertugas sebagai lapangan permainan untuk Character, kelakuan dari Character sendiri maupun terhadap Character lain sepenuhnya ditangani oleh kelas Character (src/main/java/com/AetherWars/model/Board.java).
2. O – Open-closed Principle
   (a) Kelas Container (src/main/java/com/AetherWars/model/Container.java) merupakan kelas yang memiliki List <T extends Card>. Kelas ini terbuka untuk ekstensi tetapi tertutup untuk modifikasi, dalam arti kelas lain dapat diturunkan dari kelas Container tanpa mengubah isi dari kelas tersebut. Misalnya, jika hendak ditambahkan sebuah kelas Graveyard yang menampung kartu-kartu yang telah di-destroy, tidak perlu lagi mengubah isi dari kelas Container, tetapi kita dapat menambah perilaku dan tanggung jawab dari Container pada Graveyard.
   (b) Kelas Spell (src/main/java/com/AetherWars/model/Spell.java) merupakan kelas untuk menampung kelas-kelas yang merupakan kelas Spell. Kelas ini terbuka untuk ekstensi tetapi tertutup untuk modifikasi, dalam arti kelas lain dapat diturunkan dari kelas Spell tanpa mengubah isi dari kelas Spell. Misalnya, jika hendak ditambahkan jenis Spell baru, kelas Spell tidak perlu dimodifikasi lagi, cukup menambah perilaku yang baru dari spell baru tersebut.

3. L – Liskov Substitution Principle
   (a) Kelas Character (src/main/java/com/AetherWars/model/Character.java) yang merupakan subkelas dari kelas Card (src/main/java/com/AetherWars/model/Card.java) dapat disubstitusi menjadi kelas Card. Hal ini secara logika masuk akal, karena Character merupakan Card (hubungan is-a), setiap atribut dan method yang dimiliki oleh Card juga sewajarnya dimiliki oleh Character. Sebagai aplikasinya, kelas Character dapat disimpan dalam Hand, dan Hand menampilkan atribut-atribut Character sebagai Card.
   (b) Kelas Potion (src/main/java/com/AetherWars/model/Potion.java) yang merupakan subkelas dari kelas Spell (src/main/java/com/AetherWars/model/Spell.java) dapat disubstitusi menjadi kelas Spell. Hal ini secara logika masuk akal, karena Potion merupakan Spell (hubungan is-a), setiap atribut dan method yang dimiliki oleh Spell juga sewajarnya dimiliki oleh Potion. Sebagai aplikasinya, kelas Potion dapat disimpan dalam attachedSpells di Character, dan dapat diterapkan method effect pada Potion.
   (c) Kelas Hand (src/main/java/com/AetherWars/model/Hand.java) yang merupakan subkelas dari kelas Container (src/main/java/com/AetherWars/model/Container.java) dapat disubstitusi menjadi kelas Container. Hal ini secara logika masuk akal, karena Hand merupakan Container (hubungan is-a), setiap atribut dan method yang dimiliki oleh Container juga sewajarnya dimiliki oleh Hand. Sebagai aplikasinya, dalam permainan digunakan method addCard pada saat pengambilan kartu dari Deck ke Hand, removeCard ketika memindahkan kartu dari Hand ke Board, dan sebagainya.
4. I – Interface Segregation Principle
   (a) Interface HasDuration (src/main/java/com/AetherWars/model/SpellAction) disegregasi dari interface SpellAction (src/main/java/com/AetherWars/model/SpellAction.java) karena tidak semua subkelas dari Spell memiliki durasi, melainkan ada yang langsung dipakai saat diletakkan ke kartu karakter.
5. D – Dependency Inversion Principle
   (a) Kelas Character (src/main/java/com/AetherWars/model/Character.java) bergantung kepada abstractions. Hal ini dapat dilihat pada kelas Character yang mengimplementasikan sebuah kelas abstrak yaitu CharacterAction.
   (b) Kelas Potion (src/main/java/com/AetherWars/model/Potion.java) bergantung kepada abstractions. Hal ini dapat dilihat pada kelas Potion yang mengimplementasikan sebuah kelas abstrak yaitu HasDuration.
   (c) Kelas Swap (src/main/java/com/AetherWars/model/Swap.java) bergantung kepada abstractions. Hal ini dapat dilihat dari kelas Swap yang mengimplementasikan sebuah kelas abstrak yaitu HasDuration.

2.8. Design Pattern

Beberapa design pattern yang digunakan yaitu sebagai berikut.

1. Composite, yaitu sebuah objek tersusun dari beberapa objek lain sebagai komponennya, dan objek komponen tersebut dapat juga terdiri dari objek-objek lainnya. Dalam program yang dibuat, salah satu implementasinya yaitu kelas Player terdiri dari objek Board, Hand, Show, dan Deck. Board sendiri tersusun dari List of Character, sedangkan Hand, Show, dan Deck tersusun dari List of Card.
2. Prototype, yaitu sebuah objek dibuat berdasarkan objek lain, layaknya *copy constructor* pada bahasa C++. Dalam program yang dibuat, salah satu implementasinya yaitu semua data kartu yang dibaca dari konfigurasi file akan dijadikan objek kartu, yang selanjutnya akan berfungsi hanya sebagai template ketika pembuatan Deck. Dengan kata lain, objek kartu hasil pembacaan konfigurasi file bertindak sebagai prototype.
3. Chain of Responsibility, yaitu command yang dipanggil dilanjutkan/diturunkan ke objek lain. Dalam program yang dibuat, salah satu implementasinya yaitu Player akan menerima command newTurn. Setelah itu, Player akan mendelegasi command newTurn ke kartu-kartu yang dimilikinya pada Board.
4. Singleton, yaitu sebuah kelas hanya memiliki sebuah objek. Dalam program yang dibuat, salah satu implementasinya yaitu hanya ada satu BoardController yang aktif dan digunakan, sehingga BoardController merupakan Singleton. Untuk mendapatkan BoardController tersebut, dapat digunakan method getInstance() agar dapat mengakses method-method yang dimilikinya.

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Import Deck

Importing deck diimplementasikan dengan mengambil data deck dari file.csv yang berisi id dari setiap kartu pada deck. Loading deck dilakukan dengan memanggil method loadDeck(filePath). Berikut adalah snippet dari aplikasi import deck.

```
public Deck loadDeck(String srcDeck) throws IOException, URISyntaxException, HandOverException {

    Deck tes = new Deck();
    File CSVFile = new File(getClass().getResource(srcDeck).toURI());
    CSVReader cardReader = new CSVReader(CSVFile, separator: "\t");
    List<String[]> cardID = cardReader.read();
    for (String[] id : cardID){
        Card current = Card.getNewCard(Integer.parseInt(id[0]));
        try {
            tes.addCard(current);
        } catch (Exception e){
            //do nothing ?
            throw e;
        }
    }
    tes.shuffleCards();
    return tes;
}
```

Konfigurasi deck ditulis dalam sebuah file, dengan setiap entri baris adalah id dari kartu yang diinginkan.

4. Pembagian Tugas

- 13520059 / Suryanto
  - Mengimplementasikan kelas Character, Spell, dan kelas-kelas terkait.
  - Melakukan connecting antara Frontend dengan Backend.

  - Menyusun dokumen.
- 1352071 / Wesly Giovano
  - Mengimplementasikan kelas Player, Card, Character, dan kelas-kelas terkait.
  - Melakukan connecting antara Frontend dengan Backend.
  - Menyusun dokumen.
- 13520092 / Vieri Mansyl
  - Mendesain UI.
  - Melakukan connecting antara Frontend dengan Backend.
- 13520113 / Brianaldo Phandiarta
  - Mendesain UI.
  - Melakukan connecting antara Frontend dengan Backend.
- 13520131 / Steven
  - Mengimplementasikan Board, Deck, Hand, dan kelas-kelas terkait.
  - Melakukan connecting antara Frontend dengan Backend.
  - Menyusun dokumen.
  - Mengimplementasikan Unit Testing
- 13520167 / Aldwin Hardi Swastia
  - Mendesain UI.
  - Melakukan connecting antara Frontend dengan Backend.
  - Menyusun dokumen.