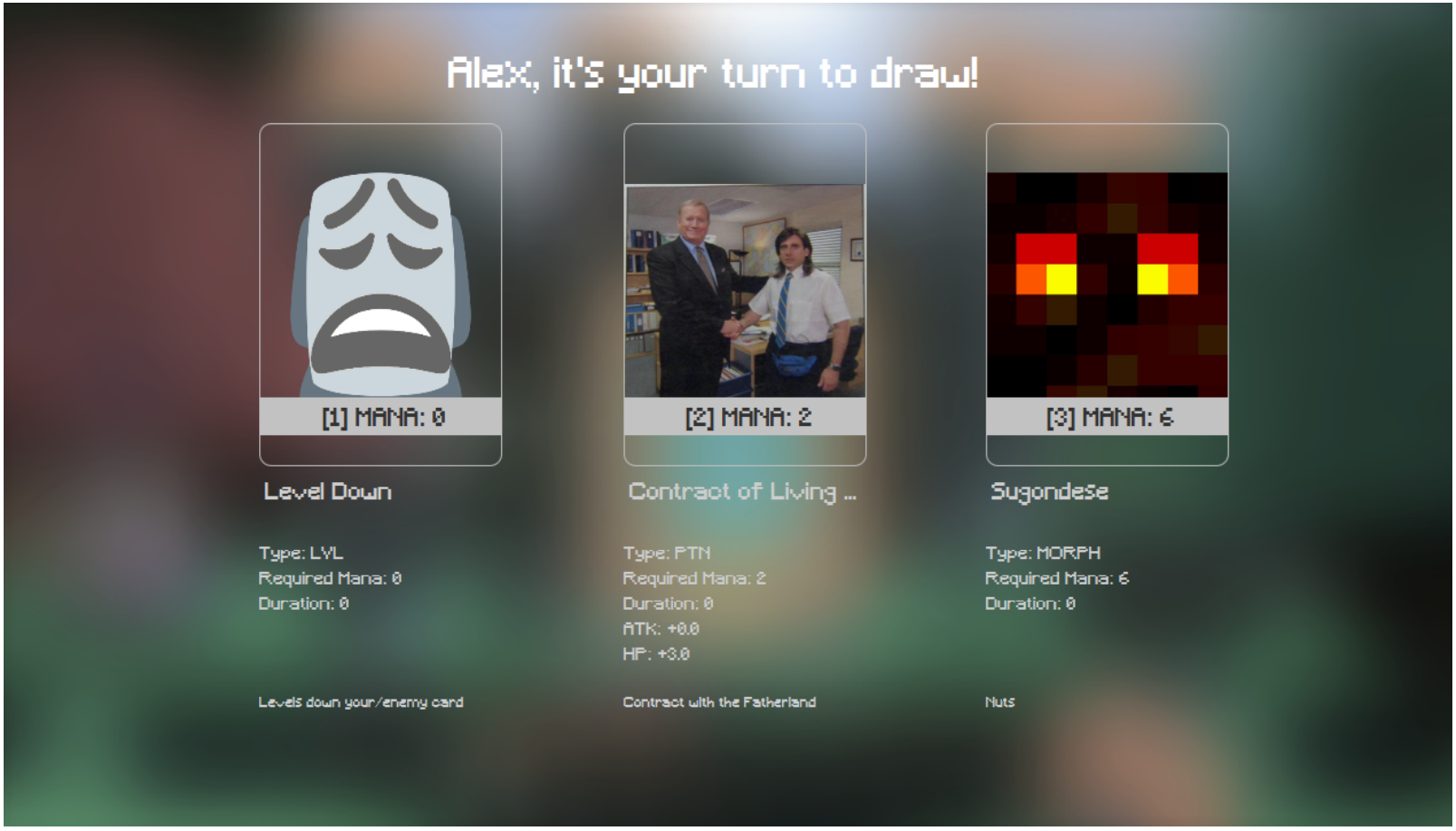
Kelas : 01
Kelompok : 06
1. 13520013 / Ilham Prasetyo Wibowo
2. 13520052 / Gregorius Moses Marevson
3. 13520103 / Amar Fadil
4. 13520124 / Owen Christian Wijaya
5. 13520139 / Fachry Dennis Heraldi
Asisten Pembimbing : Faris Rizki Ekananda
Link repository: https://github.com/clumsyyyy/TubesOOP2

1. Deskripsi Umum Aplikasi

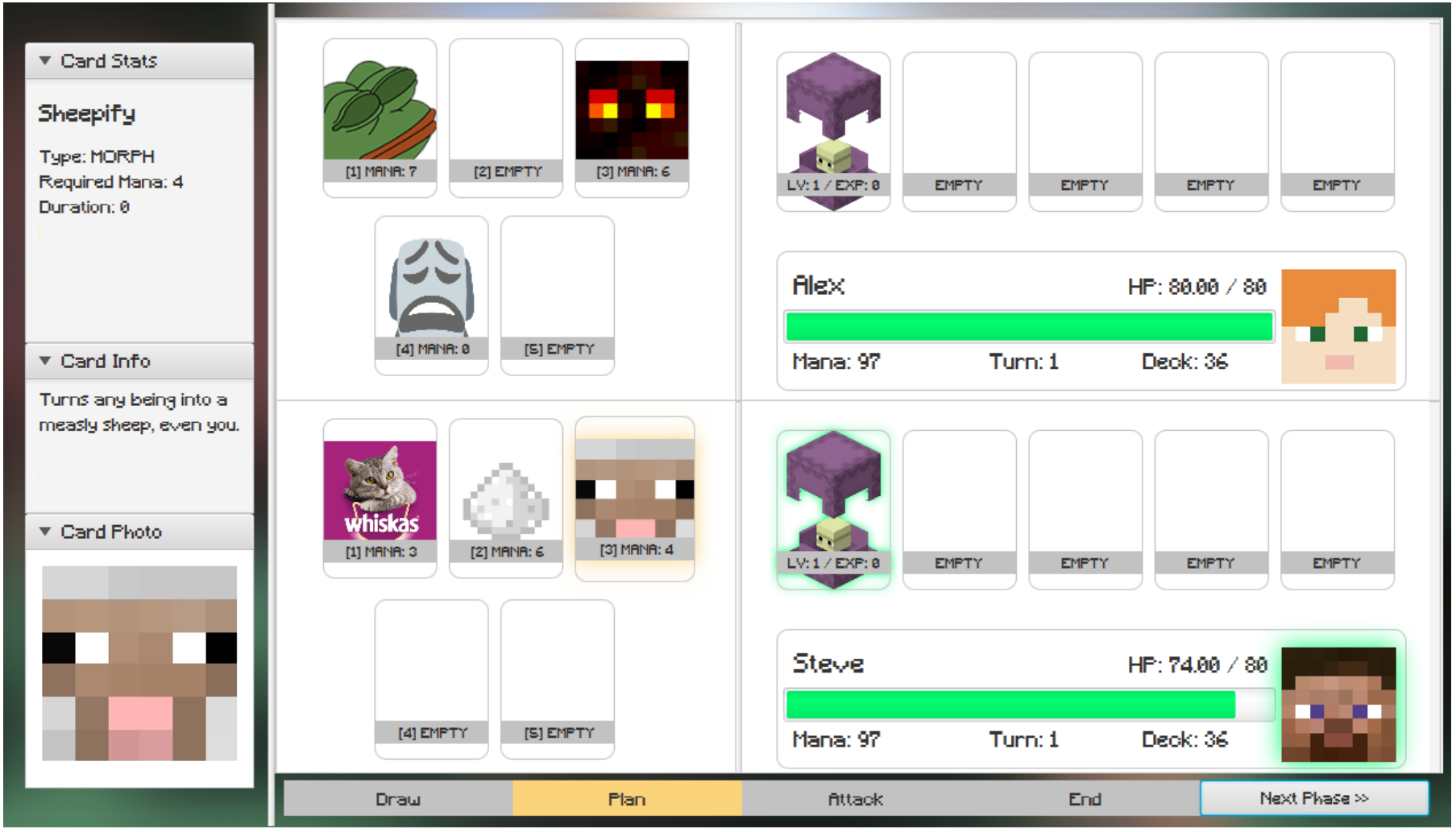
Minecraft: Aether Realm Wars adalah game kartu *turn based* untuk 2 pemain. Pemain bermain secara bergantian pada 1 layar yang sama dengan tujuan menghabiskan *health points* (HP) pemain lainnya. Dalam permainan ini terdapat empat fase dalam satu *turn* pemain: (1) draw, yaitu fase memilih satu dari tiga kartu yang diambil dari deck, (2) plan, fase melakukan konfigurasi terhadap kartu yang dimiliki, (3) attack phase, (4) end, fase untuk mengakhiri *turn* pemain. Terdapat dua jenis kartu secara umum, yaitu kartu character dan kartu spell. Kartu character adalah kartu yang dapat melakukan aksi untuk menyerang lawan sedangkan kartu spell adalah kartu yang sifatnya memberikan keunikan yang diterapkan pada kartu character. Permainan akan berakhir dan pemain dinyatakan kalah ketika HP atau kartu pada deck pemain tersebut lebih dahulu habis.

Game ini dibangun dan dikembangkan dalam bahasa Java menggunakan IDE IntellIJ. Untuk melakukan *build* aplikasi, digunakan gradle agar program dapat di-*build* secara otomatis setiap kali melakukan perubahan pada source code. Untuk keperluan pembuatan GUI digunakan JavaFX dengan bantuan SceneBuilder untuk memudahkan desain GUI. Game ini merupakan pemenuhan Tugas Besar 2 IF2210 Pemrograman Berorientasi Objek sehingga perlu diterapkan konsep-konsep yang telah diajarkan. Adapun lima konsep yang diterapkan pada game ini adalah: (1) Inheritance, (2) Interface, (3) Java API, (4) SOLID, dan (5) Design Pattern. Berikut ini adalah screenshot dari aplikasi yang telah dibuat.

Tampilan awal ketika aplikasi dijalankan



Tampilan ketika draw phase

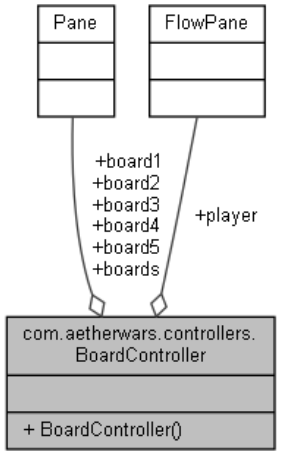


Tampilan utama permainan

2. Diagram Kelas

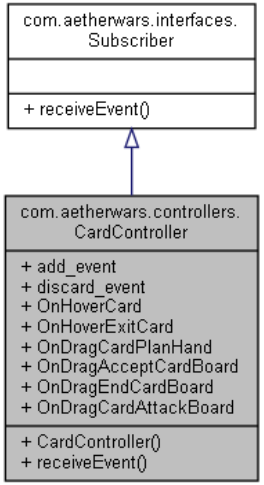
2.1. Package Controller

2.1.1. Kelas BoardController

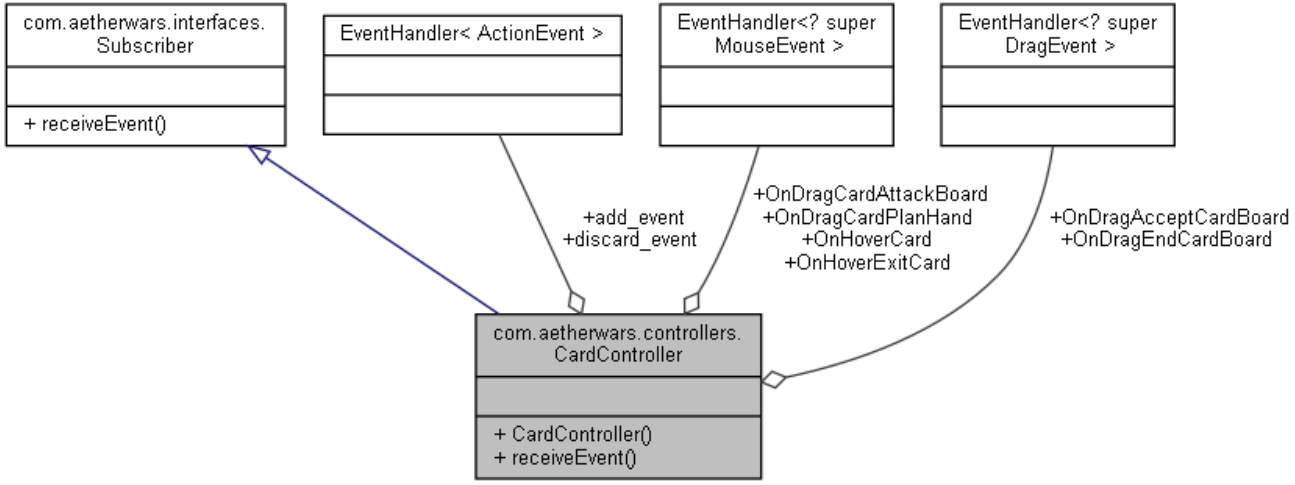


Collaboration Diagram Kelas BoardController

2.1.2. Kelas CardController

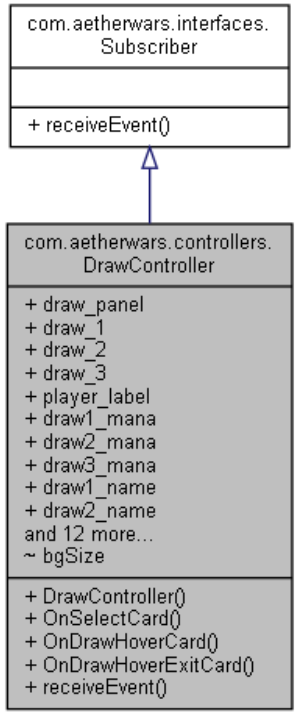


Inheritance Diagram Kelas CardController

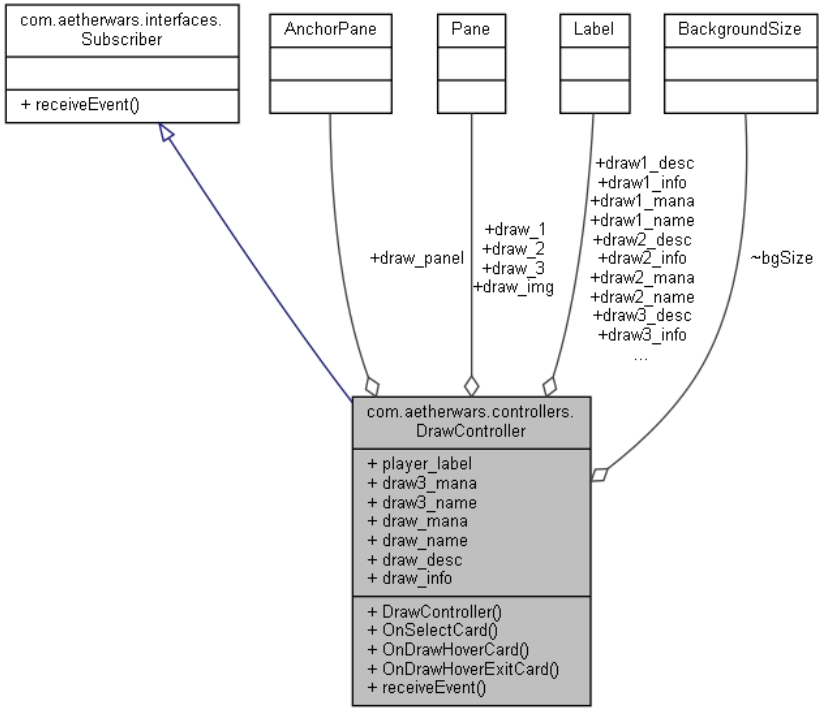


Collaboration Diagram Kelas CardController

2.1.3. Kelas DrawController

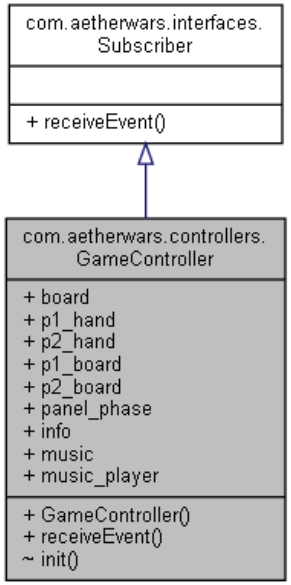


Inheritance Diagram Kelas DrawController

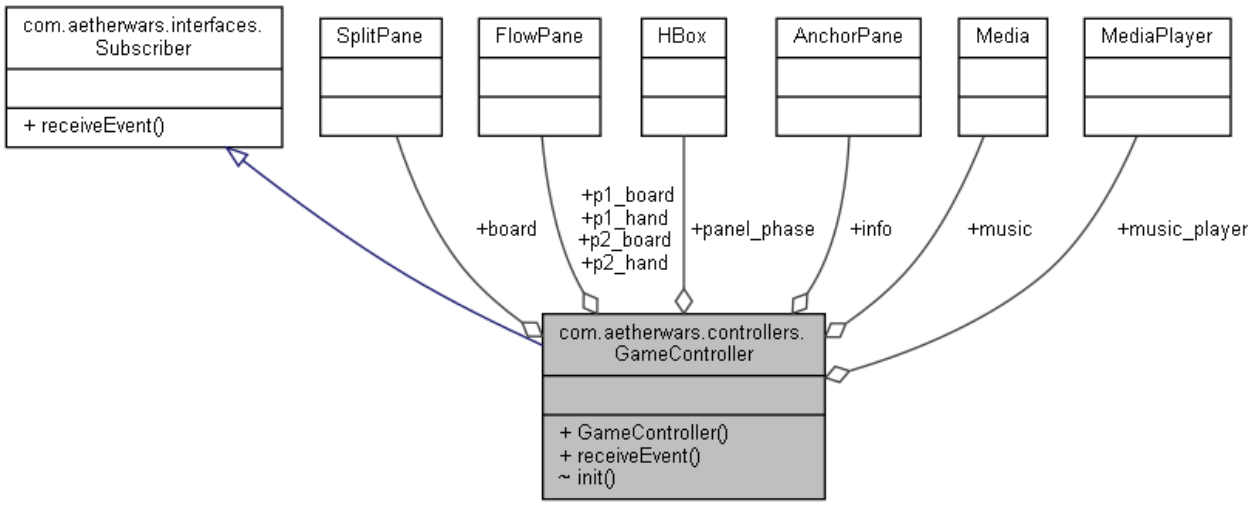


Collaboration Diagram Kelas DrawController

2.1.4. Kelas GameController

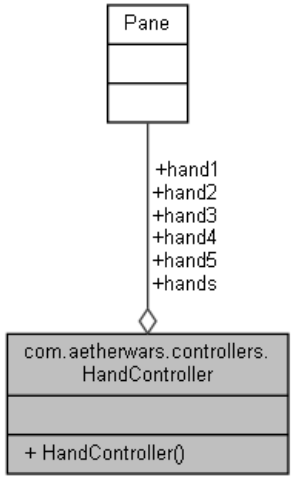


Inheritance Diagram Kelas GameController



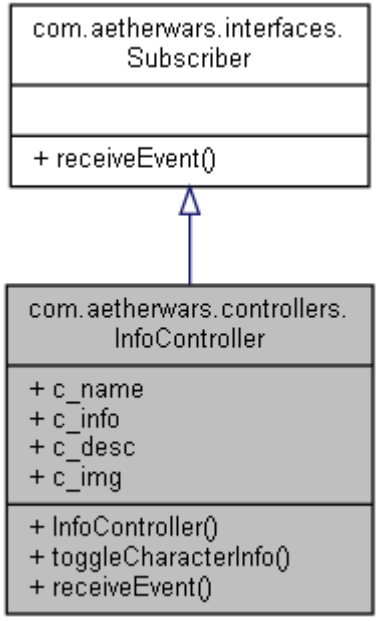
Collaboration Diagram Kelas GameController

2.1.5. Kelas HandController

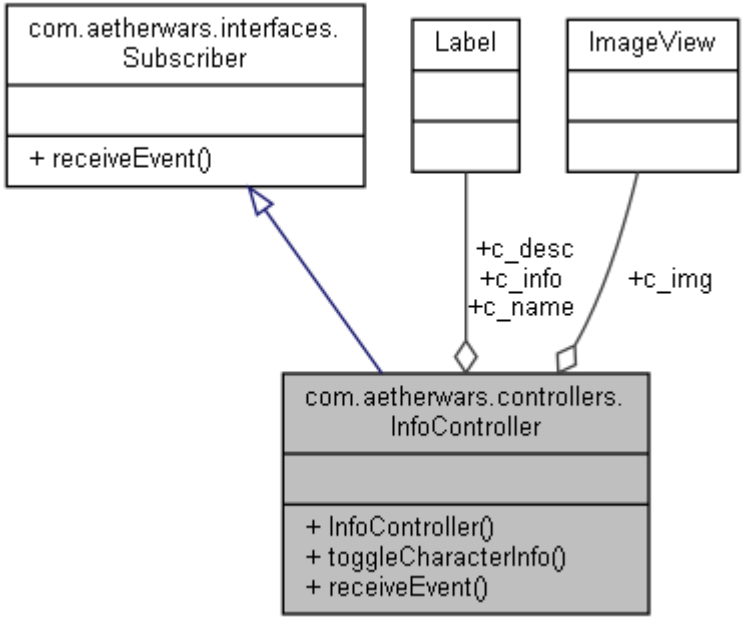


Collaboration Diagram Kelas HandController

2.1.6. Kelas InfoController

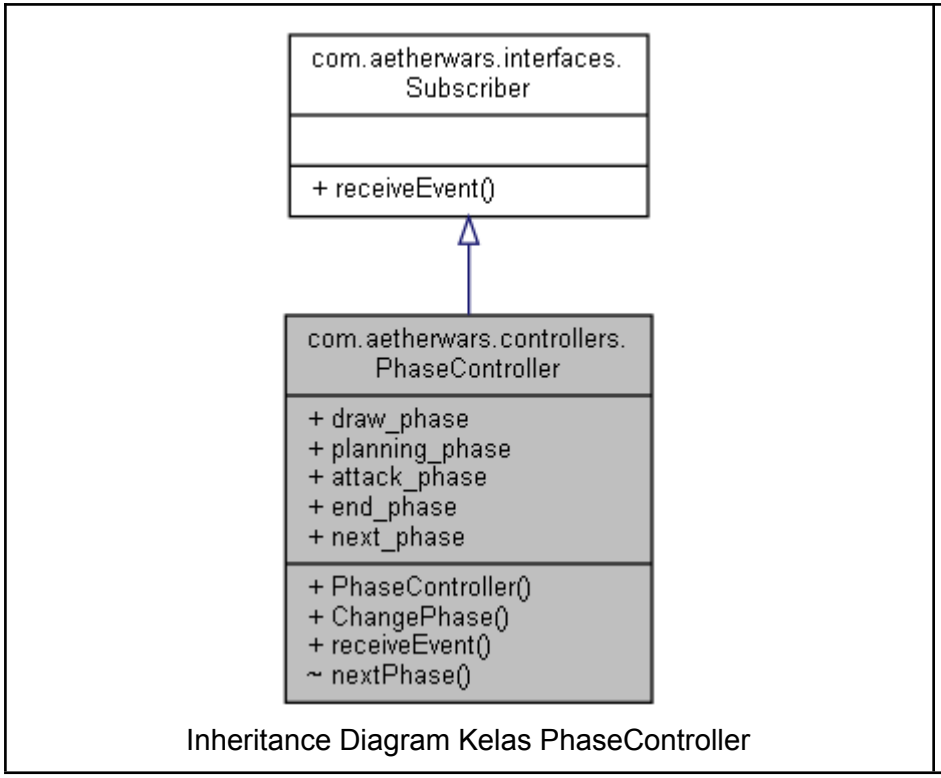


Inheritance Diagram Kelas InfoController

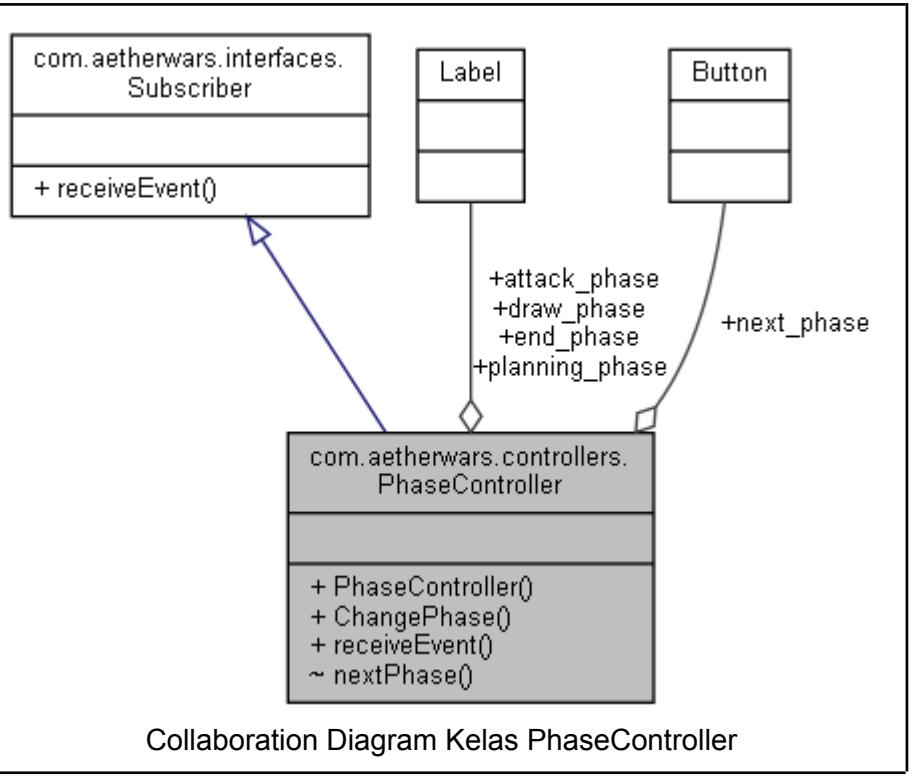


Collaboration Diagram Kelas InfoController

2.1.7. Kelas PhaseController

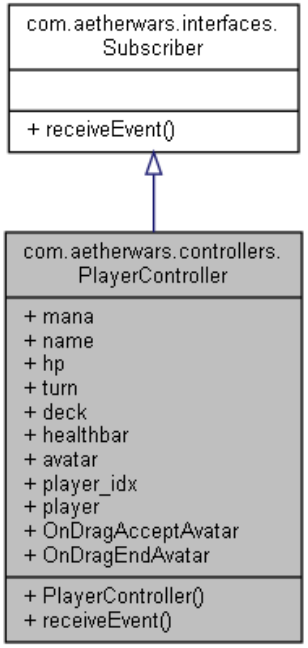


Inheritance Diagram Kelas PhaseController

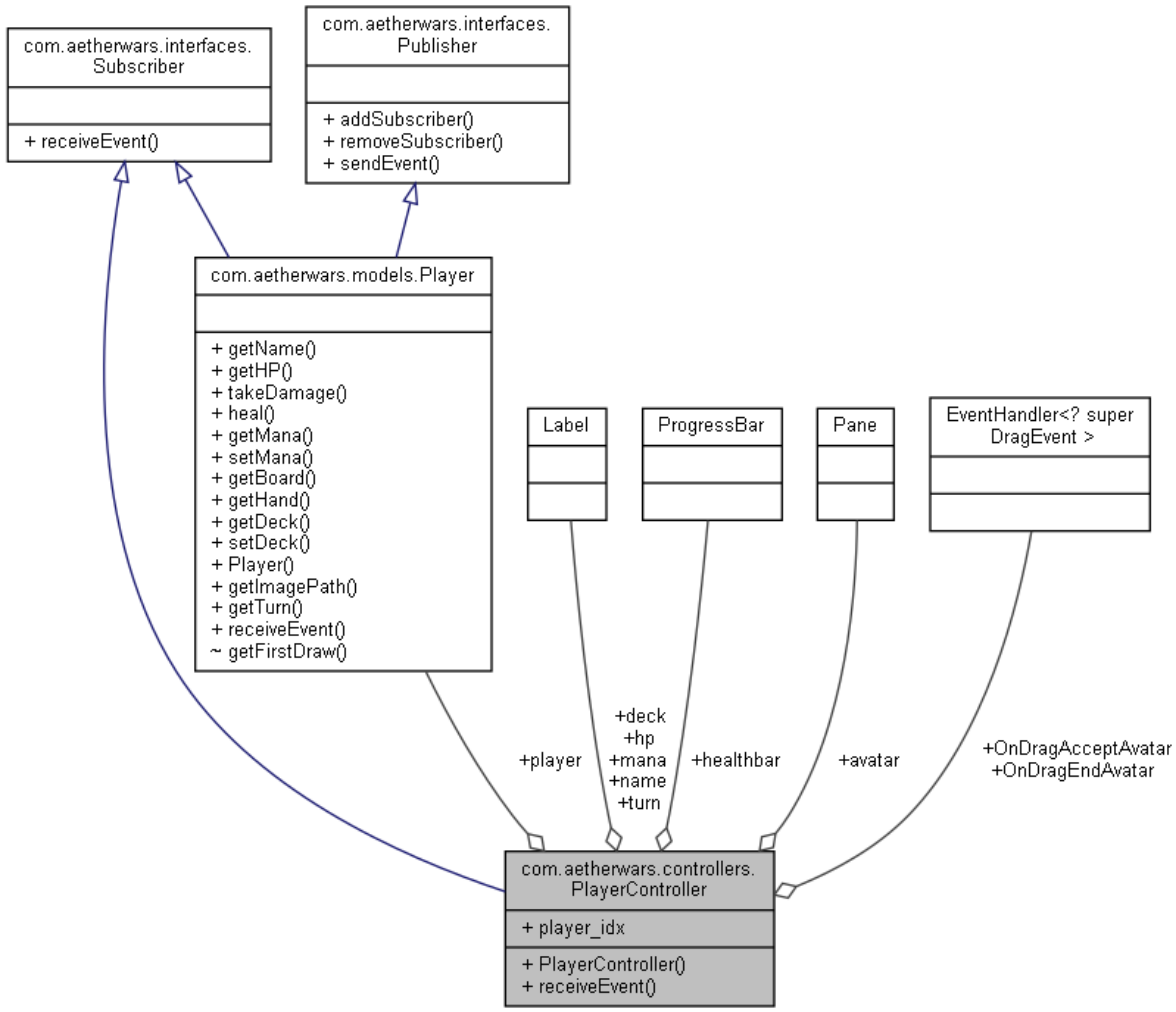


Collaboration Diagram Kelas PhaseController

2.1.8. Kelas PlayerController

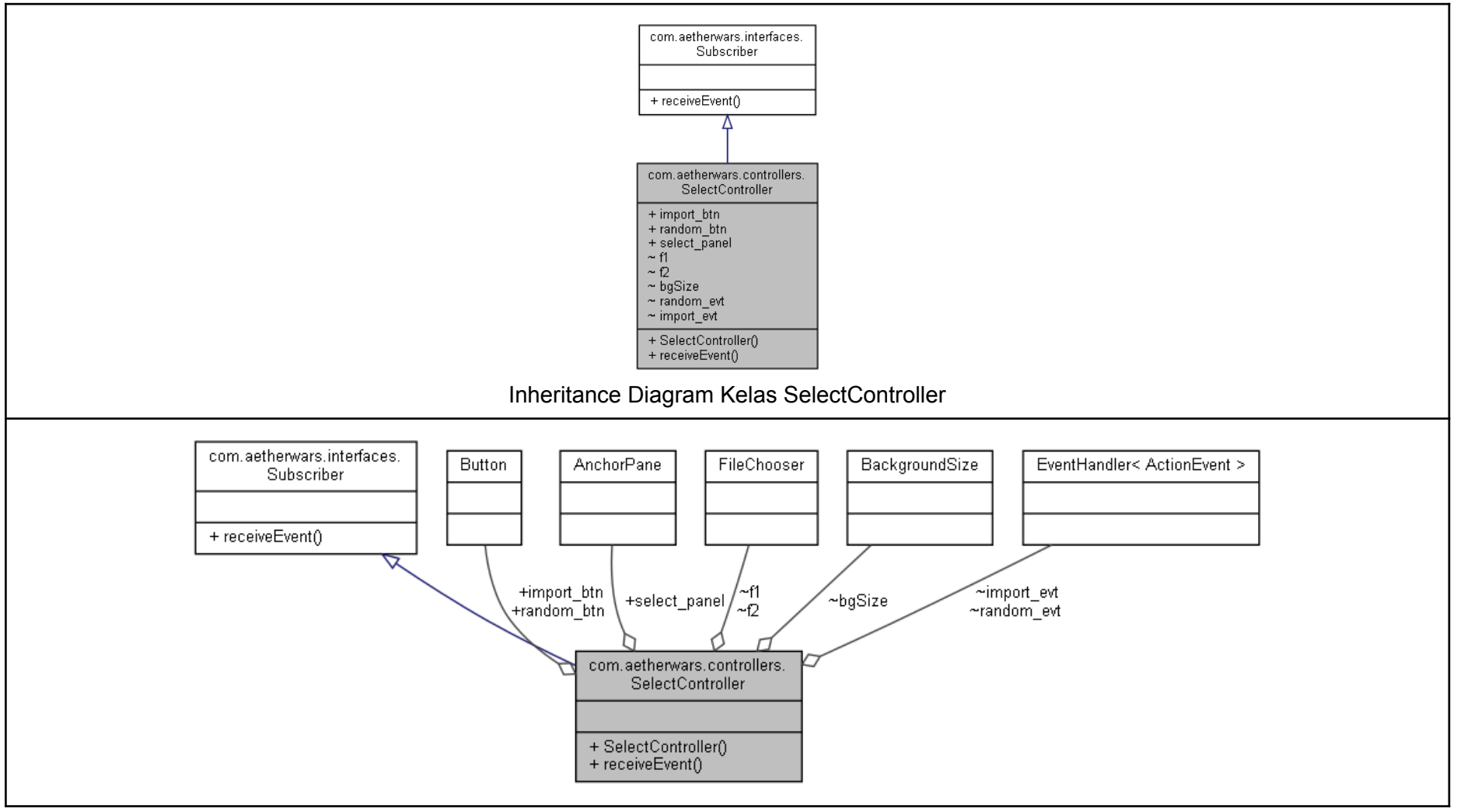


Inheritance Diagram Kelas PlayerController



Collaboration Diagram Kelas PlayerController

2.1.9. Kelas SelectController

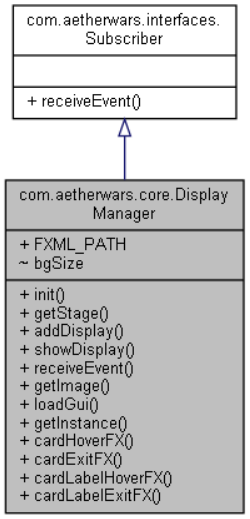


Inheritance Diagram Kelas SelectController

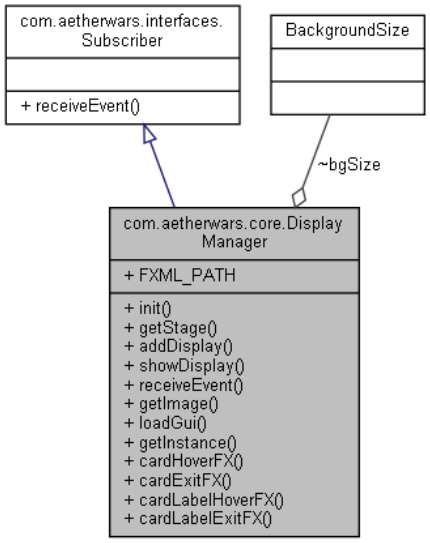
Collaboration Diagram Kelas SelectController

2.2. Package Core

2.2.1. Kelas DisplayManager

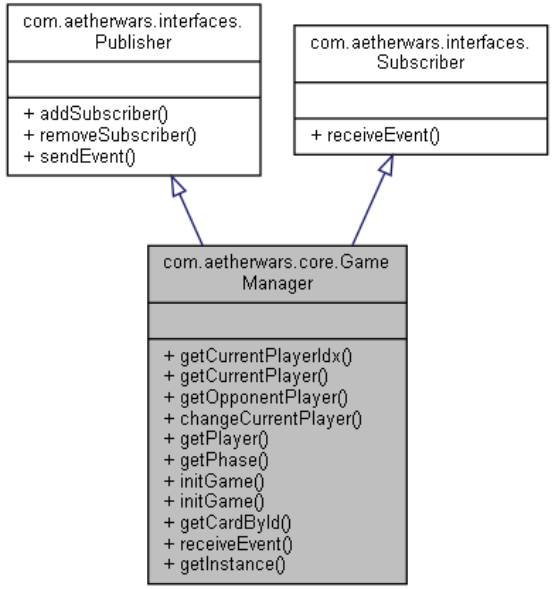


Inheritance diagram kelas DisplayManager

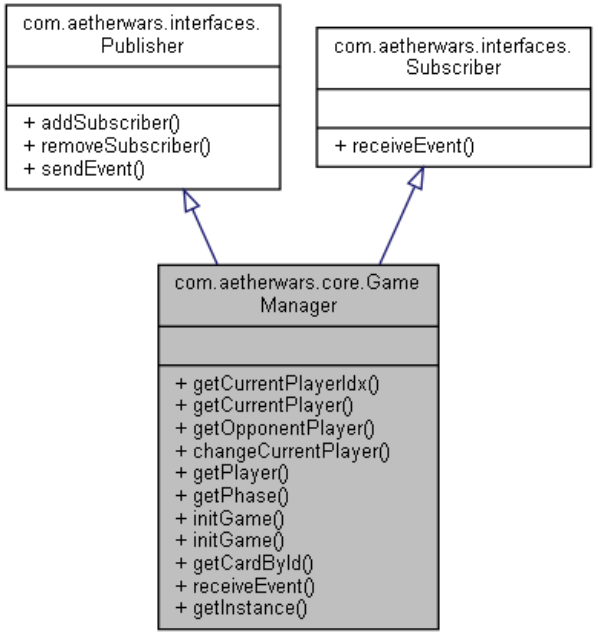


Collaboration diagram kelas DisplayManager

2.2.2. Kelas GameManager

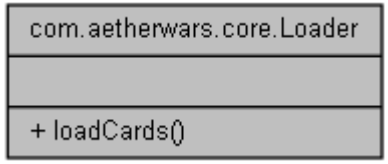


Inheritance diagram kelas GameManager



Collaboration diagram kelas GameManager

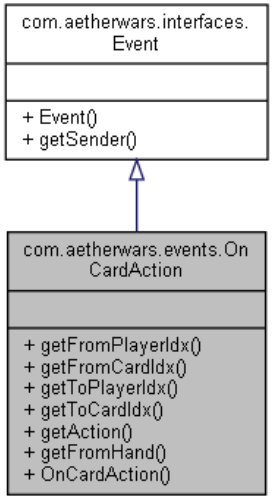
2.2.3. Kelas Loader



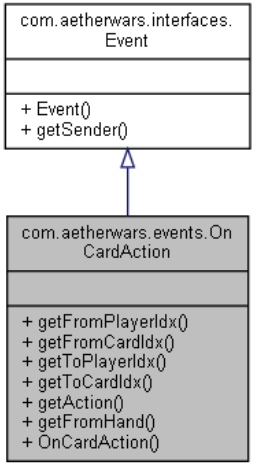
Collaboration diagram kelas Loader

2.3. Package Events

2.3.1. Kelas OnCardAction

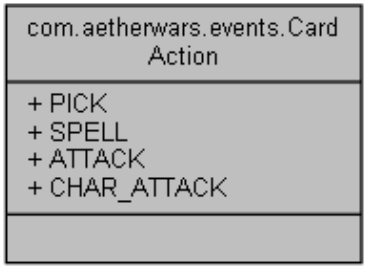


Inheritance diagram kelas OnCardAction



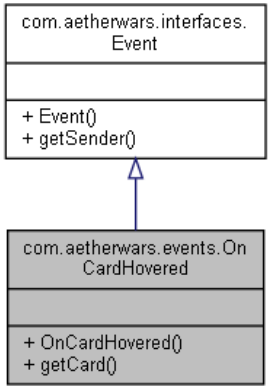
Collaboration diagram kelas OnCardAction

2.3.2. Kelas CardAction

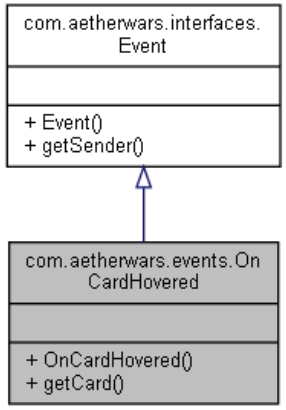


Collaboration diagram kelas CardAction

2.3.3. Kelas OnCardHovered

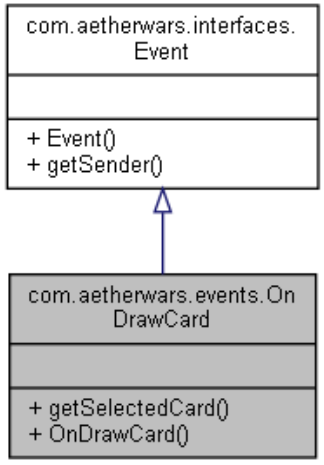


Inheritance diagram kelas OnCardHovered

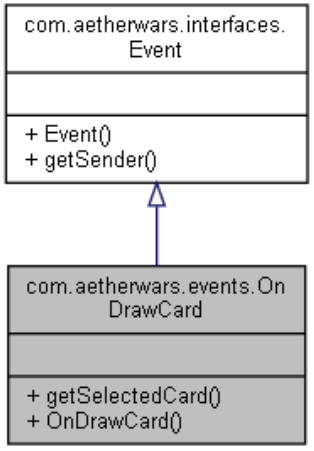


Collaboration diagram kelas OnCardHovered

2.3.4. Kelas OnDrawCard

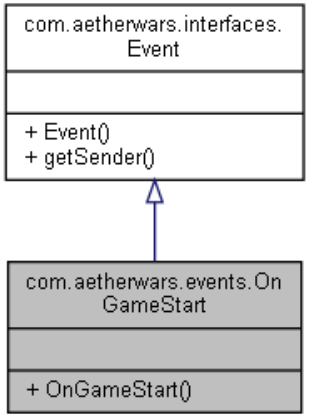


Inheritance diagram kelas OnDrawCard

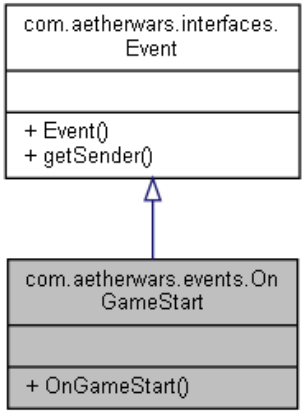


Collaboration diagram kelas OnDrawCard

2.3.5. Kelas OnGameStart

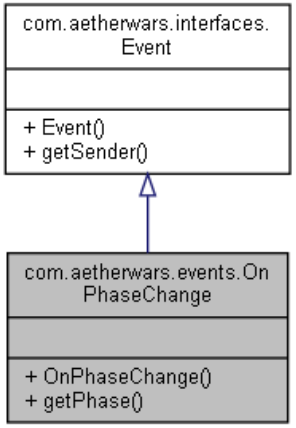


Inheritance diagram kelas OnGameStart

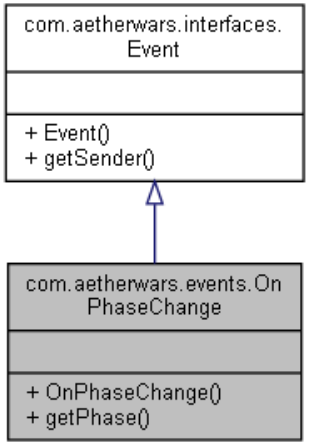


Collaboration diagram kelas OnGameStart

2.3.6. Kelas OnPhaseChange



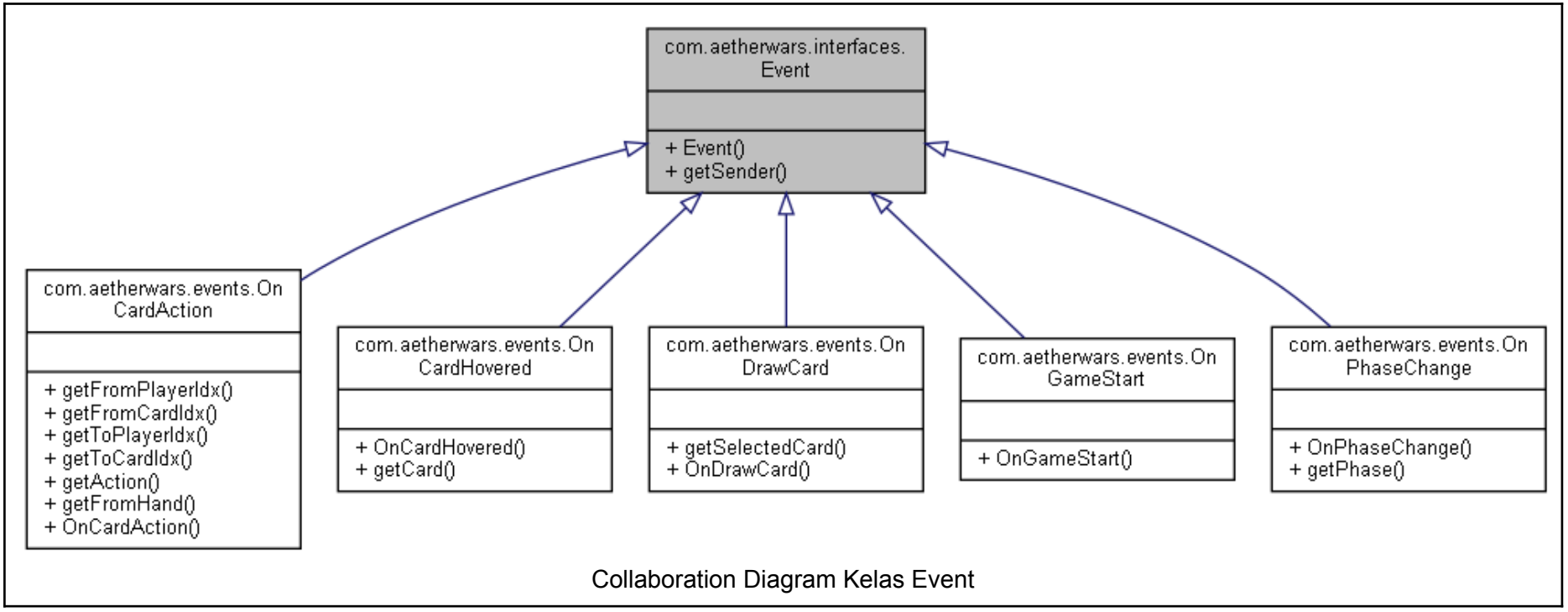
Inheritance diagram kelas OnPhaseChange



Collaboration diagram kelas OnPhaseChange

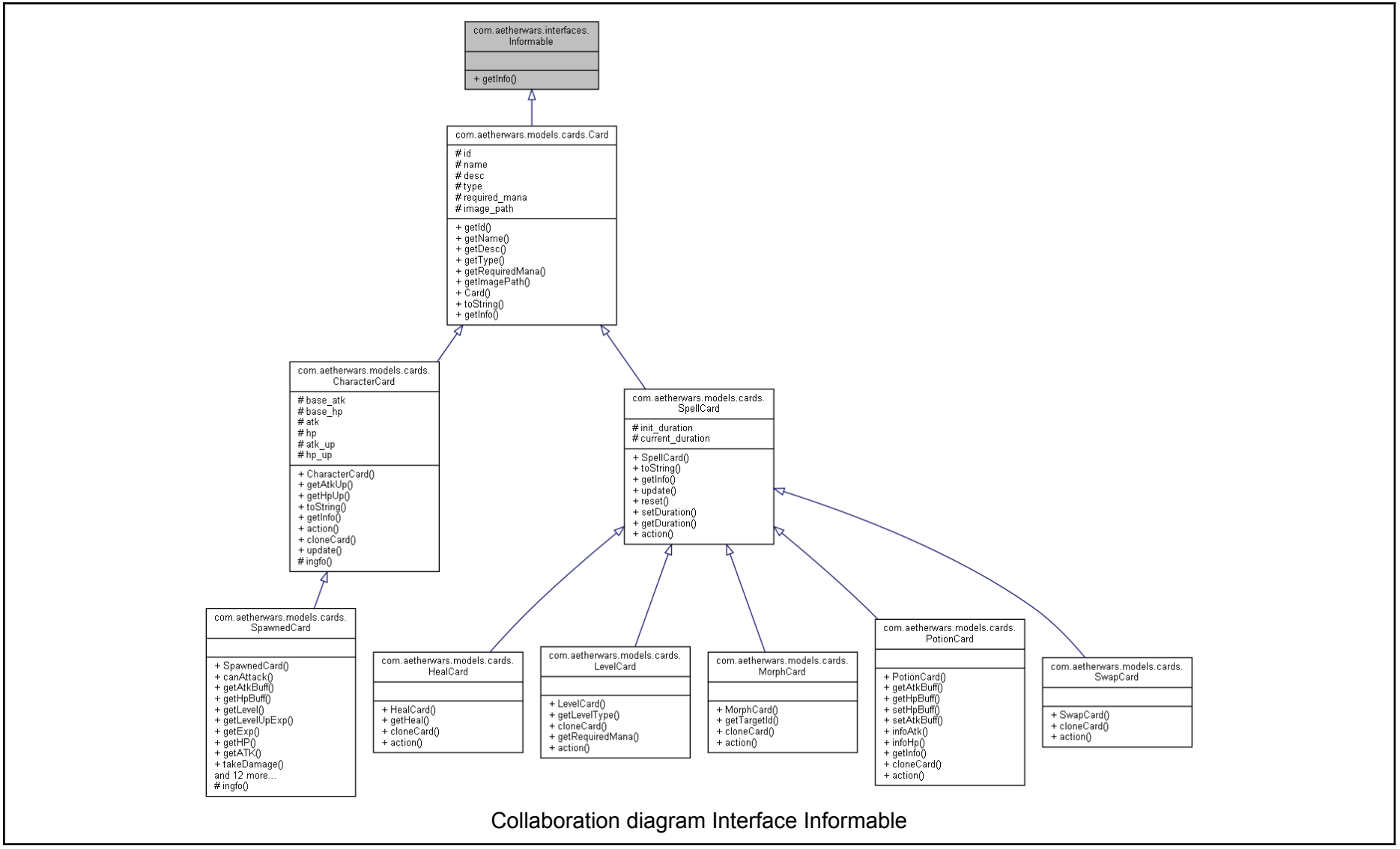
2.4. Package Interfaces

2.4.1. Kelas Event



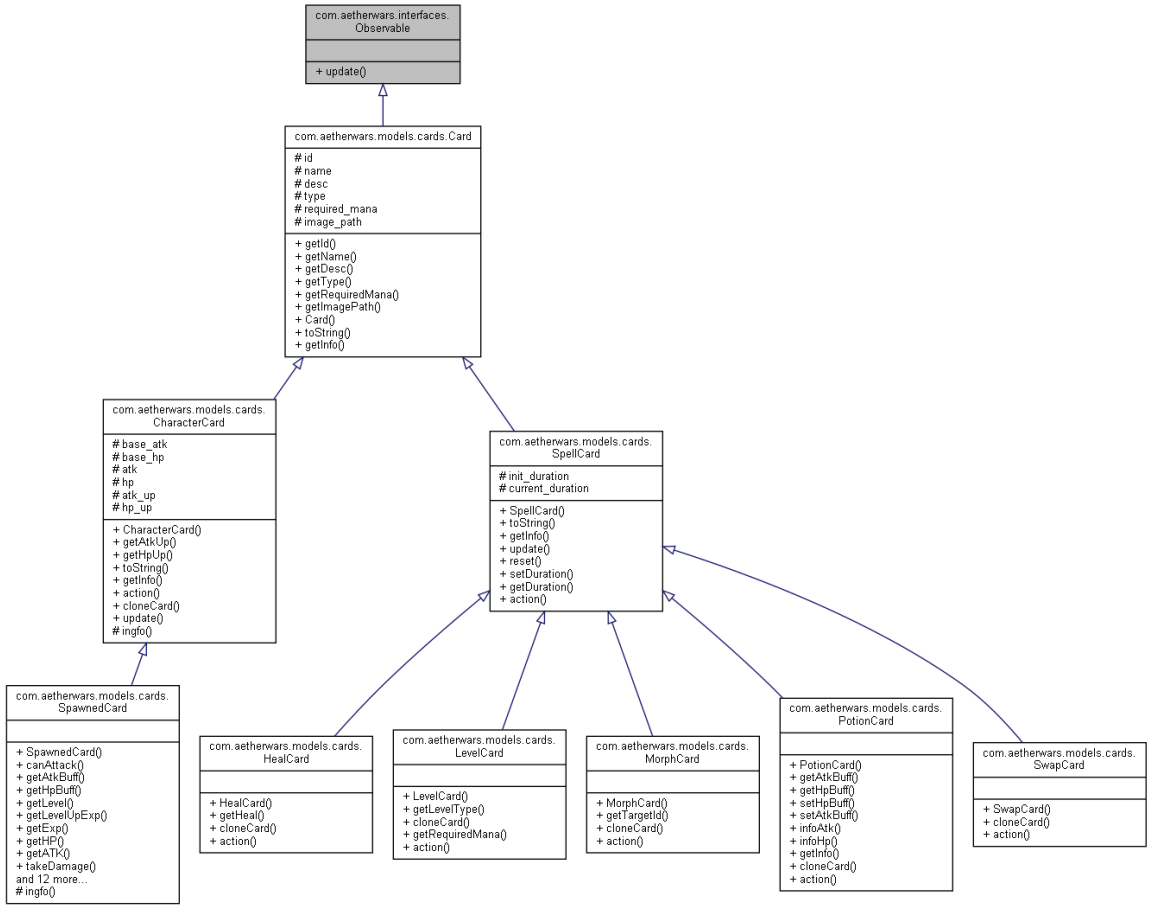
Collaboration Diagram Kelas Event

2.4.2. Interface Informable



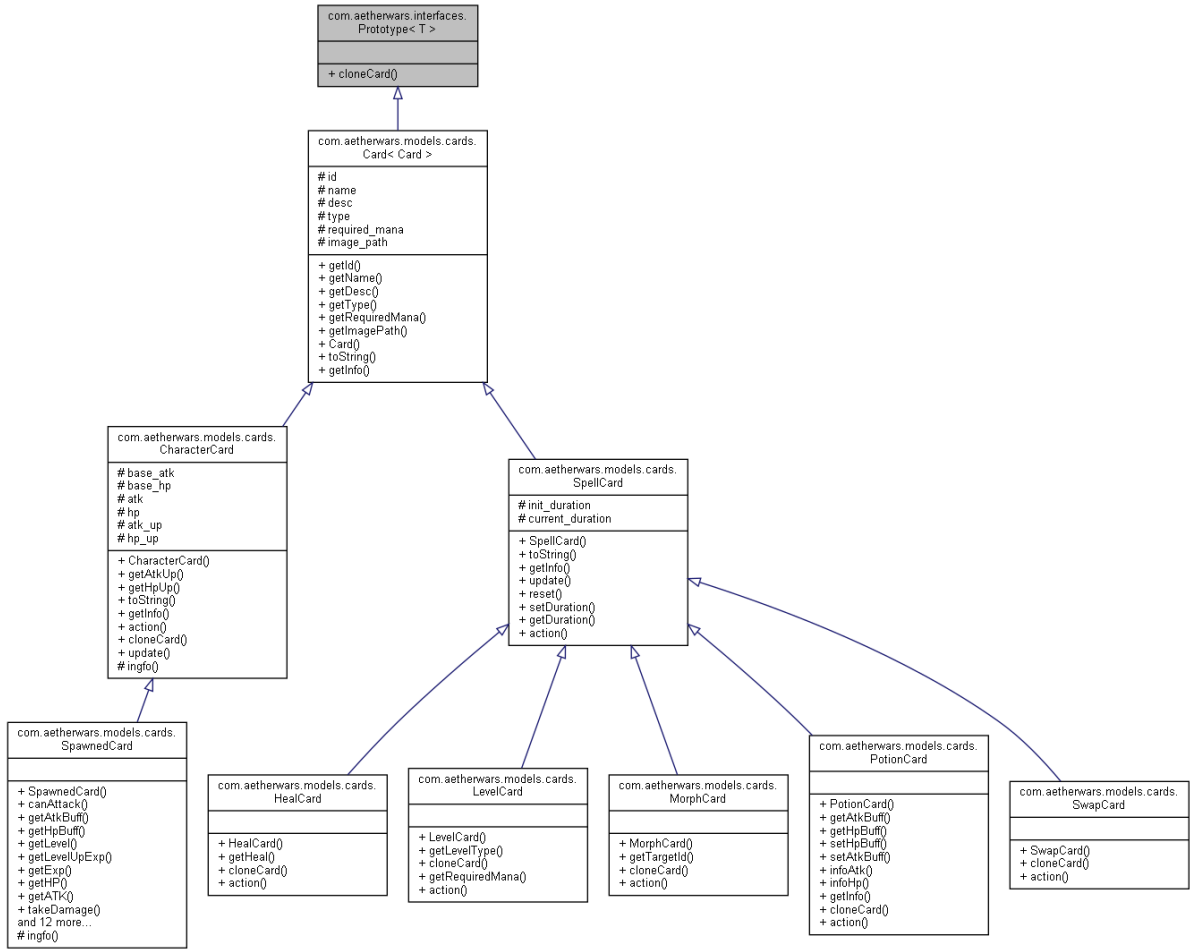
Collaboration diagram Interface Informable

2.4.3. Interface Observable



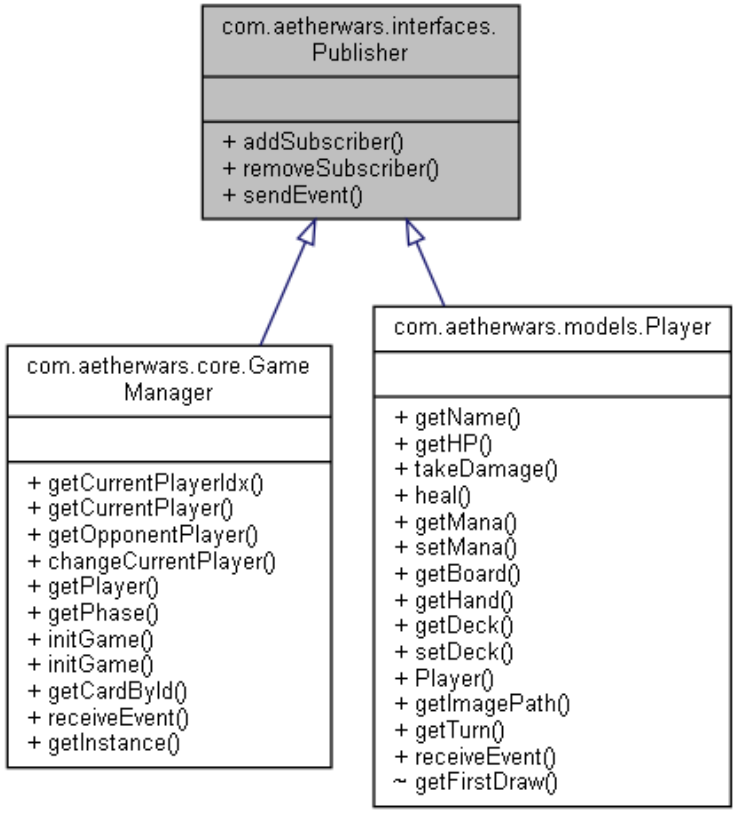
Collaboration Diagram Interface Observable

2.4.4. Interface Prototype



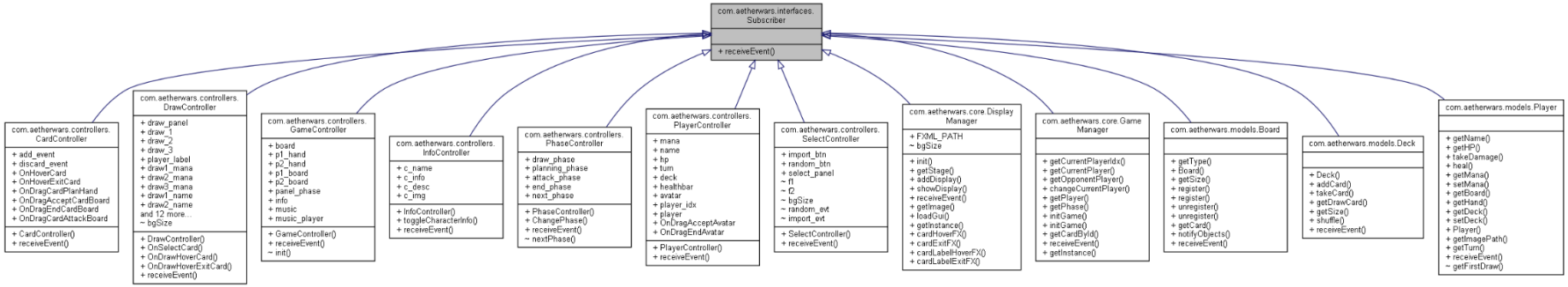
Collaboration Diagram Interface Prototype

2.4.5. Kelas Publisher



Collaboration Diagram Kelas Publisher

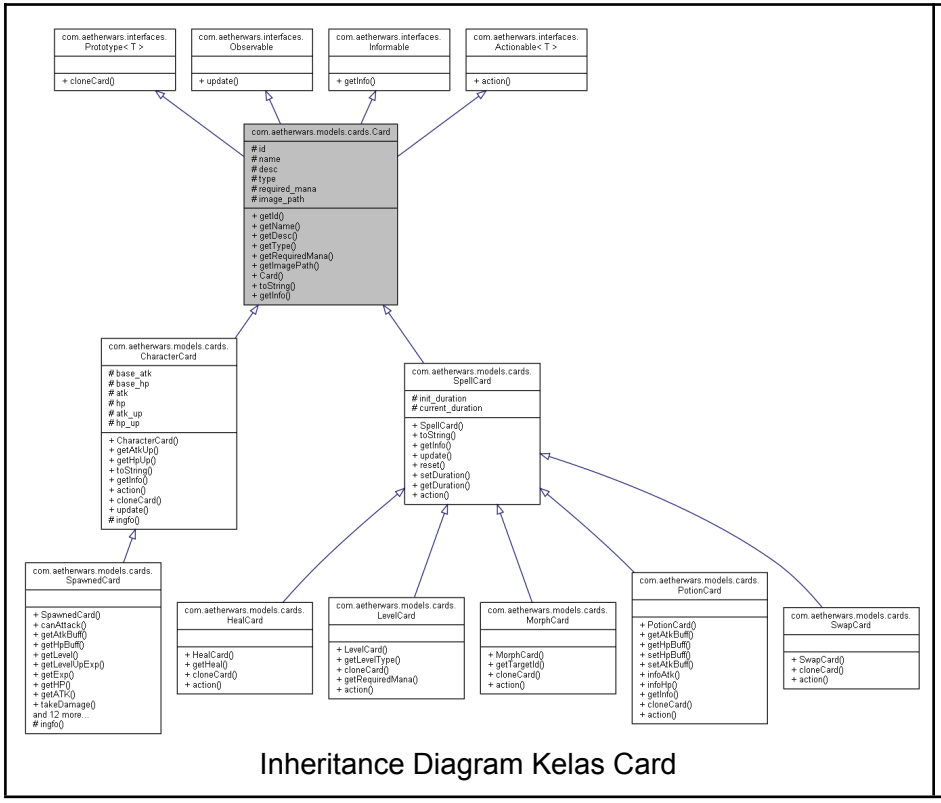
2.4.6. Interface Subscriber



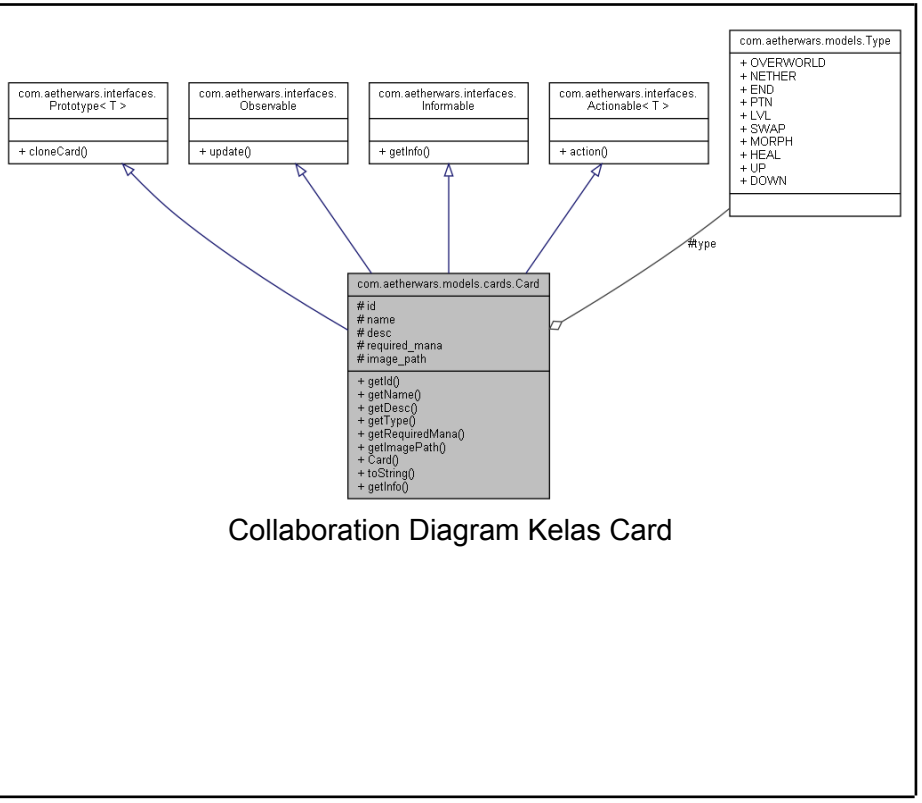
Collaboration Diagram Interface Subscriber

2.5. Package Models

2.5.1. Kelas Card

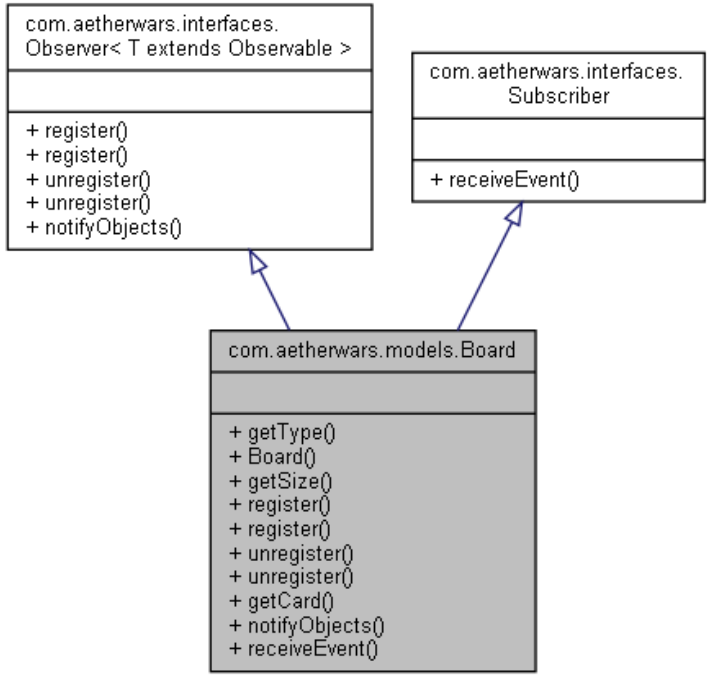


Inheritance Diagram Kelas Card



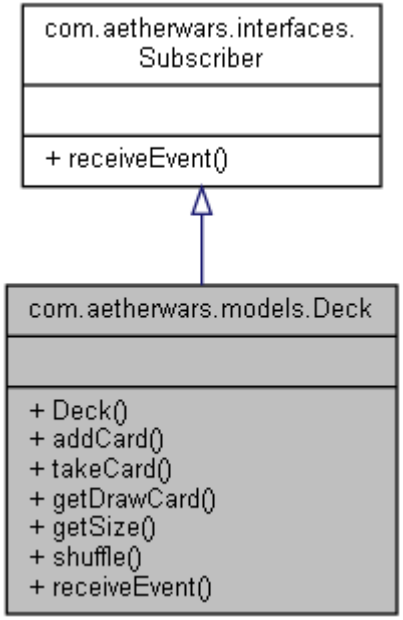
Collaboration Diagram Kelas Card

2.5.2. Kelas Board



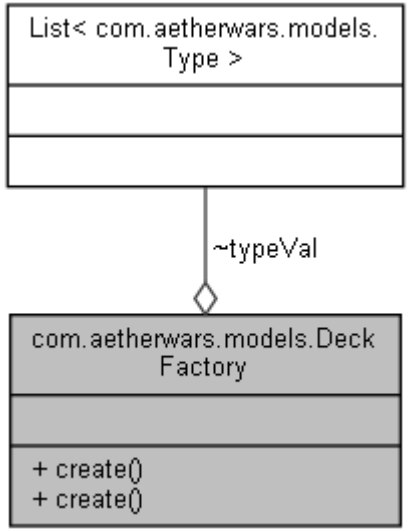
Collaboration Diagram Kelas Board

2.5.3. Kelas Deck



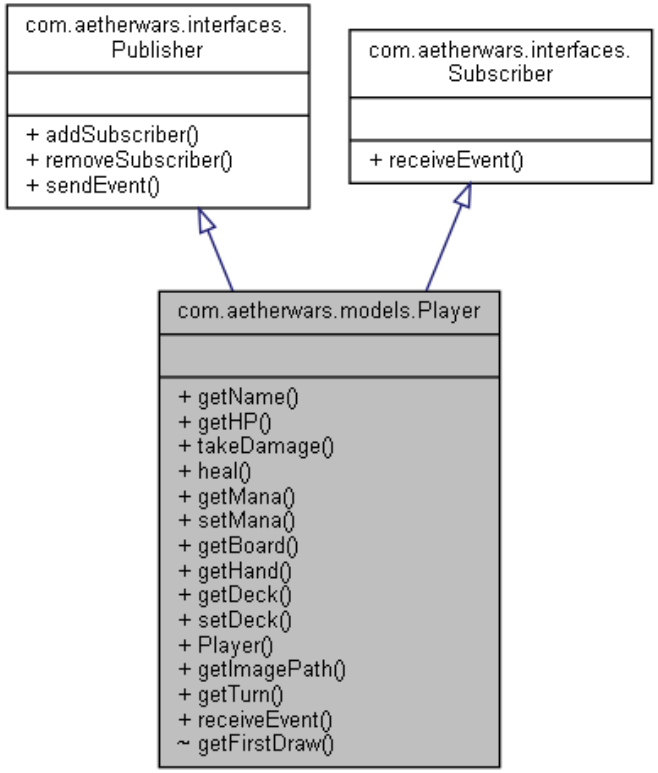
Collaboration Diagram Kelas Deck

2.5.4. Kelas DeckFactory



Collaboration Diagram Kelas DeckFactory

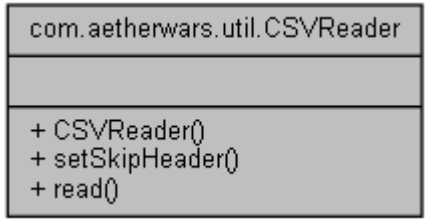
2.5.5. Kelas Player



Collaboration Diagram Kelas Player

2.6. Package Util

2.6.1. Kelas CSVReader



Collaboration Diagram Kelas CSVReader

3. Konsep OOP

3.1. Inheritance

Inheritance adalah suatu mekanisme membasiskan suatu kelas terhadap implementasi kelas lain. Dengan kata lain, inheritance adalah proses penurunan (pewarisan) kelas baru (disebut subclass) dari kelas yang sudah tersedia (disebut superclass). Konsep inheritance yang digunakan pada aplikasi terdapat pada :

- Kelas Player (aetherwars/models/Player.java) dan kelas GameManager (aetherwars/core/GameManager.java) merupakan subclass dari kelas Publisher (aetherwars/interfaces/Publisher.java)
- Kelas SpawnedCard merupakan anak dari kelas CharacterCard, kelas CharacterCard merupakan anak dari kelas Card (semua class terdapat pada folder aetherwars/models/cards)
- Kelas SwapCard, PotionCard, dan MorphCard merupakan subclass dari kelas SpellCard, dan SpellCard merupakan anak dari kelas Card (semua class terletak pada folder aetherwars/models/cards)
- Kelas Event (src/main/java/com/aetherwars/interfaces) merupakan superclass dari kelas OnCardAction, OnCardHovered, OnDrawCard, OnGameStart, OnPhaseChange (semua subclass terletak pada folder aetherwars/events)

- Interface Observer(aetherwars/interfaces/Observer.java) merupakan subinterface dari interface Observable (aetherwars/interfaces/Observable.java)

3.2. Interface

Konsep interface merupakan sebuah mekanisme yang disediakan java yang memungkinkan sebuah kelas berbagi konstanta atau bentuk metode yang dapat digunakan oleh beberapa kelas lainnya. Penggunaan interface ini memiliki banyak tujuan, berkaitan dengan konsep SOLID dan Design Pattern. Pada aplikasi, penggunaan konsep interface terdapat dalam satu folder interface (src/main/java/com/aetherwars/interface) yang berisi beberapa interface sebagai berikut.

- Event
- Informable
- Observer
- Observable
- Prototype
- Publisher
- Subscriber

3.3. Java API

API merupakan singkatan dari Application Programming Interface. Java API adalah kumpulan komponen-komponen atau kode yang sudah ada pada kelas java. Komponen-komponen tersebut dapat digunakan untuk berbagai keperluan seperti kemampuan untuk memproses file, menangani objek, string dan angka.

- File dan FileReader

  Penggunaan java.io.File terdapat pada kelas Loader (src/main/java/com/aetherwars/core/Loader.java), sedangkan penggunaan FileReader terdapat pada kelas CSVReader (src/main/java/com/aetherwars/util/CSVReader.java)

- List dan ArrayList

Penggunaan List yang menyimpan objek Card pada kelas Deck (src/main/java/com/aetherwars/models/Deck.java)

- Collection

  Penggunaan metode shuffle pada Collection untuk mengacak List of Card pada kelas Deck (src/main/java/com/aetherwars/models/Deck.java).

- Map dan HashMap

  Penggunaan Map dan HashMap pada kelas DisplayManager (src/main/java/com/aetherwars/core/DisplayManager.java).

- Random

  Penggunaan random pada kelas DrawController (src/main/java/com/aetherwars/controllers/DrawController.java).

3.4. SOLID

a. Single Responsibility

Pada aplikasi ini, telah dikelompokkan kelas-kelas sesuai dengan klasifikasinya sehingga setiap kelas punya tanggung jawabnya masing-masing. Sebagai contoh, pada kelompok events (aetherwars/events), walaupun semua kelas memiliki kesamaan tanggung jawab sebagai event listener, namun setiap kelas tersebut bertanggung jawab sebagai event listener untuk kelas yang berbeda-beda, misal, Kelas OnDrawCard (events/OnDrawCard.java), bertanggung jawab sebagai event listener untuk Kelas yang berhubungan dengan Card (models/Card.java), sedangkan Kelas OnPhaseChange (events/OnPhaseChange.java), bertanggung jawab sebagai event listener untuk Kelas Phase (models/Phase.java).

b. Open-Closed

Untuk memastikan agar tidak terjadinya *clash* ketika ditambahkannya fitur baru setiap diimplementasikan, pada aplikasi ini digunakan interface yang dikelompokkan pada folder interfaces (aetherwars/interfaces). Dengan demikian setiap kelas yang mengimplementasikan interface memiliki kontrak dan terikat sehingga jika ada perubahan pada interface, maka kelas yang mengimplementasikan interface tersebut perlu menambahkan perubahan tersebut. Selain mencegah *clash*, penerapan prinsip Open-Closed ini juga memastikan kode selalu konsisten walaupun terjadi perubahan seperti penambahan fitur.

c. Liskov Substitution Principle

Implementasi Liskov terdapat pada interface Actionable. Sebuah interface Actionable mendefinisikan sebuah action yang harus diambil sebuah kelas. Jadi, alih-alih terdapat sebuah board yang menentukan apa yang harus dilakukan apabila menerima sebuah kartu, kartu tersebut telah mendefinisikan apa yang harus dilakukan. Dengan demkian, parent class Card dapat diganti menjadi subtypes saja sehingga hanya perlu satu fungsi yang menerima satu kartu dan setiap kartu mempunyai actinon yang unik.

d. Interface Segregation

Kelas-kelas pada kelompok models (aetherwars/models) mengimplementasikan beberapa interface yang sudah disegregasi pada kelompok interfaces (aetherwars/interfaces). Sebagai contoh: Kelas Board (models/Board.java) mengimplementasikan interface Observer (interfaces/Observer.java) dan Subscriber(interfaces/Subscriber.java), sementara itu Kelas Card (models/Card.java) mengimplementasikan interface Prototype (interfaces/Prototype.java), Observable(interfaces/Observable.java), dan Informable(interfaces/Informable.java).

e. Dependency Inversion

Penggunaan interface ditujukan untuk menjamin Kelas yang memiliki kompleksitas tinggi tidak bergantung dengan kelas yang memiliki kompleksitas rendah. Pada aplikasi ini kelas kelompok models (aetherwars/models) yang memiliki kompleksitas tinggi berkomunikasi dengan kelas kelompok events (aetherwars/events) yang memiliki kompleksitas rendah melalui interface (aetherwars/interfaces).

3.5. Design Pattern

Design pattern adalah sebuah metode untuk menyelesaikan sebuah permasalahan yang biasanya berulang pada pengembangan aplikasi. Sehingga solusi bisa ditemukan dengan suatu pola tertentu. Penggunaan design pattern pada aplikasi adalah sebagai berikut.

a. Publisher-Subscriber Pattern

Publisher-subscriber digunakan untuk komunikasi antar objek antara pengirim pesan (publisher) memberikan pesan namun tidak secara langsung pada penerima (subscriber) yang spesifik, melainkan dikelompokan dan dikategorikan sesuai dengan kelasnya. Penerapan Publisher-Subsciber Pattern ini terdapat pada Kelas GameManager (aetherwars/core/GameManager.java), Kelas Player (aetherwars/models/Player.java), dan kelompok kelas controllers (aetherwars/controllers).

b. Factory Pattern

Factory pattern digunakan untuk memisahkan antara objek yang akan diinstasiasikan dengan kelas lain. Sehingga kelas yang akan menggunakan objek tersebut hanya memanggil kelas pembuatnya (factory) saja. Penerapan factory pattern terdapat pada kelas DeckFactory (src/main/java/com/aetherwars/models/DeckFactory.java) yang memiliki metode “create” yang akan menghasilkan Deck(src/main/java/com/aetherwars/models/Deck.java).

c. Singleton

Pola singleton bertanggung jawab untuk membuat sebuah objek dari suatu kelas, dan memastikan bahwa hanya satu objek yang dibuat. Kelas yang menerapkan pola singleton menyediakan metode untuk mengakses objek yang dibuat tanpa membuat objek baru. Penerapan pola singleton terdapat pada kelas GameManager(src/main/java/com/aetherwars/core/GameManager.java). Kelas memiliki atribut ins yang merupakan instansiasi dari objek GameManager.

d. Prototype

Pola prototype digunakan untuk membuat duplikat dari sebuah objek. Penggunaan pola prototype terdapat pada kelas Card (src/main/java/com/aetherwars/models/cards/Card.java). Objek card dapat menggunakan metode clone untuk membuat objek card baru yang memiliki properti sama persis dengan objek sebelumnya.

e. Chain of Responsibility

Chain of Responsibility digunakan untuk memungkinkan penerusan pemanggilan permintaaan (request) dari satu objek ke objek lain secara berantai oleh objek yang saling meng-handle satu sama lain. Ketika menerima suatu request, handler dapat memproses request atau dapat pass request kepada handler yang lain. Penggunaan chain of responsibility terdapat pada kelompok kelas events (aetherwars/events).

f. Adapter

Adapter berguna untuk menghubungkan beberapa interface yang tidak kompatibel. Dengan pola ini tidak perlu mengubah kode yang ingin digunakan. Penggunaan adapter terdapat pada kelas SpawnedCard (aetherwars/models/cards/SpawnedCard.java).

4. Bonus Yang dikerjakan

4.1. Bonus yang diusulkan oleh spek

4.1.1. Import Deck

Pemain dapat memulai permainan menggunakan *deck* berisi kartu *random* atau *deck* dari sebuah file csv. Opsi ini terdapat di menu awal, ketika pemain dapat memasukkan file csv berisi *deck* sendiri. Konfigurasi *deck* dalam csv adalah judul *deck*, diikuti oleh ID kartu per baris. Setelah itu, kelas *loader* akan memasukkan kartu-kartu yang terdeteksi (secara umum, ID kartu karakter ada di bawah 100, sedangkan ID kartu spell ada di atas 100). Apabila tidak jadi memasukkan file, *deck* yang terbuat adalah *deck random* dengan ukuran 40.

4.1.2. Spell Baru

Menambahkan jenis spell baru yaitu HEAL. HEAL memiliki efek untuk memulihkan HP player ketika digunakan sejumlah besarnya heal pada card tersebut. Sama seperti spell card lainnya, card ini juga mewarisi Kelas SpellCard.

4.1.3. Animasi

Menambahkan efek animasi popup dan glow ketika meng-hover kartu. Kartu pada *hand* akan mempunyai *glow* berwarna kuning ketika di-hover. Apabila dipindahkan ke board, kartu yang dapat melakukan serangan akan mempunyai efek *underglow* hijau. Setelah menyerang, kartu akan mempunyai efek *underglow* merah. Selain itu, meng-*hover* kartu akan membuat ukuran kartu membesar sehingga tampak jelas kartu mana yang sedang dipilih. Pemain yang sedang bermain juga ditandai oleh *underglow* hijau di kotak karakter.

5. Pembagian Tugas

- 13520013 / Ilham Prasetyo Wibowo
    - Pembuatan laporan, dokumentasi, serta README
- 13520052 / Gregorius Moses Marevson
    - Pembuatan laporan
- 13520103 / Amar Fadil
    - Pengerjaan *groundwork* final dan implementasi *design pattern*
    - Integrasi dan modularitas GUI
    - *Refactoring* dan *debugging*
- 13520124 / Owen Christian Wijaya
    - Pengerjaan *groundwork* kasar
    - Desain dan integrasi GUI, dan *debugging*
    - Pengerjaan bonus animasi ringan, halaman select, dan *heal card*
- 135020139 / Fachry Dennis Heraldi
    - Debugging
    - Pembuatan laporan dan dokumentasi

Lampiran: Asistensi

Kelas : 01

Nomor Kelompok : 06

Nama Kelompok : Kosan Benis Split 2

1. 13520013 / Ilham Prasetyo Wibowo
2. 13520052 / Gregorius Moses Marevson
3. 13520103 / Amar Fadil
4. 13520124 / Owen Christian Wijaya
5. 13520139 / Fachry Dennis Heraldi

Asisten Pembimbing : Faris Rizki Ekananda

1. Konten Diskusi

Q: Misalkan hand sudah penuh dan pada draw phase harus discard, aturan kartu yang di-discard bagaimana ya?

A: Metode discard dibebaskan ketika hand penuh saat melakukan draw (bisa discard secara random, atau bisa minta pemain untuk milih yang mana yang mau di-discard, discard dari hand atau board juga dibebaskan)

Q: Apa yang dimaksud dengan dek kosong?

A: Semua kartu sudah kosong, kalau misalnya pada draw phase dia gabisa narik kartu lagi, maka dia kalah, meskipun masih ada kartu tersisa di hand/board.

Q: Misalkan hanya tersisa dua kartu, draw bagaimana ya?

A: Tersisa dua opsi saja untuk penarikan kartu

Q: Ketika phase attack, jika player hp habis nunggu phase end buat nentuin pemenang atau bagaimana?

A: Ketika HP mencapai 0 atau ketika draw phase deck habis, langsung diumumkan pemenangnya.

Q: Deck setiap pemain apakah terpisah kak?

A: Ya, setiap pemain punya deck masing-masing.

Shared bisa berupa spell potion yang diberikan pada player

Untuk ke depannya mungkin asisten akan ngasih masukan via Github Issues, kalau saran yang dimasukkan baik boleh diimplementasikan, tapi kalau menurut kalian lebih baik implementasi sesuai sekarang juga tidak apa-apa

2. Screenshot Bukti