

Kelompok : FCK

1. 13522012 / Thea Josephine Halim
2. 13522046 / Raffael Boymian Siahaan
3. 13522062 / Salsabiila
4. 13522096 / Novelya Putri Ramadhani
5. 13522104 / Diana Tri Handayani

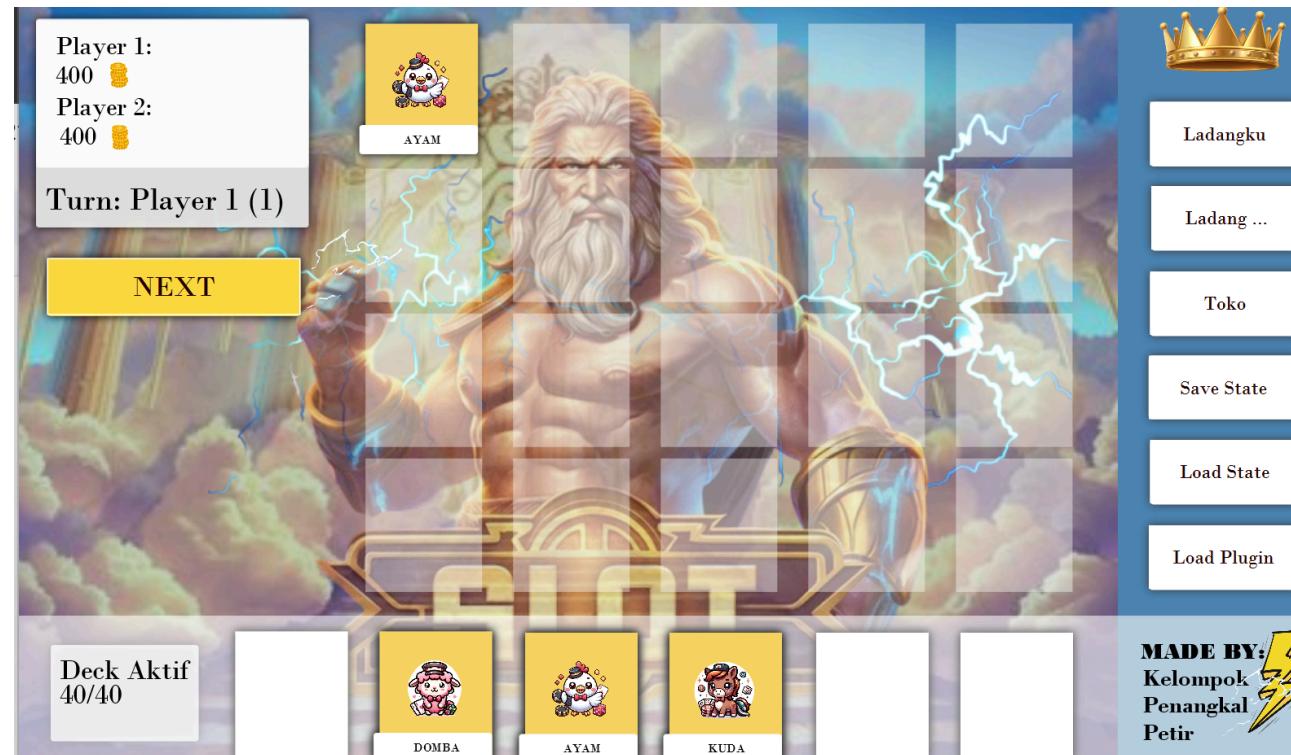
6. 10022010 / M Tegar Wijaksono

Asisten Pembimbing : 13520105 / Malik Akbar Hashemi Rafsanjani

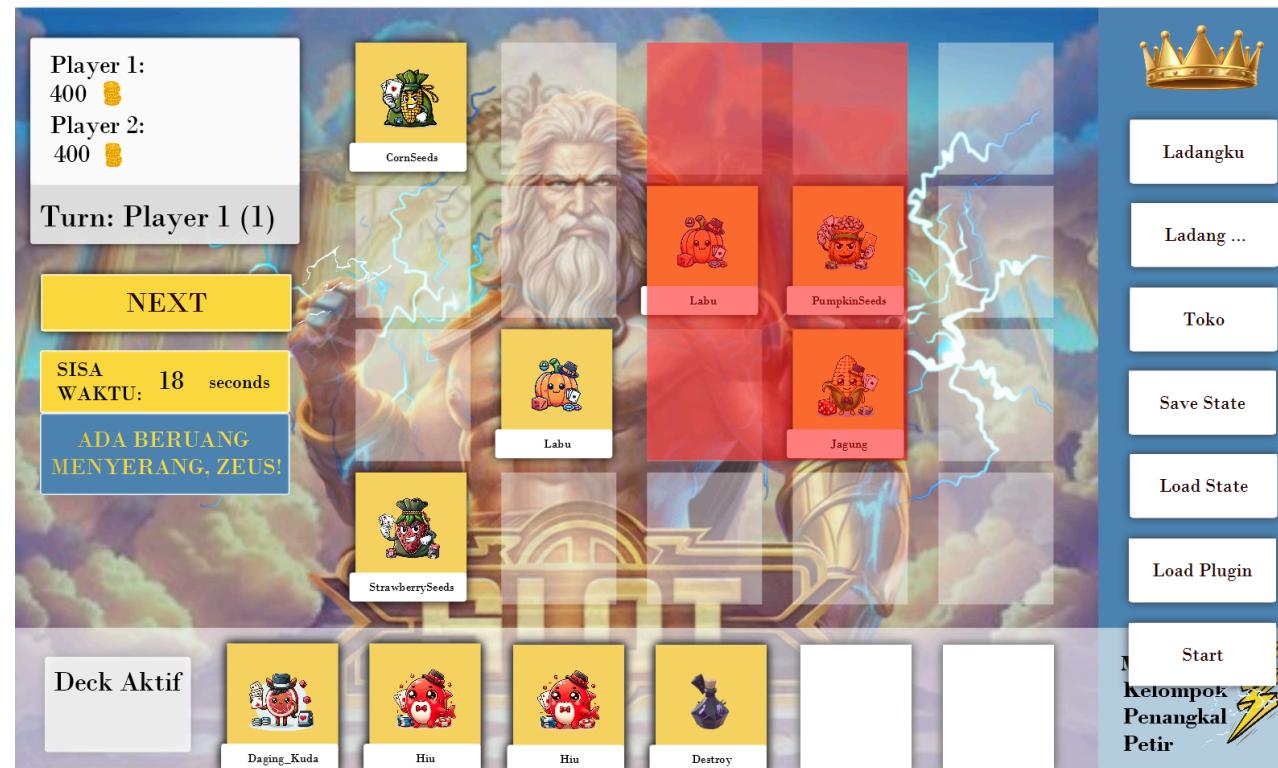
REPOSITORY : https://github.com/dianatrihy/IF2210_TB2_FCK.git

1. Deskripsi Umum Aplikasi

Demi membantu Bondowoso menaklukkan hati Roro (meskipun jin kewalahan), aplikasi ini telah dibuat sebagai program simulasi yang berbasis APG (Antarmuka Pengguna Grafis). Aplikasi ini memiliki fitur dasar menanam tanaman dan memelihara hewan pada petak ladang. Setiap objek pada permainan seperti hewan, tanaman, dan produk dilambangkan sebagai kartu. Kemudian kartu tersebut dapat diletakkan pada ladang ataupun dijual pada toko apabila kartu tersebut berada pada *deck* aktif pemain. Selain itu, objek pada ladang juga dapat dihancurkan bilamana terdapat serangan dari beruang yang terjadi secara random. Program ini juga memiliki fitur save & load untuk menyimpan dan memuat state program pada format txt. Program juga bersifat extensible dengan menyediakan dukungan plugin, sehingga pengguna dapat menambahkan format file untuk save & load selain format txt secara mudah. Dalam permainan ini, terdapat dua pemain yang memiliki ladang dan kartunya masing-masing. Pemenang akan ditentukan dari jumlah uang yang berhasil dikumpulkan oleh setiap pemain setelah 20 Turn.



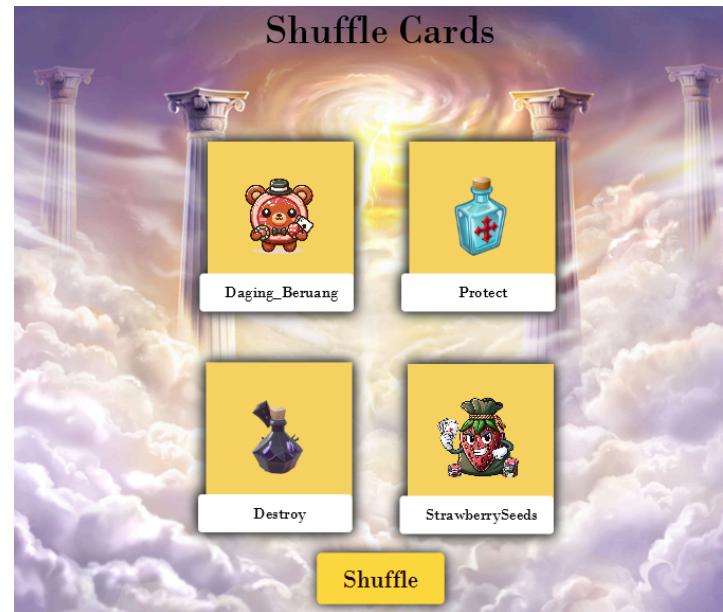
Gambar 1.1 Tampilan Utama



Gambar 1.2 Terjadi Serangan Beruang



Gambar 1.3 Tampilan Toko



Gambar 1.4 Tampilan Shuffle Kartu

2. Kakas GUI: JavaFX

Dalam pembuatan aplikasi, digunakan kakas GUI berupa JavaFX. JavaFX menyediakan API (Application Programming Interface) untuk membuat antarmuka pengguna yang responsif dan modern, khususnya dalam bahasa Java. Kakas ini juga mendukung berbagai komponen GUI seperti tombol, label, tabel, serta memberikan kemampuan untuk membuat efek visual.

Aplikasi JavaFX dimulai dengan menjalankan metode 'main' yang akan memanggil 'application.launch()'. Dalam kelas aplikasi utama, perlu meng-extend kelas javafx.application.Application dan meng-override metode 'start()'. Metode 'start(Stage primaryStage)' adalah tempat untuk mendefinisikan komponen GUI dan mengatur stage utama. Stage sendiri adalah window utama dari suatu aplikasi. Di dalam Stage, ditempatkan Scene yang berisi hirarki dari node (komponen GUI). Di dalam Scene dapat ditambahkan layout (seperti VBox, HBox,

BorderPane) dan komponen UI (seperti Button, Label, TextField). Setelah mengatur scene pada stage, primaryStage.show() dipanggil untuk menampilkan window.

Seperti yang telah dijelaskan, terdapat beberapa komponen utama agar GUI pada JavaFX dapat muncul. Komponen-komponen tersebut antara lain, Stage sebagai representasi dari window aplikasi, Scene sebagai kontainer untuk semua konten visual, Node sebagai superclass dari setiap item pada GUI, serta Layout Pane sebagai kontainer yang digunakan untuk mengatur tata letak Node.

Berbagai komponen lain untuk membangun antarmuka pengguna juga disediakan. Seperti komponen kontrol berupa Button, Label, TextField, TextArea, CheckBox, Slider, dsb. Komponen layout pane juga dapat berupa VBox, HBox, StackPane, GridPane dsb. Serta komponen chart seperti LineChart, BarChart, PieChart maupun komponen media seperti MediaView.

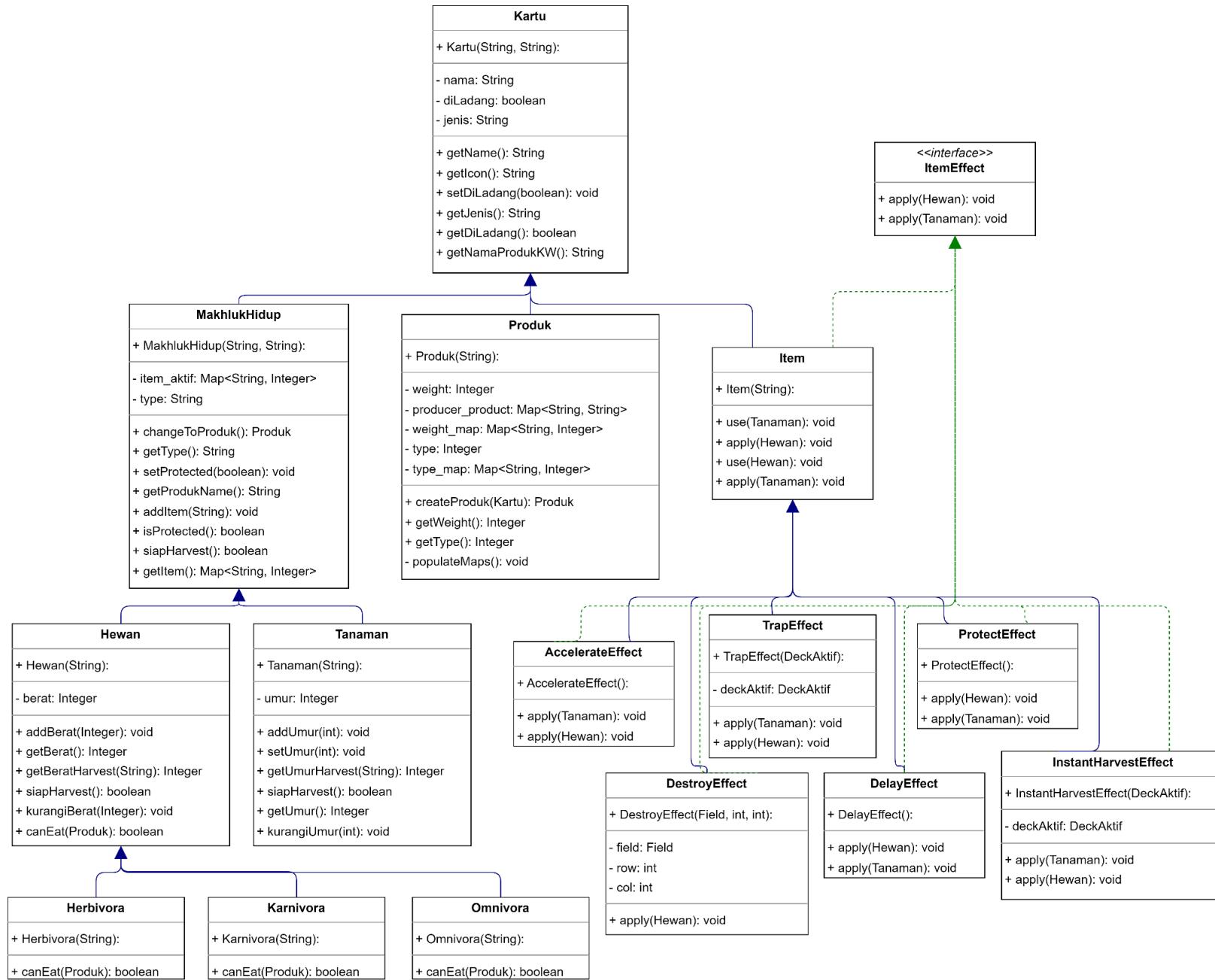
3. Plugin & Class Loader

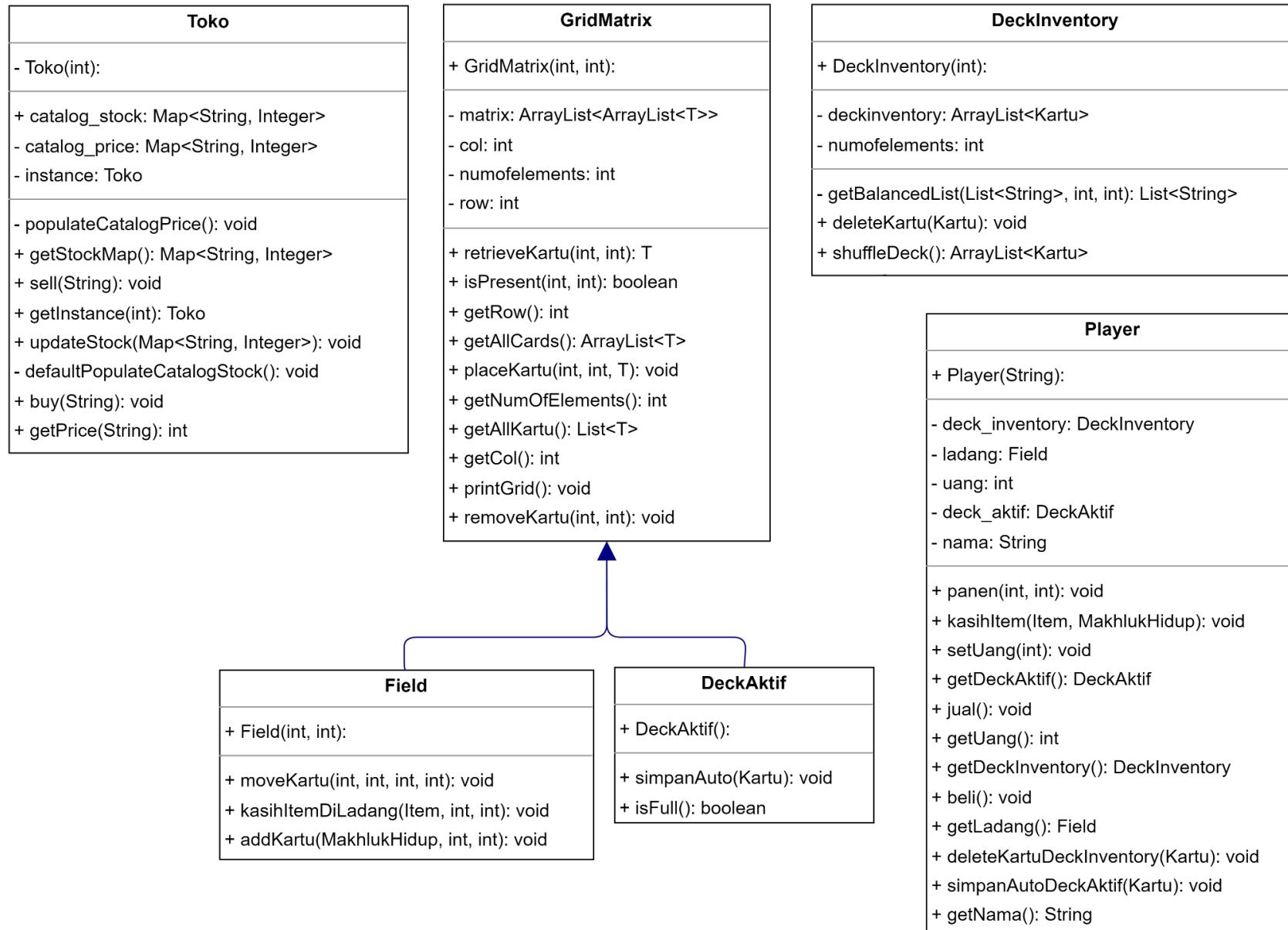
Class Loader adalah bagian dalam JVM yang bertugas untuk memuat kelas ke dalam memori. Secara *default* kelas dapat dimuat dari *filesystem*, file .jar, dan jaringan. Class Loader dapat menciptakan kelas baru pada saat *runtime* atau program utama berjalan. Namun, kelas baru tersebut mungkin tidak dapat digunakan karena program utama tidak mengetahui method dan property apa saja yang terdapat pada kelas baru. Untuk dapat melihat dan mengeksekusi isi sebuah kelas, Java memiliki fitur Reflection API. Dengan menggunakan Reflection API, program dapat memeriksa informasi tentang kelas seperti nama kelas, konstruktor, method, dan variabel. Selain itu juga dapat menciptakan instance objek kelas, memanggil method, mengakses variabel hingga memanipulasi metode dan variabel.

Salah satu pemanfaatan reflection adalah dalam pembuatan fitur plugin. Pertama-tama dibuat kelas IParser() sebagai file loader dari plugin yang akan dimasukkan. Parser tersebut bertugas untuk mengubah berbagai jenis extension file plugin menjadi format string yang sama, sehingga berbagai jenis plugin dapat diproses nantinya. Parser tersebut dipanggil dalam kelas LoadState yang bertugas menangani masukan state ataupun plugin dari luar.

Setelah itu, dibuat sebuah method di dalam kelas Game Manager utama untuk menangani fitur plugin ini. Di dalam method tersebut dimulai dari pengecekan path masukan merupakan masukan yang valid. Selanjutnya melakukan instansiasi dari ClassLoader dan ServiceLoader yang nantinya digunakan sebagai reflection kelas-kelas dan service-service dari plugin. Setiap reflection service yang didapatkan akan diiterasi untuk dimuat sebagai fitur tambahan aplikasi. Plugin yang telah dibuat berupa fitur yang memungkinkan simpan dan muat state program pada format selain txt, yaitu format yaml dan json.

4. Class Diagram





5. Konsep OOP

5.1. Inheritance

- Kelas Karnivora, Herbivora, dan Omnivora (src/main/java/org/example/if2210_tb2_fck/model/Karnivora.java, Herbivora.java, Omnivora.java) merupakan anak dari kelas Hewan (src/main/java/org/example/if2210_tb2_fck/model/Hewan.java), kelas Hewan merupakan anak dari kelas MakhlukHidup (src/main/java/org/example/if2210_tb2_fck/model/MakhlukHidup.java), kelas MakhlukHidup merupakan anak dari kelas Kartu (src/main/java/org/example/if2210_tb2_fck/model/Kartu.java).
- Kelas Produk (src/main/java/org/example/if2210_tb2_fck/model/Produk.java) merupakan anak dari kelas Kartu (src/main/java/org/example/if2210_tb2_fck/model/Kartu.java).
- Kelas AccelerateEffect, DelayEffect, TrapEffect, DestroyEffect, ProtectEffect, dan InstantHarvestEffect (src/main/java/org/example/if2210_tb2_fck/model/Item/AccelerateEffect.java, DelayEffect.java, TrapEffect.java, DestroyEffect.java, ProtectEffect.java, InstantHarvestEffect.java) merupakan anak dari kelas Item, kelas Item (src/main/java/org/example/if2210_tb2_fck/model/Item/Item.java) merupakan anak dari kelas Kartu (src/main/java/org/example/if2210_tb2_fck/model/Kartu.java).
- Kelas Field dan DeckAktif (src/main/java/org/example/if2210_tb2_fck/model/DeckAktif.java) merupakan anak dari kelas GridMatrix (src/main/java/org/example/if2210_tb2_fck/model/GridMatrix.java).

5.2. Composition

- Kelas Player (src/main/java/org/example/if2210_tb2_fck/model/Player.java) memiliki tepat satu kelas DeckInventory (src/main/java/org/example/if2210_tb2_fck/model/DeckInventory.java), Field (src/main/java/org/example/if2210_tb2_fck/model/Field.java), dan DeckAktif (src/main/java/org/example/if2210_tb2_fck/model/DeckAktif.java).
- Kelas DeckInventory (src/main/java/org/example/if2210_tb2_fck/model/DeckInventory.java) memiliki banyak kelas Kartu (src/main/java/org/example/if2210_tb2_fck/model/Kartu.java).
- Kelas DestroyEffect (src/main/java/org/example/if2210_tb2_fck/model/Item/DestroyEffect.java) memiliki tepat satu kelas Field src/main/java/org/example/if2210_tb2_fck/model/Field.java().

- Kelas TrapEffect (src/main/java/org/example/if2210_tb2_fck/model/Item/TrapEffect.java) memiliki tepat satu kelas DeckAktif (src/main/java/org/example/if2210_tb2_fck/model/DeckAktif.java).
- Kelas InstantHarvestEffect (src/main/java/org/example/if2210_tb2_fck/model/Item/InstantHarvestEffect.java) memiliki tepat satu kelas DeckAktif (src/main/java/org/example/if2210_tb2_fck/model/DeckAktif.java).

5.3. Interface

- Interface ItemEffect (src/main/java/org/example/if2210_tb2_fck/model/Item/ItemEffect.java)
- Interface ICommand (src/main/java/org/example/if2210_tb2_fck/command/ICommand.java)
- Interface IParser (src/main/java/org/example/if2210_tb2_fck/parser/IParser.java)
- Interface Plugin (src/main/java/org/example/if2210_tb2_fck/plugin/Plugin.java)

5.4. Method Overriding dan Method Overloading

- Method CanEat pada kelas Herbivora, Karnivora, dan Omnivora (src/main/java/org/example/if2210_tb2_fck/model/Karnivora.java, Herbivora.java, Omnivora.java)
- Method Apply pada kelas Item, AccelerateEffect, DelayEffect, TrapEffect, DestroyEffect, ProtectEffect, dan InstantHarvestEffect (src/main/java/org/example/if2210_tb2_fck/model/Item/Item.java, AccelerateEffect.java, DelayEffect.java, TrapEffect.java, DestroyEffect.java, ProtectEffect.java, InstantHarvestEffect.java)
- Method use pada kelas Item (src/main/java/org/example/if2210_tb2_fck/model/Item/Item.java)
- Method execute pada kelas berbagai command (src/main/java/org/example/if2210_tb2_fck/command)

5.5. Polymorphism

- Kelas GridMatrix () memiliki atribut matrix

5.6. Java API Collection

- Kelas MakhlukHidup (src/main/java/org/example/if2210_tb2_fck/model/MakhlukHidup.java) memiliki Map dengan *key* berupa String dan *value* berupa Integer.
- Kelas DeckInventory (src/main/java/org/example/if2210_tb2_fck/model/DeckInventory.java) memiliki ArrayList of Kartu

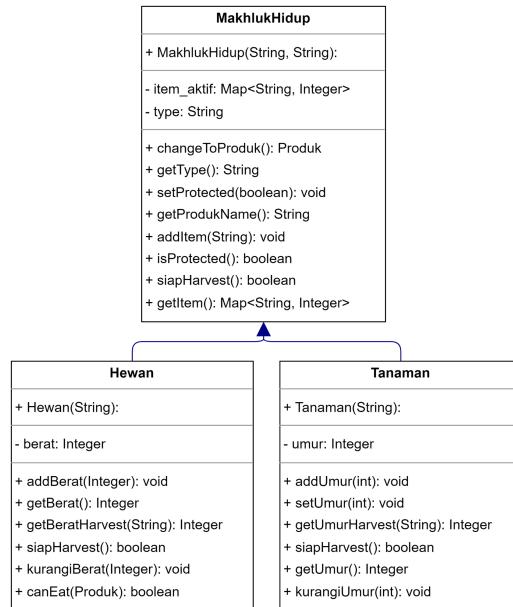
- Kelas Toko (src/main/java/org/example/if2210_tb2_fck/model/Toko.java) memiliki Map dengan *key* berupa String dan *value* berupa Integer.
- Kelas GridMatrix (src/main/java/org/example/if2210_tb2_fck/model/GridMatrix.java) memiliki ArrayList of ArrayList of T.

5.7. SOLID

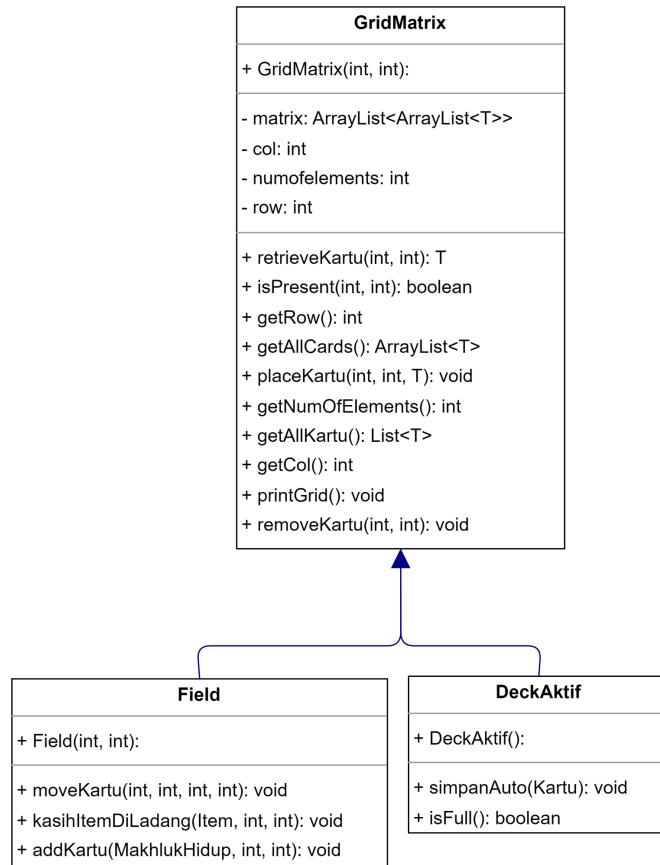
- S: Kelas Toko (src/main/java/org/example/if2210_tb2_fck/model/Toko.java) hanya bertugas untuk mengatur apapun yang disimpan dalam fitur Toko, sedangkan isian dari toko seperti Kartu diatur oleh kelas Kartu (src/main/java/org/example/if2210_tb2_fck/model/Kartu.java)
- O: Menggunakan implementasi interface dengan kelas-kelas seperti kartu item untuk menentukan efek yang akan muncul dari penggunaan kartu tersebut
- L: Salah satu contoh penggunaan prinsip adalah pada kelas jualController dgn penggunaan sebuah list dari kartu dan kemudian memanggil kelas produk yang merupakan subkelas kartu dari list tersebut
- I: Dalam implementasi interface tidak semua kelas harus mengimplementasikan methodnya
- D: Kelas kartu dan subkelasnya seperti makhluk hidup, item, dan produk bergantung pada abstraksi objek

5.8. Design Pattern

- Template Method: MakhlukHidup (src/main/java/org/example/if2210_tb2_fck/model/MakhlukHidup.java) sebagai kelas yang mendefinisikan kerangka kerja dari kelas Hewan (src/main/java/org/example/if2210_tb2_fck/model/Hewan.java) dan Tanaman (src/main/java/org/example/if2210_tb2_fck/model/Tanaman.java). Tujuannya dari design pattern ini untuk mendefinisikan kerangka kerja algoritma secara umum di kelas MakhlukHidup dan membiarkan Hewan dan Tanaman mengimplementasikan detail-detail spesifik dari langkah-langkah tertentu seperti pada method siapHarvest. Kelebihannya adalah dapat mengurangi *redundancy* penulisan kode, jika tidak diterapkan harus mengubah kode pada masing-masing subclass jika terjadi perubahan yang sama.

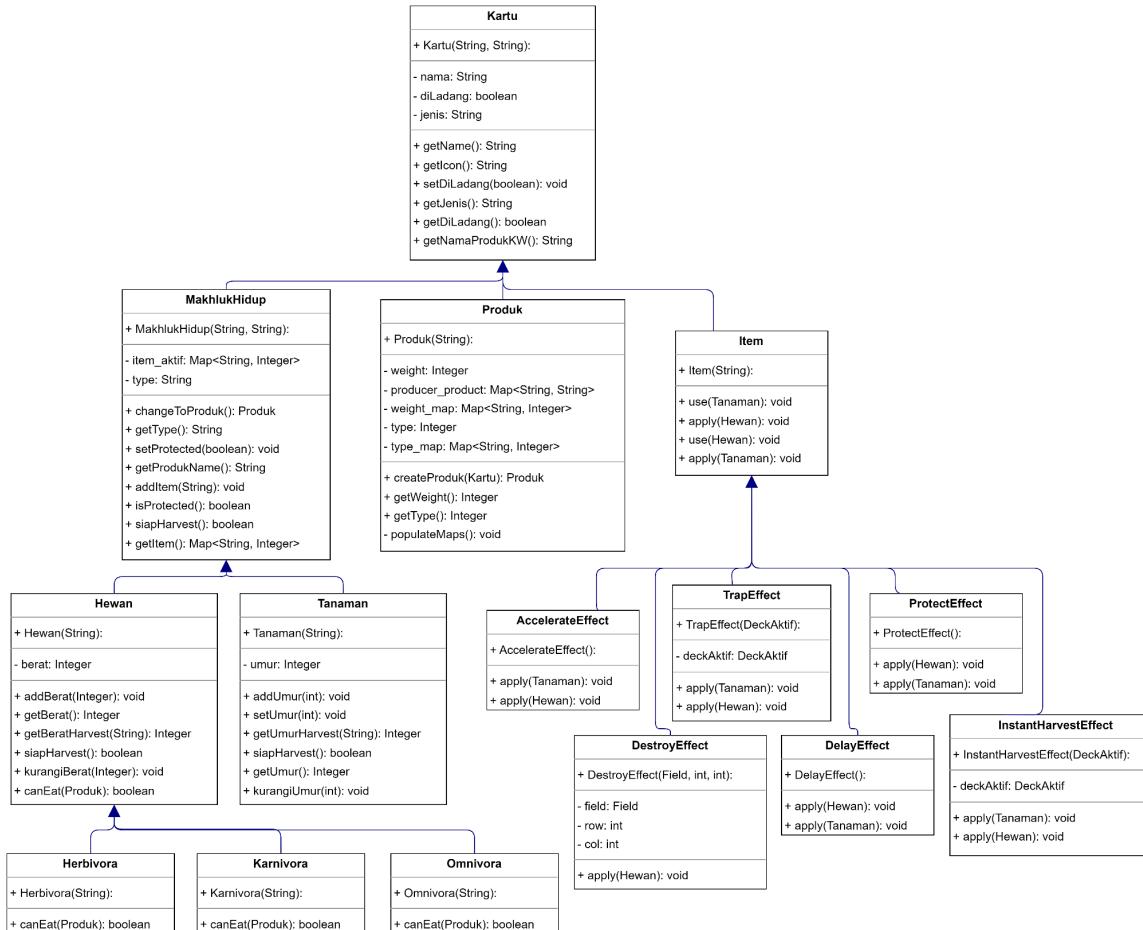


- Factory Method: **GridMatrix** (src/main/java/org/example/if2210_tb2_fck/model/GridMatrix.java) sebagai kelas factory dan menghasilkan kelas **Field** (src/main/java/org/example/if2210_tb2_fck/model/Field.java) dan kelas **DeckAktif** (src/main/java/org/example/if2210_tb2_fck/model/DeckAktif.java). Tujuan diterapkan design pattern ini untuk membuat objek tanpa menentukan kelas spesifik yang dibuat.



- Composite: Kartu (src/main/java/org/example/if2210_tb2_fck/model/Kartu.java) memiliki turunan MakhlukHidup(src/main/java/org/example/if2210_tb2_fck/model/MakhlukHidup.java), Item(), dan Produk (src/main/java/org/example/if2210_tb2_fck/model/Produk.java). Kelas MakhlukHidup memiliki turunan Hewan (src/main/java/org/example/if2210_tb2_fck/model/Hewan.java) dan Tanaman (src/main/java/org/example/if2210_tb2_fck/model/Tanaman.java). Kelas Hewan memiliki turunan Herbivora, Karnivora, Omnivora (src/main/java/org/example/if2210_tb2_fck/model/Herbivora.java, Karnivora.java, Omnivora.java). Kelas Item memiliki turunan berbagai

jenis Item. Sehingga keseluruhan kelas tersebut membentuk hierarki secara komposit. Tujuan diterapkan design pattern ini untuk mengoperasikan seluruh turunan kartu secara seragam.



5.9. Reflection

- Method LoadPlugin pada kelas GameManagerController (src/main/java/org/example/if2210_tb2_fck/controller/GameManagerController.java) menggunakan reflection class loader untuk membuat fitur plugin
-

5.10. Threading

- Threading serangan beruang / BearAttack (src/main/java/org/example/if2210_tb2_fck/command/BearAttack.java) untuk mensinkronisasi waktu penyerangan beruang yang akan mengupdate timer setiap 0.1 detik

6. Bonus Yang dikerjakan

6.1. Mengubah Ukuran Grid

- Mengambil ukuran ladang yang ingin diload
-

6.2. Memperindah UI

User Interface pada aplikasi dibuat dalam satu tema yang sama yaitu kasino (slot judi ~~buat bayar ukt mahal~~). Baik dari background setiap window hingga asset animasi karakter-karakter pada kartu dibuat dalam model dan tema yang sama, sehingga

lebih menarik untuk dimainkan (Roro). Tema ini terinspirasi dari iklan yang selalu muncul pada story Instagram pembuat, serta kemuakan akan desain *aesthetic* dengan warna pastel itu. Selain itu juga karena permainan ini yang juga mengandung unsur keberuntungan dalam mendapat kartu shuffle.

7. Pembagian Tugas

- 13522012 / Thea Josephine Halim
 - Membuat design GUI
 - Mekanisme serangan beruang
 - Kelas dan Command yang berhubungan dengan Ladang dan Deck
- 13522046 / Raffael Boymian Siahaan
 - Mekanisme GUI Drag and Drop
 - Kelas dan Command yang berhubungan dengan Item
- 13522062 / Salsabiila
 - Membuat kelas dan command toko dan handling yang berhubungan dengan toko
 - Mekanisme serangan beruang
- 13522096 / Novelya Putri Ramadhani
 - Kelas Player
 - Mekanisme Plugin
- 13522104 / Diana Tri Handayani
 - Set up spesifikasi sistem
 - Kelas dan Command yang berhubungan dengan Kartu dan MakhlukHidup serta turunannya
 - Laporan
- 10023510 / M Teguh Wijaksono
 - Datang asistensi habis tu ngilang

8. Foto Kelompok



Dokumentasi kelompok pose mengerjakan tugas besar bersama (yang ngilang gausah diajak foto).

Lampiran Asistensi

Kode Kelompok : FCK

Nama Kelompok : Penangkal Petir

1. 10023510 / M Teguh Wijaksono
2. 13522012 / Thea Josephine Halim
3. 13522046 / Raffael Boymian Siahaan
4. 13522062 / Salsabiila
5. 13522096 / Novelya Putri Ramadhani
6. 13522104 / Diana Tri Handayani

Asisten Pembimbing : 13520105 / Malik Akbar Hashemi Rafsanjani

1. Konten Diskusi

GUI

- ada xml, styling, proglang ngendaliin xmlnya
- behavior dari xml manggil controller
- udh ada komponennya
- javafx scene builder bisa generate xml, styling tidak masalah

IDE

- jangan selain IntelliJ IDEA karena ribet
- akun std bisa ultimate opsional
- maven tinggal pencet, clean, compile, gak perlu mikir commandnya
- main program sama controller buat xml
- JAR pake package kyknya perlu bljr command
- kalo nambah-nambah dependencies/lib, configure di pom.xml/build.gradle

TAHAPAN SERANGAN BERUANG

- perlu belajar multithreading
- pake multithreading terutama buat update time
- cegah race condition pake sync
- hati-hati miss beberapa cases

PLUGIN

- di application, bikin interface yg dipake di keseluruhan
- plugin nambahin fungsionalitas selama runtime seperti vscode extension PAKE Java Reflection
- bikin jar lain, diupload (tiba-tiba aplikasi ada tambahan fungsionalitas)
- !!!! harus pake Open Closed principle serta Reflection !!!!

DESIGN PATTERN !!!

- se bisa mungkin pakai
- refactoring.guru ada contoh kode
- contoh func reactivity pas click suatu button komponen lain berubah -> pake Observer
- pake sebanyak mungkin

UNIT TESTING

- pake GUnit (?)
- tiap fungsi lewat if else nambah coverage

JAR

- bisa dijalankan di Linux

MAIN PROGRAM

- load main xml

TIPS

- pisah java class yg buat controller sama logic
- kelas ui nge init nonui/logic (contoh di PlayerBoardController)
- Unit testing single responsibility, 1 kelas cuma bisa modif atributnya sendiri
- jangan reinvent the wheel
- jangan spam sync
- dibanding exception, pakai result, err, if err ==

2. Tindak Lanjut

- Memutuskan untuk menggunakan maven dan javafx, karena ada scenebuilder untuk memudahkan membuat fxml.
- Mempelajari design pattern lebih lanjut
- Hal yang perlu diperhatikan secara lebih adalah fitur plugin dan multithreading pada serangan beruang
- Semua anggota harus segera coba IntelliJ untuk membiasakan

3. Foto Dokumentasi

Lupa foto kak waktu itu..