

Kelompok : SC4

1. 13522122 / Maulvi Ziadinda Maulana
2. 13522134 / Shabrina Maharani
3. 13522153 / Muhammad Fauzan Azhim
4. 13522157 / Muhammad Davis Adhipramana
5. 13522164 / Valentino Chryslie Triadi

Asisten Pembimbing : Marcellus Michael Herman Kahari

## 1. Deskripsi Umum Aplikasi



Gambar 1.1 Preview Aplikasi yang Telah Dibuat

Sumber : Dokumentasi Penulis

Aplikasi yang dibuat pada Tugas Besar 2 ini merupakan sebuah program pengelolaan ladang berbasis GUI dengan menggunakan Java. Ladang terdiri dari petak-petak yang berbentuk seperti representasi matriks, di mana setiap petaknya dapat ditanami tanaman atau ditempatkan hewan. Fitur utama dari aplikasi ini mencakup penanaman tanaman, pemeliharaan hewan, penjualan produk dari hasil panen di ladang melalui toko, aktivitas memanen hewan maupun tumbuhan sehingga menjadi produk, pemberian kartu item terhadap ladang sendiri (untuk item baik) maupun ladang musuh (untuk item buruk), serta penghancuran tanaman atau hewan (hilang dari ladang sendiri) oleh serangan beruang.

Aplikasi ini dirancang untuk dimainkan oleh dua pemain, masing-masing memiliki ladang dan kartu sendiri. Pemenang ditentukan berdasarkan jumlah uang yang berhasil dikumpulkan setelah 20 turn. Selain itu, aplikasi ini dibuat dengan antarmuka pengguna grafis (GUI) menggunakan library seperti JavaFX, yang telah diusahakan untuk didesain semenarik mungkin. Program ini juga memiliki fitur save & load untuk menyimpan dan memuat keadaan permainan dalam format txt, serta terdapat plugin untuk memudahkan pengguna untuk melakukan penambahan format file save & load dalam format selain txt.

## 2. Kakas GUI: JavaFX

Kakas GUI yang kami pakai adalah JavaFX. Kakas tersebut sebenarnya cukup mirip dengan Java Swing, perbedaanya adalah JavaFX memiliki lebih banyak fitur. Kami memilih kakas ini karena memiliki fitur FXML dan CSS yang bisa memudahkan pembuatan UI dan juga memiliki *scene builder* yang bisa mempercepat proses penggeraan aplikasi.

Lifecycle dari aplikasi JavaFX dimulai dengan membuat class yang meng-extends `javafx.application.Application`, yang merupakan base class untuk semua aplikasi JavaFX. Child class yang meng-extends kelas application harus memiliki implementasi metode start dan metode start ini akan di-*override*. Di dalam metode start, User Interface akan disiapkan dan diinisialisasi. Metode start akan menerima sebuah parameter objek Stage yang didalamnya akan menyiapkan sebuah scene. Stage adalah jendela utama yang digunakan untuk menampilkan Scene. Sedangkan scene sendiri adalah sebuah objek, yang berisi hubungan - hubungan node yang mewakili elemen-elemen UI. Untuk menjalankan aplikasi JavaFX, metode `launch(args)` dipanggil dari *method main*, yang secara otomatis akan memanggil metode start setelah inisialisasi platform JavaFX selesai. Selama runtime, javaFX akan melakukan event loop untuk menangani interaksi pengguna dan pembaruan UI. Saat aplikasi ditutup, metode `stop()` akan dipanggil untuk membersihkan sumber daya sebelum aplikasi benar-benar berhenti.

Komponen utama yang ada sudah dijelaskan pada paragraf sebelumnya yaitu, Stage, Scene, dan Node. Komponen utama yang pertama adalah Stage, yang merupakan *main window* untuk menampilkan scene. Setiap aplikasi JavaFX memiliki minimal satu stage, yaitu primary stage. Selanjutnya adalah Scene, yang bisa dianggap sebagai wadah untuk menyimpan elemen-elemen UI. Scene ini diatur di dalam stage dan berisi node - node yang terhubung untuk membentuk UI aplikasi. Komponen terakhir adalah Node, yang merupakan elemen dasar dari scene dan berisi beberapa komponen seperti tombol, label, dan panel.

JavaFX menyediakan beragam komponen yang diperlukan untuk membangun antarmuka pengguna (UI). Di antara komponen-komponen tersebut terdapat Control Component, yang mencakup elemen-elemen UI seperti Button, Label, TextField, CheckBox, RadioButton, dan ComboBox yang memungkinkan user untuk berinteraksi dengan aplikasi. Selain itu, JavaFX menyediakan Layout Pane, kelas yang berguna untuk mengatur tata letak UI, seperti HBox, VBox, GridPane, dan StackPane. Selain itu ada juga komponen Media Player yang disediakan oleh

JavaFX yang bisa digunakan untuk memutar musik di dalam aplikasi. JavaFX sendiri memiliki juga file berformat XML yang disebut FXML untuk mendesain struktur dari UI. Komponen - komponen UI tersebut juga dapat di-style menggunakan CSS.

### 3. Plugin & Class Loader

Program Class Loader kami implementasikan ke dalam sebuah kelas dengan nama JarLoader yang bertugas untuk memuat yang diimplementasikan dalam sebuah file JAR. JarLoader menggunakan pola Singleton, memastikan hanya ada satu instance dari kelas ini yang dibuat, dengan menyediakan metode getInstance() untuk mendapatkan objek dari kelas JarLoader. Kelas ini memiliki metode loadJar(String path) bertugas untuk memuat file JAR dari path yang diberikan dan mencari implementasi dari Interface SaveLoad. Selanjutnya akan digunakan kelas URLClassLoader dari Java Dibuat untuk memuat kelas dari file JAR. Selain itu, kelas ServiceLoader juga digunakan untuk memuat implementasi dari antarmuka SaveLoad yang ada di dalam file JAR, dan mengecek apakah ada implementasi SaveLoad yang ditemukan. Jika ditemukan, kelas tersebut akan diambil dan diperiksa apakah memiliki anotasi SaveLoadAnnotation. Jika anotasi ditemukan, metode load Jar mengembalikan implementasi dalam bentuk pasangan (entry) Map.Entry<String, SaveLoad>, dengan type yang didapatkan dari annotation sebagai kunci dan sebuah instance dari SaveLoad sebagai value.

Setelah dimuat dan diverifikasi oleh JarLoader, entry map yang di return akan diterima oleh sebuah kelas Plugins. Kelas ini juga menggunakan pola Singleton untuk memastikan hanya ada satu instance yang dibuat. Plugin-plugin disimpan dalam sebuah Map yang berisi pasangan tipe plugin (String) dan instance SaveLoad. Kelas ini memiliki 2 metode utama yaitu addClass(Map.Entry<String, SaveLoad> entry) digunakan untuk menambahkan plugin baru ke dalam Map, dan metode getPlugin(String type) digunakan untuk mengambil plugin berdasarkan tipe yang diberikan. Tujuan utama dari kelas Plugins adalah untuk mengelola dan memuat plugin-plugin dari file JAR sehingga user bisa menambahkan plugin secara dinamis ketika runtime. Perlu diketahui juga bahwa sesuai spesifikasi yang diberikan, kelas ini tidak memiliki fungsi untuk menghapus plugin yang telah diupload oleh user.

Plugin yang dibuat sebagai Jar eksternal akan mengimplementasikan Interface SaveLoad dari program utama yang berarti bahwa kelas tersebut bisa menyimpan dan memuat data permainan. Antarmuka Save memiliki metode save(String folderName) untuk menyimpan data, sedangkan antarmuka Load memiliki metode loadGameState() dan loadPlayer(int no\_player) untuk memuat data permainan. Selain itu kelas yang ada di dalam plugin juga harus memiliki anotasi SaveLoadAnnotation pada kelas untuk menandai kelas implementasi SaveLoad dan memberikan tipe plugin. Plugin akan menggunakan API yang tersedia dari program utama, seperti Gamestate.getInstance().getShop(), untuk mendapatkan data permainan ketika runtime. Dengan implementasi ini, plugin dapat menambah fungsionalitas program utama untuk memuat dan menyimpan data permainan secara dinamis dalam berbagai tipe data.

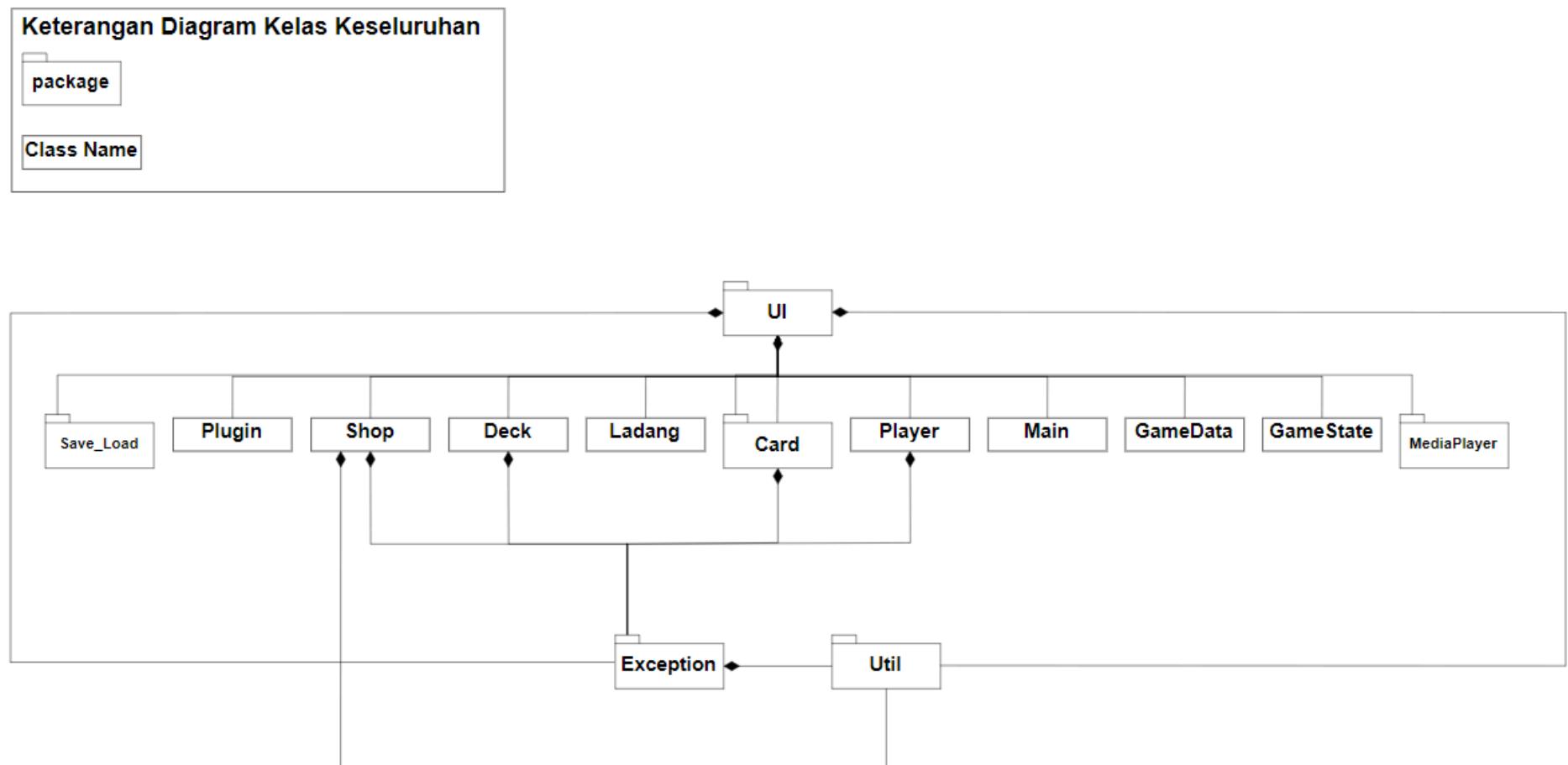
## 4. Class Diagram

Diagram kelas yang disajikan dalam bagian ini merepresentasikan struktur umum dari program permainan yang telah dibuat dengan berbagai kelas yang saling berinteraksi. Pada pusat diagram ini terdapat Package UI (User Interface), yang di dalamnya terdapat kelas-kelas yang bertanggung jawab sebagai penghubung antara pemain (user) dengan program. Dalam package UI terdapat kelas seperti GameWindowController yang dapat mengirimkan informasi kepada pemain dan menerima input dari pemain, kemudian mengarahkan informasi tersebut ke kelas-kelas lain yang sesuai, seperti Player, GameState, atau kelas-kelas lainnya di luar package UI, untuk diproses sesuai dengan spesifikasi yang diberikan.

Kelas-kelas di luar package UI terhubung dengan UI melalui dengan adanya proses import suatu kelas atau package yang dibutuhkan oleh masing-masing proses yang akan ditampilkan oleh GUI. Hal ini menunjukkan bahwa kelas-kelas tersebut dapat menerima perintah atau data dari UI dan mengirimkan kembali hasilnya sehingga pemain dapat melihat secara langsung hasil dari proses permainan yang mereka lakukan. Misalnya, ketika pemain melakukan aksi seperti memilih empat Card pada awal permainan. Informasi ini akan dikirim oleh objek GUI yaitu Pane ke kelas GameWindowController terlebih dahulu untuk dilakukan proses terkait penambahan kartu yang nantinya kartu tersebut akan muncul sebagai objek Card dalam Active Deck dengan pengiriman informasi ke kelas Card dan Deck.

Selain itu, pada diagram kelas ini ditunjukkan adanya hubungan komposisi antara kelas-kelas di luar package UI dengan package UI, yang berarti UI dapat mengelola siklus hidup objek-objek dari kelas tersebut. Ini memungkinkan UI untuk bertindak sebagai pengontrol utama yang menyimpan informasi berupa perilaku pemain dan menjalankan logika program berdasarkan input tersebut. Misalnya, setiap kali pemain melakukan aksi tertentu, UI tidak hanya mengirimkan informasi tersebut ke kelas yang relevan, tetapi juga mungkin menyimpan status terkini dari pemain-pemain di kelas GameState untuk memastikan bahwa permainan dapat dilanjutkan dari titik terakhir state program pemain tersebut jika terjadi penyimpanan status atau perubahan dalam permainan. Berikut adalah diagram kelas secara umum dari program yang telah dibuat serta detail dari masing-masing package ataupun kelas yang terdefinisi.

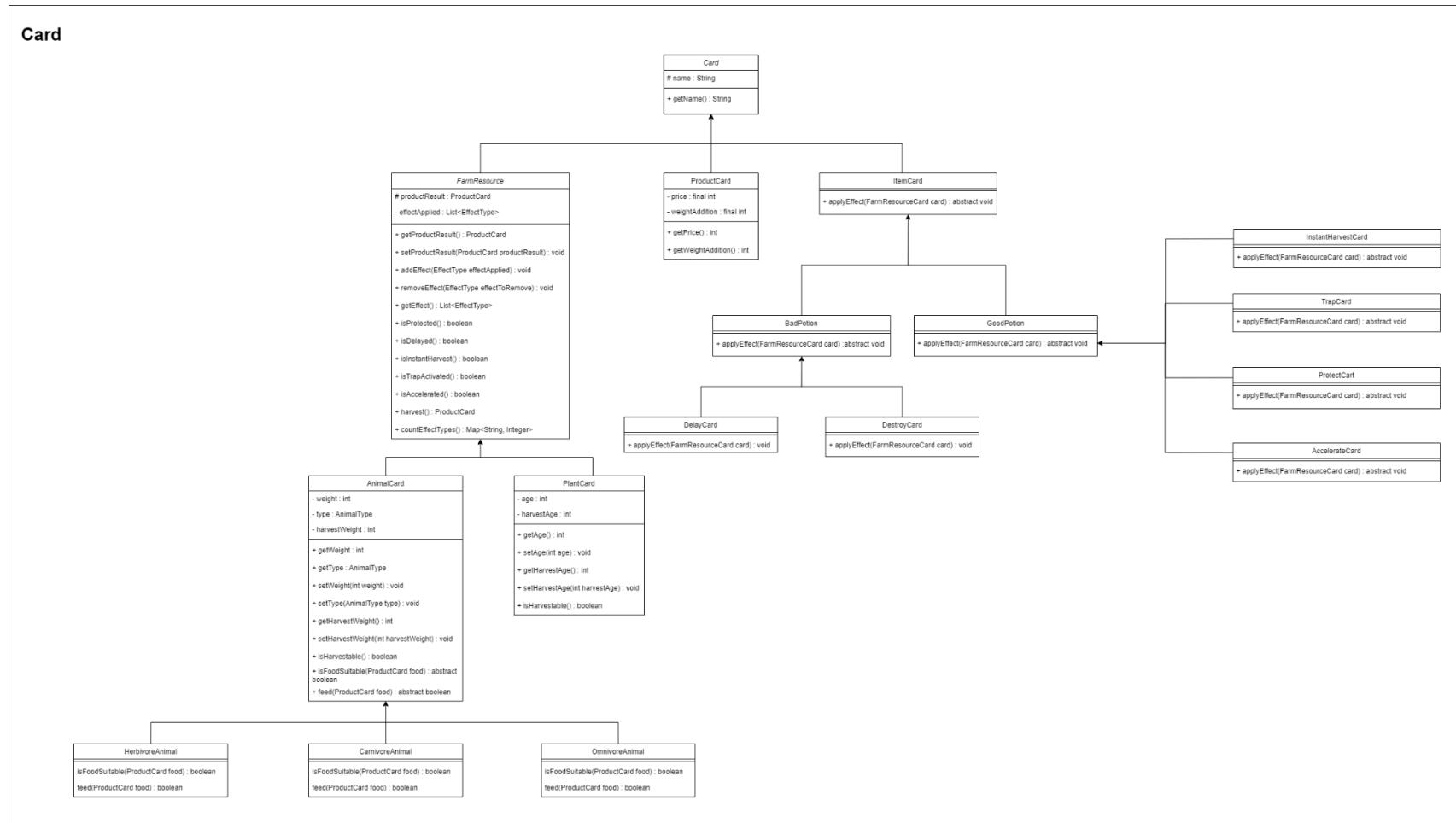
#### 4.1. Class Diagram Keseluruhan



Gambar 4.1 Diagram Kelas Keseluruhan

Sumber : [draw.io](https://draw.io) page All

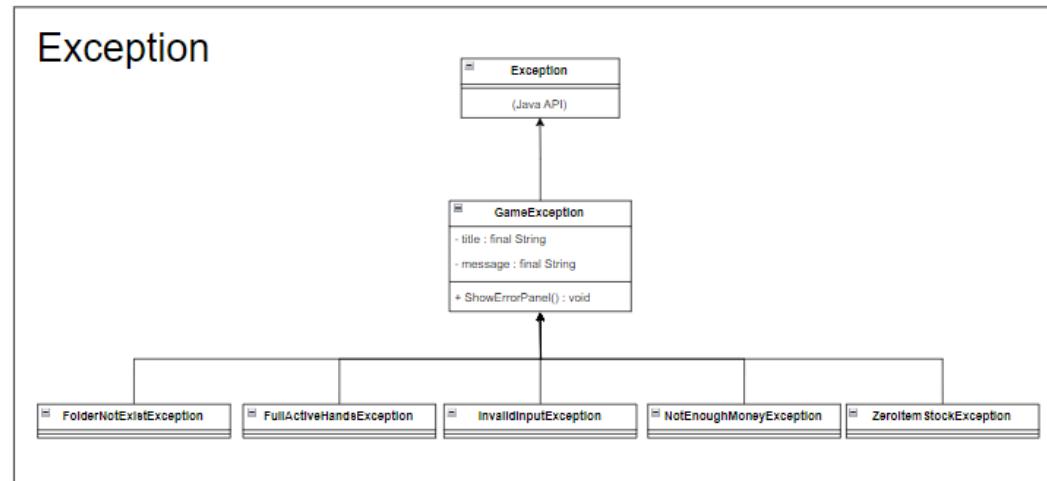
## 4.2. Detail Package Card



Gambar 4.2 Detail Diagram Kelas Package Card

Sumber : [draw.io](https://draw.io) page Card

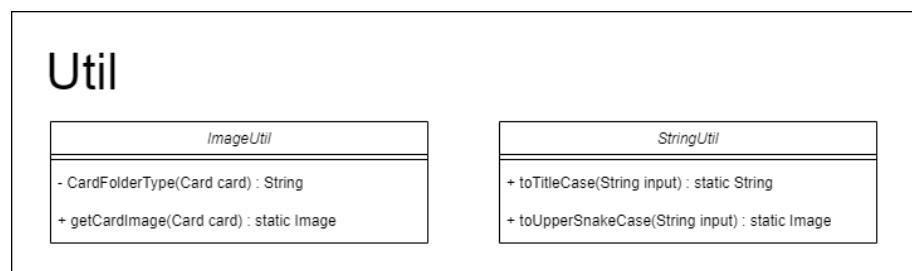
### 4.3. Detail Package Exception



Gambar 4.3 Detail Diagram Kelas Package Exception

Sumber : [draw.io](#) page Exception

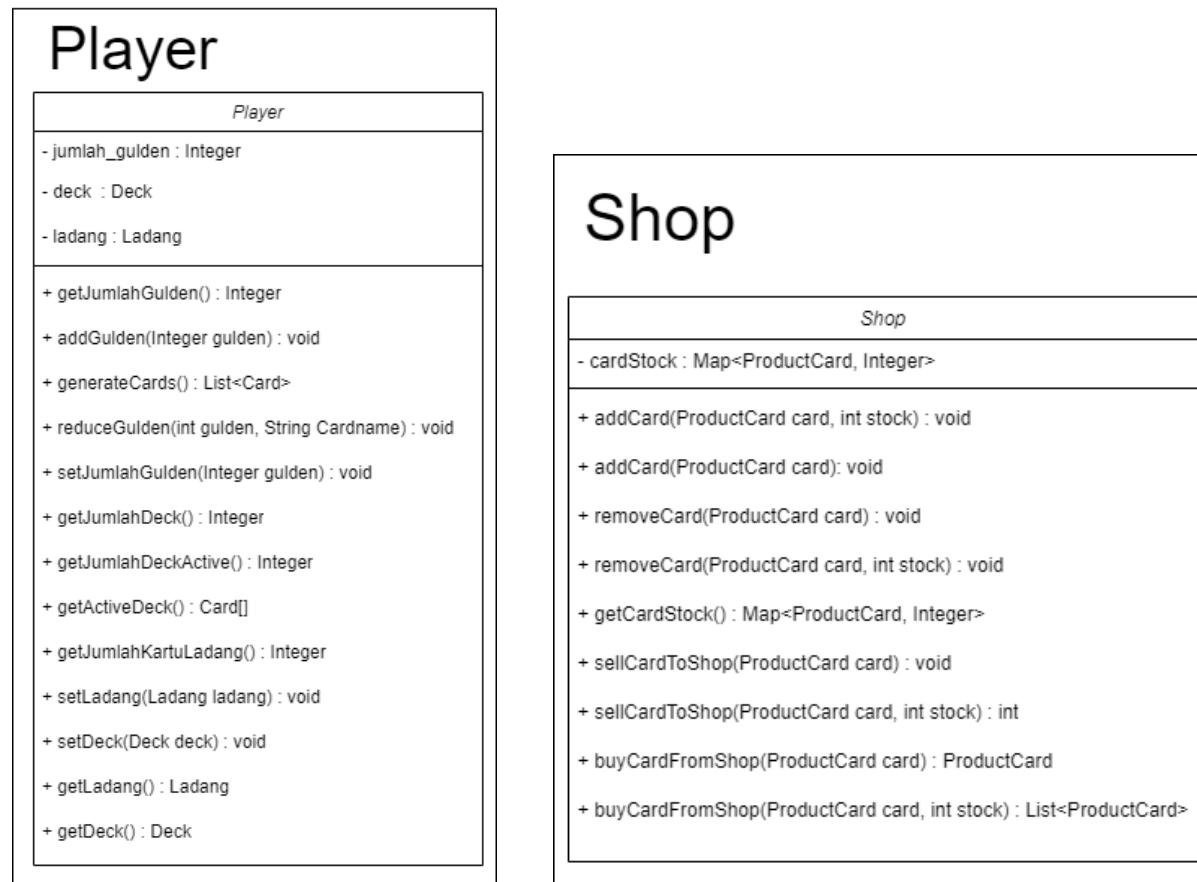
### 4.4. Detail Package Util



Gambar 4.4 Detail Diagram Kelas Package Util

Sumber : [draw.io](#) page Util

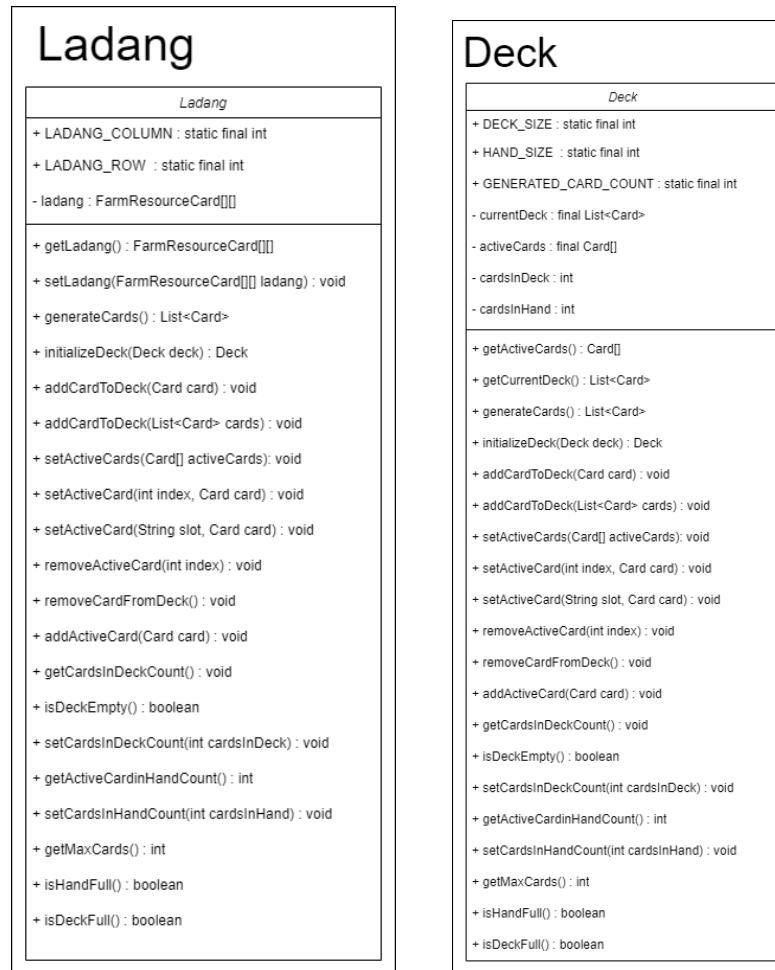
## 4.5. Detail Class Player dan Shop



Gambar 4.5 Detail Diagram Kelas Player dan Shop

Sumber : [draw.io](#) page Player dan Shop

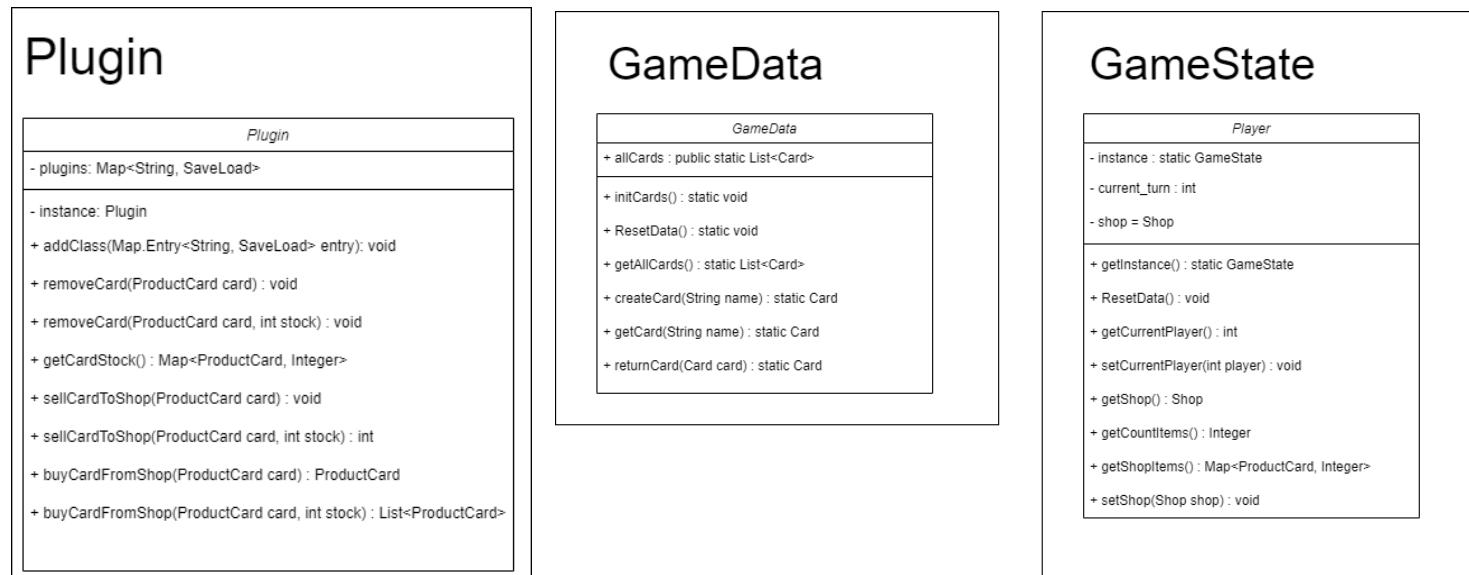
## 4.6. Detail Class Ladang dan Deck



Gambar 4.6 Detail Diagram Kelas Ladang dan Deck

Sumber : [draw.io](https://draw.io/page/Ladang%20dan%20Deck) page Ladang dan Deck

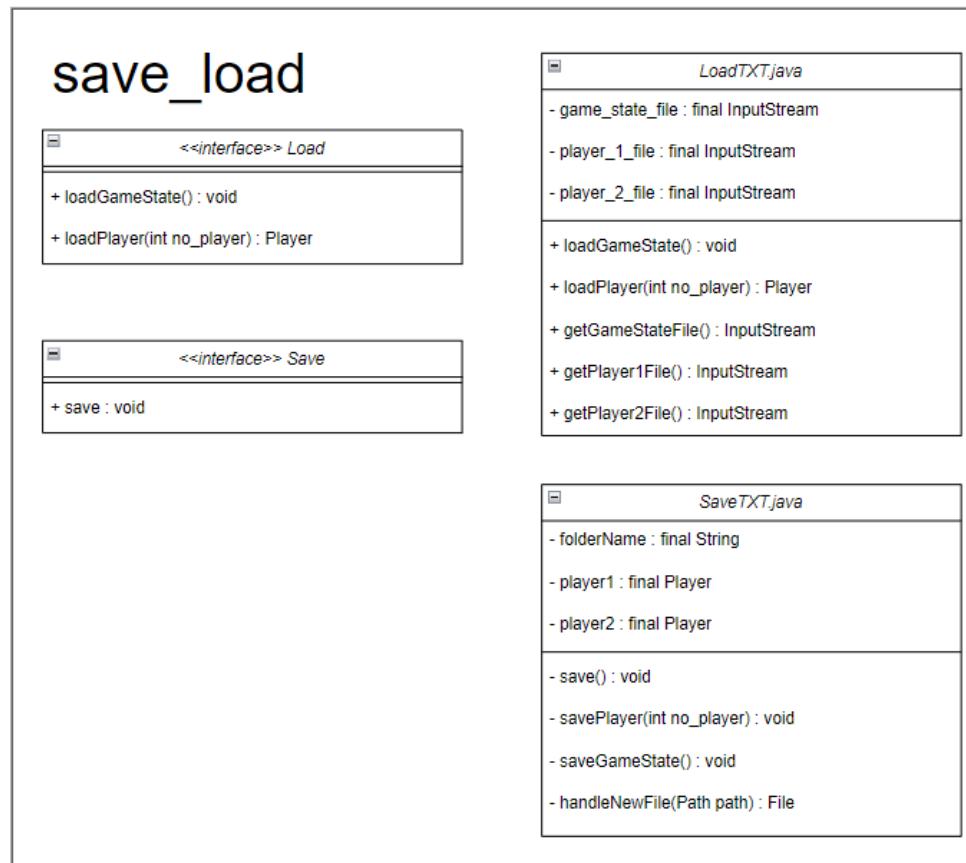
#### 4.7. Detail Class Plugin, GameData, GameState



Gambar 4.7 Detail Diagram Kelas Plugin, GameData, dan GameState

Sumber : [draw.io](#) page Plugin dan GameData&GameState

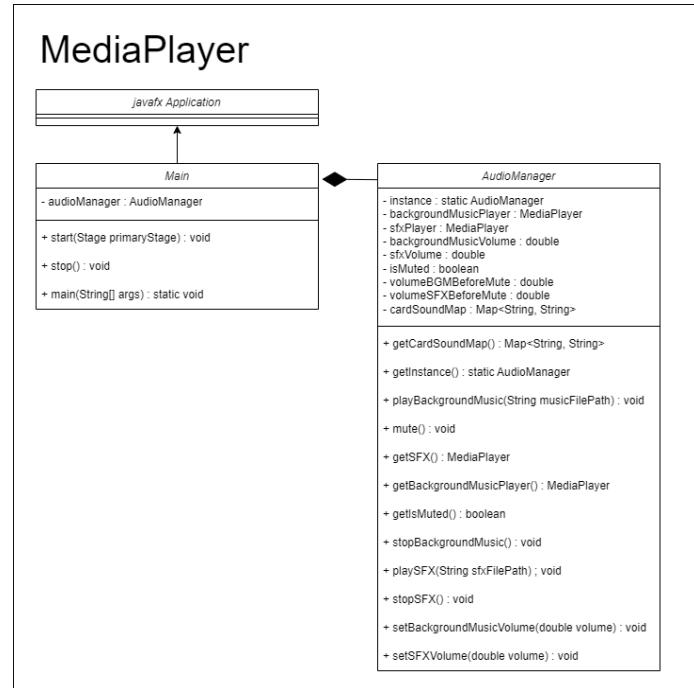
#### 4.8. Detail Package Save\_Load



Gambar 4.8 Detail Diagram Package Save\_Load

Sumber : [draw.io](#) page Save\_Load

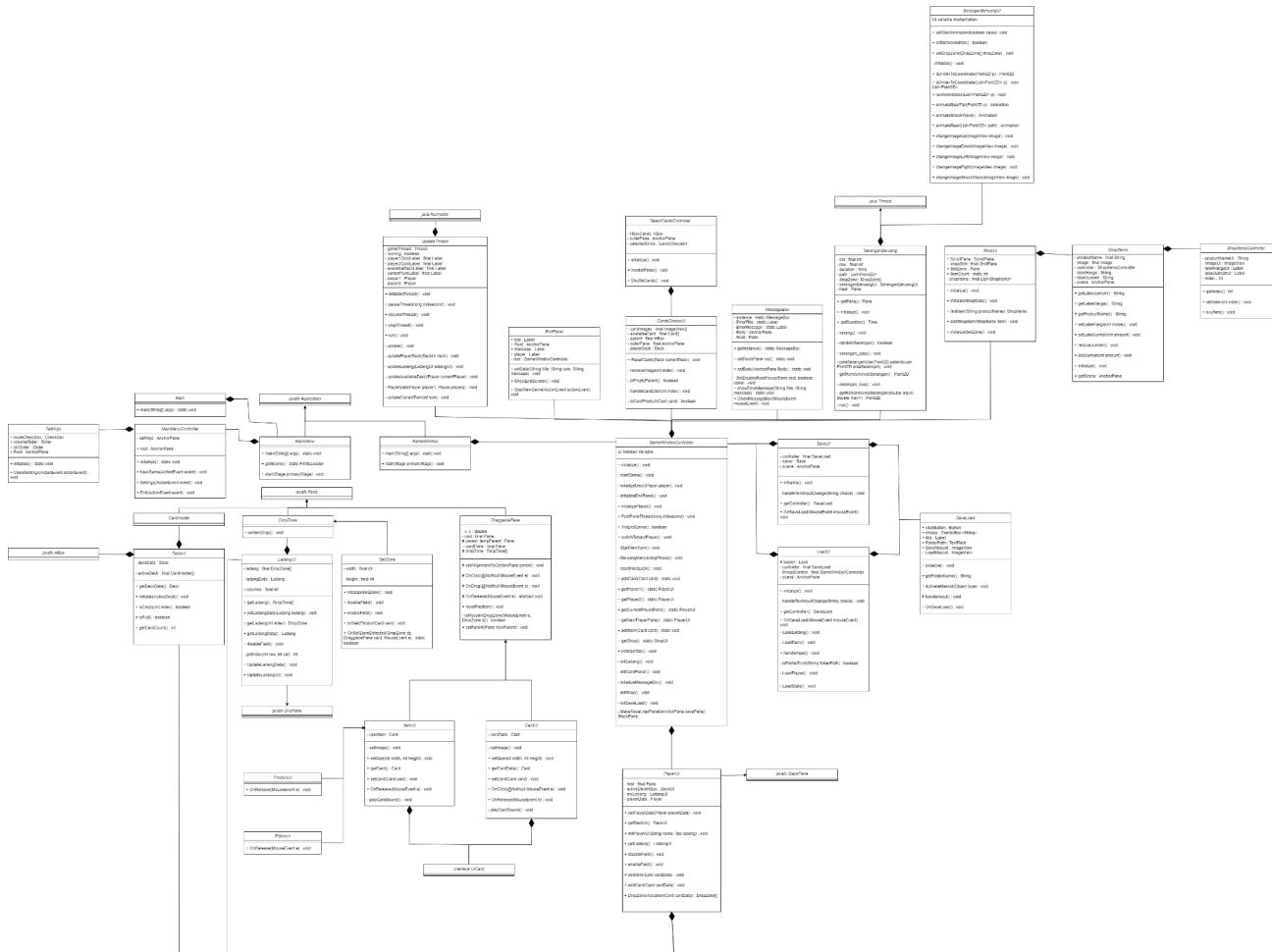
## 4.9. Detail Package MediaPlayer



Gambar 4.9 Detail Diagram Package MediaPlayer

Sumber : [draw.io](#) page MediaPlayer

## 4.10. Detail Package UI



**Gambar 4.10 Detail Diagram Package UI**

Sumber : [draw.io](#) page UI

## 5. Konsep OOP

### 5.1. Inheritance

Berikut adalah daftar penggunaan konsep Inheritance dalam aplikasi:

- Kelas AnimalCard (src/main/java/oop/if2210\_tb2\_sc4/card/AnimalCard.java) dan kelas PlantCard (src/main/java/oop/if2210\_tb2\_sc4/card/PlantCard.java) merupakan anak dari kelas FarmResourceCard (src/main/java/oop/if2210\_tb2\_sc4/card/FarmResourceCard.java).
- Kelas DelayCard (src/main/java/oop/if2210\_tb2\_sc4/card/DelayCard.java), ProtectCard (src/main/java/oop/if2210\_tb2\_sc4/card/ProtectCard.java), TrapCard (src/main/java/oop/if2210\_tb2\_sc4/card/TrapCard.java), InstantHarvestCard (src/main/java/oop/if2210\_tb2\_sc4/card/InstantHarvestCard.java), dan AccelerateCard (src/main/java/oop/if2210\_tb2\_sc4/card/AccelerateCard.java) merupakan anak dari Kelas ItemCard (src/main/java/oop/if2210\_tb2\_sc4/card/ItemCard.java)
- Kelas ItemCard (src/main/java/oop/if2210\_tb2\_sc4/card/ItemCard.java), ProductCard (src/main/java/oop/if2210\_tb2\_sc4/card/ProductCard.java), dan FarmResourceCard (src/main/java/oop/if2210\_tb2\_sc4/card/FarmResourceCard.java) adalah anak dari Kelas Card (src/main/java/oop/if2210\_tb2\_sc4/card/FarmResourceCard.java).
- Kelas GameException (src/main/java/oop/if2210\_tb2\_sc4/Exception/GameException.java) adalah turunan dari kelas *built in* yaitu Exception.
- Kelas FolderNotExistException (src/main/java/oop/if2210\_tb2\_sc4/Exception/FolderNotExistException.java), FullActiveHand (src/main/java/oop/if2210\_tb2\_sc4/Exception/FullActiveHandsException.java), dan InvalidInputException (src/main/java/oop/if2210\_tb2\_sc4/Exception/InvalidInputException.java) adalah turunan dari kelas GameException (src/main/java/oop/if2210\_tb2\_sc4/Exception/GameException.java).

### 5.2. Composition

Berikut adalah daftar penggunaan konsep composition dalam aplikasi.

- Kelas GameData (src/main/java/oop/if2210\_tb2\_sc4/GameData.java) memiliki tepat satu kelas FarmResourceCard (src/main/java/oop/if2210\_tb2\_sc4/card/FarmResourceCard.java)

- Kelas Player (src/main/java/oop/if2210\_tb2\_sc4/Player.java) memiliki tepat satu kelas Deck (src/main/java/oop/if2210\_tb2\_sc4/Deck.java) dan satu kelas ladang (src/main/java/oop/if2210\_tb2\_sc4/Ladang.java)
- Kelas GameState (src/main/java/oop/if2210\_tb2\_sc4/GameState.java) memiliki dua kelas Player(src/main/java/oop/if2210\_tb2\_sc4/Player.java) dan tepat satu kelas Shop (src/main/java/oop/if2210\_tb2\_sc4/Shop.java).
- Kelas Deck (src/main/java/oop/if2210\_tb2\_sc4/Deck.java) memiliki banyak kelas Card (src/main/java/oop/if2210\_tb2\_sc4/card/Card.java)
- Kelas SeranganBeruang (src/main/java/oop/if2210\_tb2\_sc4/UI/SeranganBeruang.java) memiliki tepat satu kelas LadangUI (src/main/java/oop/if2210\_tb2\_sc4/UI/LadangUI.java) dan satu kelas SeranganBeruangUI (src/main/java/oop/if2210\_tb2\_sc4/UI/SeranganBeruangUI.java)

### 5.3. Interface

Berikut adalah penggunaan konsep Interface dalam aplikasi.

- Interface SelectionFinishListener (src/main/java/oop/if2210\_tb2\_sc4/UI/SelectionFinishListener.java)
- Interface SaveLoad (src/main/java/oop/if2210\_tb2\_sc4/save\_load/SaveLoad.java)
- Interface UICard (src/main/java/oop/if2210\_tb2\_sc4/UI/UICard.java)
- Interface Eatable (src/main/java/oop/if2210\_tb2\_sc4/card/Eatable.java)
- Interface SelectionFinishListener (src/main/java/oop/if2210\_tb2\_sc4/UI/SelectionFinishListener.java)

### 5.4. Method Overriding dan Method Overloading

Berikut adalah penggunaan Overriding dan OverLoading dalam aplikasi.

- Method overloading setActiveCards(Card[] activeCards) , setActiveCard(int index, Card card), dan setActiveCard(String slot, Card card) pada kelas Deck (src/main/java/oop/if2210\_tb2\_sc4/Deck.java).
- Method overloading removeActiveCard(int index) pada kelas Deck (src/main/java/oop/if2210\_tb2\_sc4/Deck.java).
- Method overloading setCard(int row, int column, FarmResourceCard card) dan setCard(String slot, FarmResourceCard card) pada kelas Ladang (src/main/java/oop/if2210\_tb2\_sc4/Ladang.java).
- Method overriding OnRelease(MouseEvent e) pada kelas (src/main/java/oop/if2210\_tb2\_sc4/UI/PotionUI.java)
- Method overriding applyEffect(FarmResourceCard card) pada kelas TrapCard (src/main/java/oop/if2210\_tb2\_sc4/card/TrapCard.java)

## 5.5. Polymorphism

Berikut adalah penggunaan Polymorphism dalam aplikasi.

- Method `onSellZoneDetected` pada kelas `SellZone` (`src/main/java/oop/if2210_tb2_sc4/UI/SellZone.java`) menerima kelas `DraggablePane`, tapi method ini dipanggil pada Kelas `ItemUI` (`src/main/java/oop/if2210_tb2_sc4/UI/ItemUI.java`) dengan parameter `ItemUI`.
- Kelas `GameData` (`src/main/java/oop/if2210_tb2_sc4/GameData.java`) memiliki atribut `allcards` (`List<Card>`), namun dalam implementasinya `allCards` ditambahkan elemen yang merupakan instance dari kelas-kelas yang merupakan child dari `Card` (`src/main/java/oop/if2210_tb2_sc4/Card/Card.java`)

## 5.6. Java API Collection

Berikut adalah penggunaan Java API Collection dalam aplikasi.

- Kelas `Deck` (`src/main/java/oop/if2210_tb2_sc4/card/Deck.java`) memiliki list of `Card`.
- Kelas `Shop` (`src/main/java/oop/if2210_tb2_sc4/Shop.java`) memiliki Map dengan key `Product Card` dan valuenya adalah `Integer`.
- Kelas `SaveTXT` (`src/main/java/oop/if2210_tb2_sc4/save_load/SaveTXT.java`) memanfaatkan java stream API `forEach`.
- Kelas `SeranganBeruang` (`src/main/java/oop/if2210_tb2_sc4/UI/SeranganBeruang.java`) memiliki set of `Point2D`.
- Kelas `SeranganBeruang` (`src/main/java/oop/if2210_tb2_sc4/UI/SeranganBeruang.java`) memanfaatkan java stream API `forEach`.

## 5.7. SOLID

<b>SOLID</b>			
SOLID	Nama Kelas	Path	Penjelasan
S	Deck	<code>src/main/java/oop/if2210_tb2_sc4/Deck.java</code>	memiliki tanggung jawab terhadap seluruh kartu yang ada di dalam deck

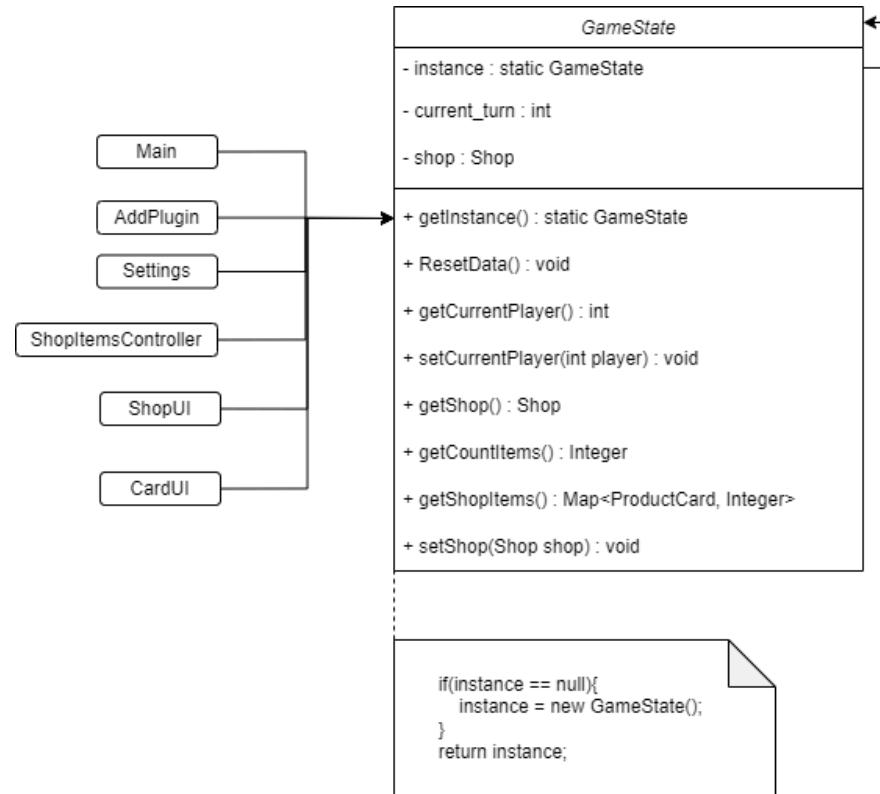
	Ladang	src/main/java/oop/if2210_tb2_sc4/Ladang.java	memiliki tanggung jawab terhadap seluruh kartu yang ada di ladang dan seluruh proses di ladang
	Shop	src/main/java/oop/if2210_tb2_sc4/Shop.java	Memiliki tanggung jawab terhadap seluruh item di toko dan metode untuk jual beli
O	Plugins	src/main/java/oop/if2210_tb2_sc4/Plugins.java	Dengan kelas ini, program tidak perlu untuk dimodifikasi untuk menambahkan fitur baru seperti save dan load dengan tipe file JSON atau XML
	Card	src/main/java/oop/if2210_tb2_sc4/card/Card.java	Card tidak perlu dimodifikasi untuk menambahkan jenis entity card baru, seperti obstacle, building, atau yang lainnya. Hal tersebut dimungkinkan untuk terjadi akibat dari kelas-kelas baru tersebut yang dapat di extend dari kelas card
L	FarmResourceCard	src/main/java/oop/if2210_tb2_sc4/card/FarmResourceCard.java	Dapat disubstitusi dengan semua kartu yang merupakan child atau inheritance dari FarmResourceCard, seperti AnimalCard, PlantCard, dan semua <i>inherited object</i> nya.
	ItemCard	src/main/java/oop/if2210_tb2_sc4/card/ItemCard.java	Dapat disubstitusi oleh semua kartu yang merupakan child atau inheritance dari ItemCard, seperti BadPotion, GoodPotion, atau <i>inherited object</i> lainnya.
	AnimalCard	src/main/java/oop/if2210_tb2_sc4/card/AnimalCard.java	Dapat disubstitusi oleh semua kartu yang merupakan child atau inheritance dari AnimalCard, seperti OmnivoreAnimal, CarnivoreAnimal, dan HerbivoreAnimal.
I	OmnivoreAnimal	src/main/java/oop/if2210_tb2_sc4/card/OmnivoreAnimal.java	Kelas ini mengimplementasikan interface Eatable (src/main/java/oop/if2210_tb2_sc4/card/Eatable.java)

			va) yang memungkinkan kelas ini untuk tidak mengerjakan proses yang tidak seharusnya.
	SaveTXT	src/main/java/oop/if2210_tb2_sc4/save_load/ SaveTXT.java	Kelas ini mengimplementasikan interface Save (src/main/java/oop/if2210_tb2_sc4/save_load/Save.java) yang memungkinkan kelas ini hanya bertanggung jawab terhadap proses penyimpanan tanpa harus memaksa kelas ini untuk melakukan pemuatan
	LoadTXT	src/main/java/oop/if2210_tb2_sc4/save_load/ LoadTXT.java	Kelas ini mengimplementasikan interface Load (src/main/java/oop/if2210_tb2_sc4/save_load/Load.java) yang memungkinkan kelas ini hanya bertanggung jawab terhadap proses pemuatan tanpa harus memaksa kelas ini untuk melakukan penyimpanan
D	LoadUI	src/main/java/oop/if2210_tb2_sc4/UI/LoadUI.java	Kelas LoadUI bergantung pada LoadTXT dan kelas Plugins yang akan ditambahkan. LoadUI akan bergantung pada LoadTXT dan Plugins melalui antarmuka interface load, yang nantinya dapat diimplementasikan ketika ada penambahan plugin baru.
	SaveUI	src/main/java/oop/if2210_tb2_sc4/UI/SaveUI.java	Kelas SaveUI bergantung pada SaveTXT dan kelas Plugins yang akan ditambahkan. SaveUI akan bergantung pada SaveTXT dan Plugins melalui antarmuka interface save, yang nantinya dapat diimplementasikan ketika ada penambahan plugin baru.

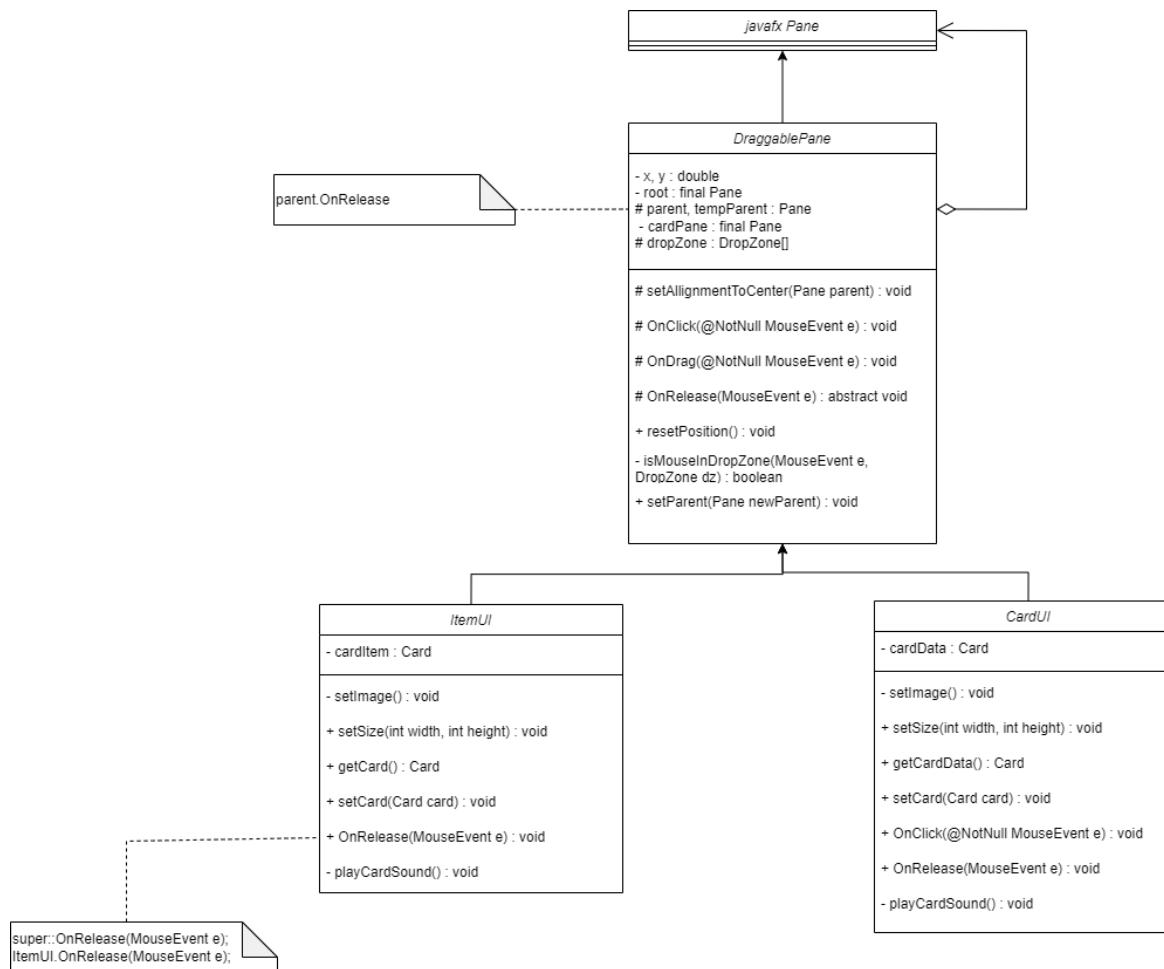
## 5.8. Design Pattern

Berikut adalah penggunaan Polymorphism dalam aplikasi.

- Creational pattern dengan jenis Singleton Pattern: GameState (src/main/java/oop/if2210\_tb2\_sc4/GameState.java) sebagai kelas singleton dengan tujuan untuk memastikan hanya ada satu instance dalam satu runtime, kelebihannya adalah mudah diakses, tidak mungkin menimbulkan kekeliruan pengaksesan objek, hanya perlu satu static variabel untuk menciptakan static class, jika tidak diterapkan harus menciptakan kelas dengan variabel yang harus distatikan semuanya.

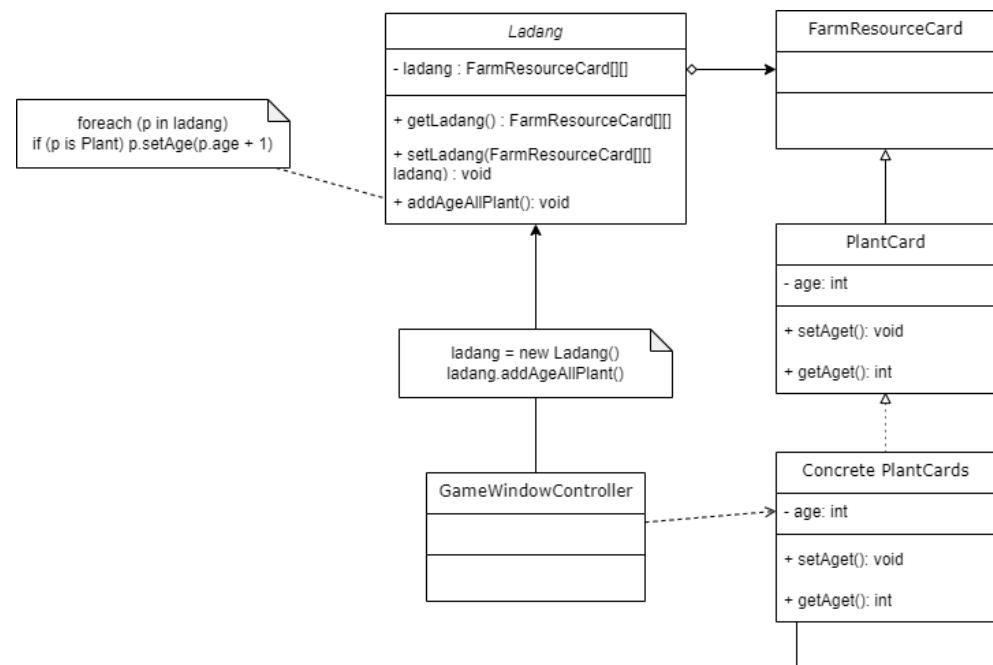


- Structural Patterns dengan jenis Decorator Pattern : DraggablePane (src/main/java/oop/if2210\_tb2\_sc4/UI/DraggablePane.java) sebagai kelas Decorator Pattern yang meng-inherit kelas Pane dan membutuhkan kelas Pane sebagai atribut, kemudian kelas DraggablePane juga di-inherit oleh beberapa kelas lainnya seperti CardUI (src/main/java/oop/if2210\_tb2\_sc4/UI/CardUI.java). Kelebihan dari penggunaan pattern ini termasuk fleksibilitas dan keterbacaan kode yang lebih baik karena fitur tambahan dapat disusun secara modular dan dipisahkan dari kode utama objek. Jika pattern ini tidak diterapkan, setiap penambahan fungsionalitas baru seperti draggable atau elemen UI lainnya harus ditambahkan langsung ke dalam kelas Pane atau subkelasnya, yang dapat menyebabkan kode menjadi monolitik, sulit dirawat, dan tidak fleksibel untuk perubahan di masa depan.



- Behavioral Patterns dengan jenis Observer Pattern : GameWindowController (src/main/java/oop/if2210\_tb2\_sc4/UI/GameWindowController.java) sebagai client memanggil kelas Ladang yang di dalam kelas Ladang, ia akan melakukan notify ke seluruh kelas PlantCard untuk memanggil metode addAge. Tujuan utama penggunaan Observer Pattern di sini adalah untuk memastikan bahwa setiap perubahan dalam kondisi Ladang secara otomatis dan efisien diperbarui pada

setiap PlantCard, sehingga menjaga sinkronisasi data tanpa perlu kode yang berulang atau rumit. Kelebihan dari penggunaan pattern ini adalah memudahkan dalam menambah atau mengurangi pengamat (dalam hal ini, PlantCard) tanpa perlu mengubah kode Ladang atau GameWindowController, meningkatkan skalabilitas dan maintainability. Jika design pattern ini tidak diterapkan, akan terjadi ketergantungan yang kuat antara Ladang dan PlantCard, yang mengakibatkan kode menjadi lebih sulit untuk dirawat dan diperluas, serta meningkatkan risiko inkonsistensi data karena setiap pembaruan harus dilakukan secara manual dan berulang-ulang di berbagai tempat.



## 5.9. Reflection

- Method `loadJar` pada kelas `JarLoader` (`src/main/java/oop/if2210_tb2_sc4/JarLoader.java`) menggunakan reflection `getAnnotation` untuk mengecek tipe plugin yang masuk (apakah XML atau JSON)

- Kelas LoadUI (src/main/java/oop/if2210\_tb2\_sc4/UI/LoadUi.java) menggunakan reflection getConstructor untuk membuat instance baru dari kelas yang diupload oleh user dalam bentuk Jar.
- Kelas SaveLoadAnnotation (src/main/java/oop/if2210\_tb2\_sc4/save\_load/SaveLoadAnnotation.java) menggunakan reflection Retention, RuntimePolicy, Target, dan ElementType untuk membuat sebuah annotation.

## 5.10. Threading

Berikut adalah penggunaan konsep threading yang digunakan dalam aplikasi.

- Threading digunakan dalam kelas SeranganBeruang (src/main/java/oop/if2210\_tb2\_sc4/UI/SeranganBeruang.java) karena aplikasi perlu tetap responsif selama serangan berlangsung. Tanpa penggunaan thread, aplikasi akan terkunci (freeze) saat serangan berlangsung, karena eksekusi program secara berurutan dan blocking. Dengan thread, serangan beruang dapat berlangsung di latar belakang, sementara aplikasi tetap merespons input pengguna.
- Threading digunakan dalam kelas UpdateThread (src/main/java/oop/if2210\_tb2\_sc4/UI/UpdateThread.java) karena aplikasi perlu untuk mengatur pembaruan UI secara terpisah agar menjaga responsivitas UI. Pembaruan yang membutuhkan waktu lama yang dilakukan tanpa multithreading dapat menghambat responsivitas UI karena ketika update dilakukan, thread tidak bisa menerima input jadi seakan-akan program mengalami lag.

## 6. Bonus Yang dikerjakan

### 6.1. Unit Testing

- Menambahkan unit testing untuk card dengan menggunakan JUnit5

## 6.2. Memperindah UI

- Menambahkan tampilan UI
- Menambahkan backsound
- Menambahkan efek suara untuk entity
- Menambahkan pengaturan untuk suara
- Menambahkan animasi untuk serangan beruang
- Menambahkan efek untuk clock pada serangan beruang

## 7. Pembagian Tugas

Pada bagian ini, tuliskan peran terbesar untuk masing-masing anggota kelompok. Tuliskan sesuai kenyataan. Misalnya:

- 13522122 / Maulvi Ziadinda Maulana
  - Membuat seluruh struktur kelas dalam program
  - Membuat mekanisme plugin
  - Melengkapi laporan
- 13522134 / Shabrina Maharani
  - Mendekomposisi kelas seperti kelas Animal dan ItemCard
  - Membuat inisialisasi proses UI untuk detail informasi FarmResourceCard
  - Membuat diagram kelas dan melengkapi laporan
- 13522153 / Muhammad Fauzan Azhim
  - Bertanggung jawab untuk tampilan (termasuk memperindah UI) dan penggunaan javafx
  - Membuat teknis Serangan Beruang
  - Menghubungkan antara UI dengan proses dalam program

- 13522157 / Muhammad Davis Adhipermana
  - Bertanggung jawab untuk tampilan (termasuk memperindah UI) dan penggunaan javafx
  - Menghubungkan antara UI dengan proses dalam program
  - Integrasi Save Load
- 13522164 / Valentino Chryslie Triadi
  - Membuat mekanisme plugin
  - Menginisialisasi struktur kelas dalam program
  - Membuat Save Load

## 8. Foto Kelompok



## Lampiran Dokumen Asistensi

Kode Kelompok : SC4

Nama Kelompok : DD JATINANGOR V2

1. 13522122 / Maulvi Ziadinda Maulana
2. 13522134 / Shabrina Maharani
3. 13522153 / Muhammad Fauzan Azhim
4. 13522157 / Muhammad Davis Adhipramana
5. 13522164 / Valentino Chryslie Triadi

Asisten Pembimbing : Marcellus Michael Herman Kahari

### 1. Konten Diskusi

Pertanyaan	Jawaban
Untuk pembuatan UML, apakah seharusnya dibuat dalam bentuk Model Relasional atau bagaimana ya kak?	Pembuatan UML dibuat dalam bentuk diagram kelas
Untuk item berupa protect card, jika suatu hewan atau tanaman diberikan item destroy atau delay, apakah protect card tersebut hilang?	Tidak, tetap ada
Saat aksi bebas tetap bisa di serang dengan serangan beruang atau tidak?	Bisa
Jumlah yang ditampilkan untuk deck itu merupakan jumlah sisa kartu atau jumlah kartu yang sudah diambil?	Sisa kartu
Untuk bonus pengubahan ukuran grid, itu hanya jumlah grid nya saja yang berubah atau ukuran kartu nya juga berubah?	Semuanya ikut berubah

Untuk plugins itu implementasi loadernya dibebaskan atau bagaimana ya kak?	Dibebaskan
--	------------

## 2. Tindak Lanjut

Berikut adalah beberapa hal yang kami lakukan untuk menanggapi hasil dari konten diskusi.

- Membuat UML dalam bentuk Class Diagram.
- Merubah perlaku efek dari kartu item-item yang berkaitan.
- Tetap menggunakan kode yang sudah ada untuk serangan beruang karena sudah sesuai.
- Menampilkan jumlah sisa kartu untuk tampilan jumlah deck.
- Untuk bonus pengubahan ukuran grid tidak dikerjakan karena waktu yang tidak cukup
- Untuk implementasi plugins tetap berpacu pada spesifikasi tetapi metode pengimplementasiannya dilakukan dengan mencari sumber dari luar

### 3. Foto Dokumentasi

