

Kelompok : BRO

1. 13522014 / Raden Rafly Hanggaraksa Budiarto
2. 13522057 / Moh Fairuz Alaudding Yahya
3. 13522066 / Nyoman Ganadipa Narayana
4. 13522084 / Dhafin Fawwaz Ikramullah
5. 13522095 / Rayhan Fadhlwan Azka

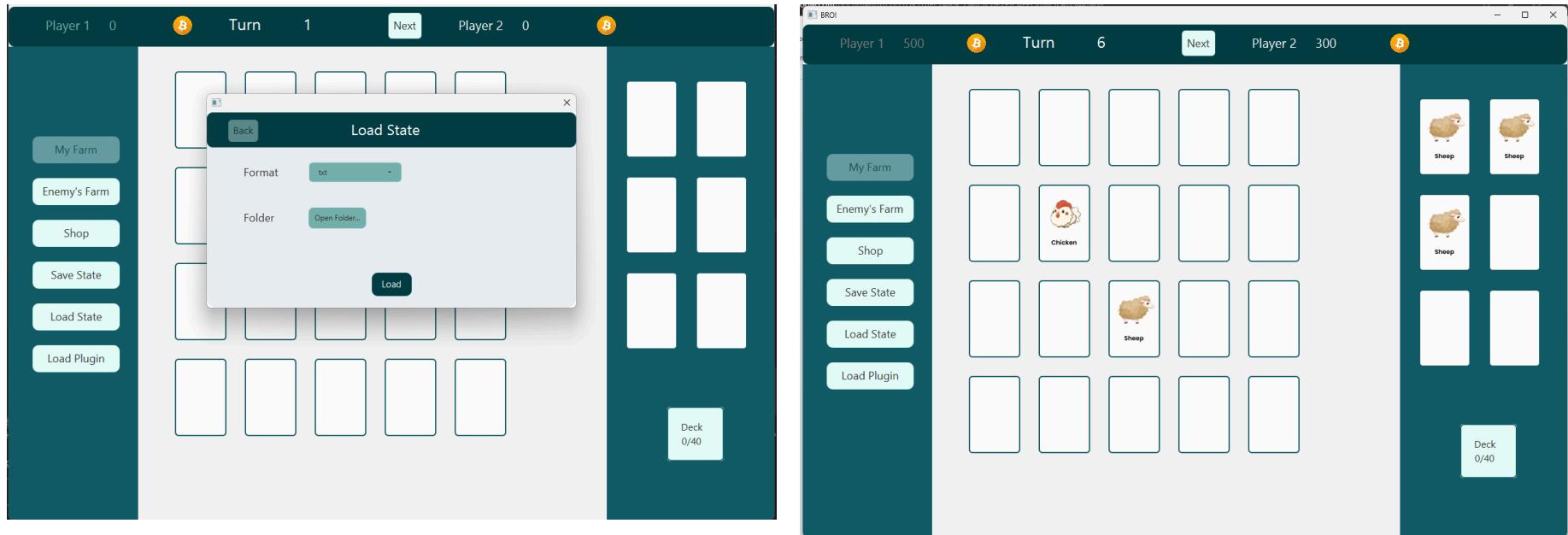
Asisten Pembimbing : Malik Akbar Hashemi Rafsanjani

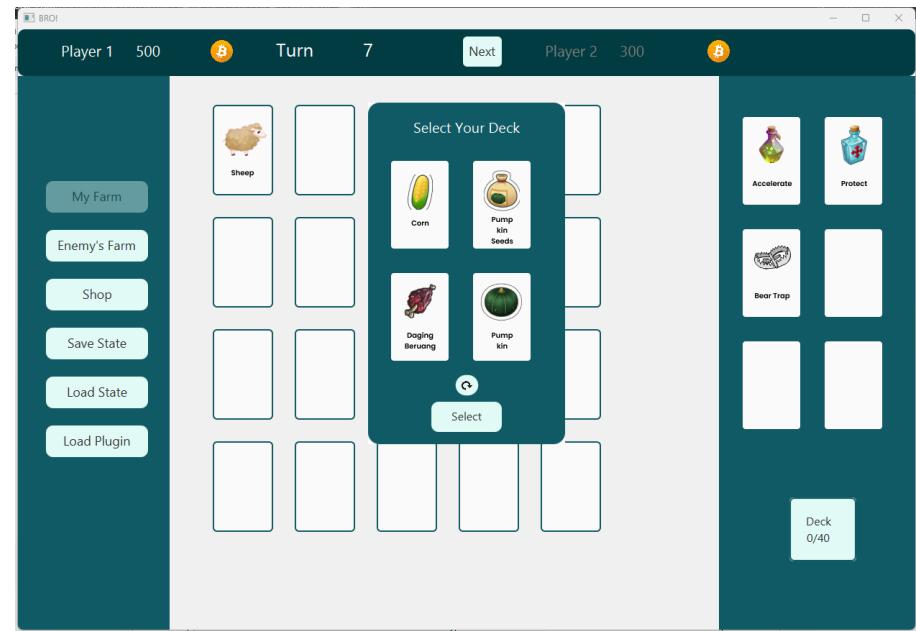
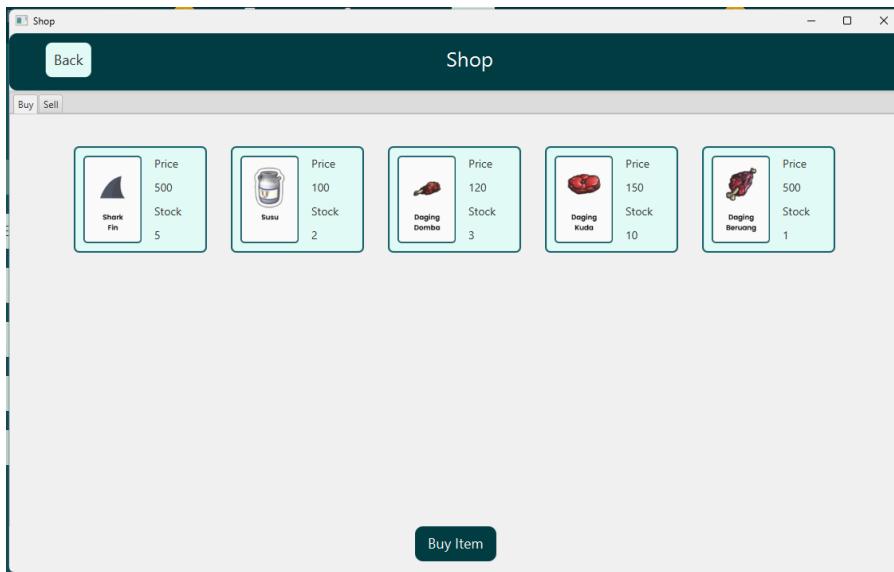
1. Deskripsi Umum Aplikasi

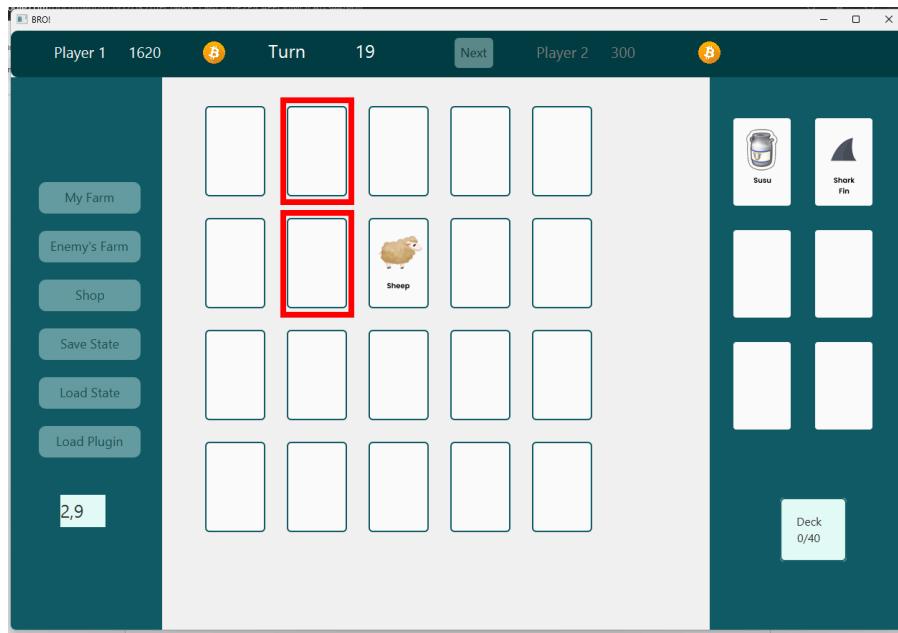
Program ini merupakan program untuk pengelolaan ladang. Ladang sendiri terdiri atas beberapa petak yang berbentuk seperti halnya matriks, di mana setiap petaknya dapat ditanami tanaman atau ditempatkan hewan. Program memiliki fitur dasar seperti penanaman tanaman dan pemeliharaan hewan pada petak ladang. Program ini mirip seperti permainan kartu di mana setiap kartu melambangkan sebuah objek misalnya tanaman atau hewan. Nantinya setiap kartu dapat diletakkan pada ladang ataupun dijual pada toko bila 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. Terdapat 2 pemain pada permainan ini yang memiliki ladang dan kartunya masing-masing. Pemenang akan ditentukan dari jumlah uang yang berhasil dikumpulkan oleh setiap pemain setelah 20 Turn.

Aplikasi dapat mengalami bug apabila dijalankan melalui JAR. Disarankan lakukan run melalui command.







2. Kakas GUI: JavaFX

Dalam membuat aplikasi ini, Kakas GUI yang kami gunakan adalah **JavaFX**. JavaFX adalah platform perangkat lunak untuk membuat dan menyampaikan aplikasi desktop, serta aplikasi web kaya yang dapat berjalan di berbagai macam perangkat. JavaFX mendukung komputer desktop di Microsoft Windows, Linux (termasuk Raspberry Pi), dan macOS, serta perangkat seluler yang menjalankan iOS dan Android, melalui Gluon Mobile.

JavaFX memiliki siklus hidup aplikasi yang terdiri dari beberapa tahap utama: inisialisasi, mulai, berjalan, dan penghentian. Tahap inisialisasi diawali dengan pemanggilan metode `init()`, yang digunakan untuk menyiapkan sumber daya atau melakukan pengaturan awal sebelum aplikasi diluncurkan. Selanjutnya, metode `start(Stage primaryStage)` dipanggil setelah inisialisasi selesai. Pada tahap ini, antarmuka pengguna (UI) didefinisikan dan jendela utama aplikasi ditampilkan. `Stage` berfungsi sebagai wadah utama untuk elemen-elemen UI, sementara `Scene` menampung konten yang akan ditampilkan dalam `Stage`.

Setelah UI siap dan aplikasi berjalan, JavaFX memasuki tahap berjalan di mana aplikasi merespons interaksi pengguna melalui loop event JavaFX. Pada tahap ini, aplikasi berfungsi penuh dan menangani input pengguna serta memperbarui UI sesuai kebutuhan. Ketika aplikasi hendak dihentikan, metode `stop()` dipanggil untuk membersihkan sumber daya yang digunakan oleh aplikasi sebelum keluar. Ini memastikan bahwa semua proses yang berjalan di latar belakang dihentikan dengan benar dan memori yang digunakan dibebaskan. Siklus hidup ini memberikan kerangka kerja yang terstruktur untuk mengembangkan aplikasi JavaFX yang efisien dan responsif.

Dalam projek ini, kami menggunakan beberapa komponen natif yang dimiliki oleh JavaFX sebagai basis dari komponen - komponen yang akan kami buat. Komponen natif yang kami gunakan adalah Label, Button, TilePane, ChoiceBox, List, Tab, TabPane, dan TilePane.

JavaFX menyediakan berbagai komponen yang memungkinkan pengembang untuk membangun antarmuka pengguna (UI) yang kaya dan interaktif. Berikut adalah beberapa komponen utama yang disediakan oleh JavaFX:

Komponen Dasar	Kontrol	<ul style="list-style-type: none"> • Button: Tombol interaktif yang dapat diklik untuk memicu tindakan. • Label: Komponen untuk menampilkan teks statis. • TextField: Bidang teks untuk input satu baris. • TextArea: Bidang teks untuk input multi-baris. • PasswordField: Bidang teks yang menyembunyikan input, biasanya untuk kata sandi. • CheckBox: Kotak centang untuk memilih atau tidak memilih opsi.
----------------	---------	--

		<ul style="list-style-type: none"> RadioButton: Tombol radio untuk memilih satu opsi dari sekumpulan pilihan. ComboBox: Kotak dropdown untuk memilih satu opsi dari daftar. ListView: Daftar elemen yang dapat dipilih. TableView: Tabel untuk menampilkan data dalam bentuk baris dan kolom. TreeView: Tampilan pohon untuk menampilkan data hierarkis.
	Layout Panes	<ul style="list-style-type: none"> HBox: Mengatur elemen anak secara horizontal. VBox: Mengatur elemen anak secara vertikal. BorderPane: Mengatur elemen anak dalam lima area: atas, bawah, kiri, kanan, dan tengah. GridPane: Mengatur elemen anak dalam grid (baris dan kolom). StackPane: Menumpuk elemen anak satu di atas yang lain. FlowPane: Mengatur elemen anak dalam urutan aliran, baik horizontal maupun vertikal.
Komponen Lanjutan	Menus	<ul style="list-style-type: none"> MenuBar: Baris menu di bagian atas jendela aplikasi. Menu: Menu dalam MenuBar yang mengandung item menu. MenuItem: Item dalam Menu yang dapat dipilih.
	Dialogs	<ul style="list-style-type: none"> Alert: Dialog untuk menampilkan pesan informasi, peringatan, atau konfirmasi. Dialog: Dialog kustom yang dapat dikonfigurasi sesuai kebutuhan.
	Media	<ul style="list-style-type: none"> MediaView: Komponen untuk menampilkan dan mengontrol media seperti video dan audio.
	Charts	<ul style="list-style-type: none"> LineChart: Diagram garis untuk menampilkan data seri waktu atau kategori. BarChart: Diagram batang untuk membandingkan berbagai kategori. PieChart: Diagram lingkaran untuk menampilkan bagian dari keseluruhan.
	Canvas	<ul style="list-style-type: none"> Canvas: Area gambar yang dapat digunakan untuk menggambar grafis kustom menggunakan API grafis langsung.
	Web	<ul style="list-style-type: none"> WebView: Komponen untuk menampilkan konten web menggunakan mesin rendering WebKit.

3. Plugin & Class Loader

Dalam implementasi plugin, File JAR yang akan dijadikan plugin harus memiliki implementasi dari **interface StatePlugin**. Interface ini memiliki prosedur load yang menerima parameter file yang akan di-load serta state dari game.

StatePlugin.java

```
public interface StatePlugin {  
    void Load(File gameStateFile, File player1File, File player2File, GameState state) throws Exception;  
}
```

Interface dari *StatePlugin* akan dijadikan sebagai atribut dari kelas *StateLoader*. Fungsi dari kelas ini adalah untuk membaca file dari hasil load.

StateLoader.java

```
public class StateLoader {  
    String path;  
    String player1FileName;  
    String player2FileName;  
    String gameStateFileName;  
    StatePlugin plugin;  
    ...  
}
```

Selanjutnya, kelas StateLoader akan menginisialisasi plugin serta path dari file yang akan dibacanya.

MainController.java

```
...
@FXML
public void initialize() {
    StateLoader loader = new StateLoader();
    state = loader.setPath("state", "gamestate.txt", "player1.txt", "player2.txt")
        .setPlugin(new TextLoader())
        .loadState();
...
}
```

Pada contoh diatas, digunakan Plugin bawaan yaitu *TextLoader* yang digunakan untuk membaca file dalam bentuk txt. Plugin dapat diubah - ubah hanya dengan mengganti argumen dari *setPlugin*. Kami telah membuat JAR file yang berperan sebagai plugin untuk membaca JSON dan XML .Namun, **kami tidak berhasil untuk membaca file JAR secara dinamis** sehingga kami tidak dapat mengimplementasikan load file dalam bentuk lain.

4. Class Diagram

Berikut merupakan kelas diagram dari program.

https://drive.google.com/file/d/1w_VUL0pIS4Ysh4Arevo5j1AWVwsIWT5t/view?usp=sharing

Program pada awalnya diinisialisasi melalui kelas StateLoader, yang mengurus penulisan atau penyimpanan state dari program ke dalam file eksternal. Hasil dari StateLoader ini kemudian dimasukkan ke dalam kelas GameState, yang mengawasi seluruh komunikasi yang dilakukan oleh setiap kelas controller. Kelas yang berperan sebagai entitas yang bisa digeser adalah Kelas DraggableItem yang menginherit ImageView. Kelas ini selanjutnya akan menjadi superclass dari Kelas Card dan Empty Card, sedangkan Kelas Card akan menjadi superclass bagi Kelas CreatureCard, ItemCard, dan ProductCard.

Kelas Carnivore, Herbivore, dan Omnivore merupakan anak kelas dari Kelas Animal. Kelas Animal dan Kelas Plant merupakan anak kelas dari Kelas Creature. Kelas ProductAnimal dan ProductPlant merupakan anak kelas dari Kelas Product. Kelas Creature, Product, dan Item merupakan anak kelas dari Resource.

Untuk kelas GUI, Kelas MainController berperan sebagai fungsional dari event - event yang terjadi pada GUI. Kelas ini merupakan entry point dari program ini yang akan menjalankan fungsionalitas dari program.

5. Konsep OOP

5.1. Inheritance

- Kelas Carnivore (src/main/java/org/bro/tubesoop2/animal/Carnivore.java), Herbivore (src/main/java/org/bro/tubesoop2/animal/Herbivore.java), Omnivore (src/main/java/org/bro/tubesoop2/animal/Omnivore.java) merupakan anak kelas dari Kelas Animal (src/main/java/org/bro/tubesoop2/animal/Animal.java).
- Kelas Animal (src/main/java/org/bro/tubesoop2/animal/Animal.java) dan Kelas Plant (src/main/java/org/bro/tubesoop2/plant/Plant.java) merupakan anak kelas dari Kelas Creature (src/main/java/org/bro/tubesoop2/creature/Creature.java).
- Kelas ProductAnimal (src/main/java/org/bro/tubesoop2/product/ProductAnimal.java) dan ProductPlant (src/main/java/org/bro/tubesoop2/product/ProductPlant.java) merupakan anak kelas dari Kelas Product (src/main/java/org/bro/tubesoop2/product/Product.java).
- Kelas Creature (src/main/java/org/bro/tubesoop2/creature/Creature.java), Product (src/main/java/org/bro/tubesoop2/product/Product.java), dan Item (src/main/java/org/bro/tubesoop2/item/Item.java) merupakan anak kelas dari Resource (src/main/java/org/bro/tubesoop2/resource/Resource.java).

5.2. Composition

- Kelas Creature (src/main/java/org/bro/tubesoop2/creature/Creature.java) memiliki banyak Kelas Product (src/main/java/org/bro/tubesoop2/product/Product.java) dan Item (src/main/java/org/bro/tubesoop2/item/Item.java).
- Kelas Player (src/main/java/org/bro/tubesoop2/player/Player.java) memiliki banyak Kelas Resource (src/main/java/org/bro/tubesoop2/resource/Resource.java).
- Kelas SeranganBeruang (src/main/java/org/bro/tubesoop2/seranganberuang/SeranganBeruang.java) memiliki banyak Kelas Resource (src/main/java/org/bro/tubesoop2/resource/Resource.java).
- Kelas Toko (src/main/java/org/bro/tubesoop2/toko/Toko.java) memiliki banyak Kelas Resource (src/main/java/org/bro/tubesoop2/resource/Resource.java).

5.3. Interface

- Interface StatePlugin (src/main/java/org/bro/tubesoop2/state/StatePlugin.java).

5.4. Method Overriding dan Method Overloading

- Method load pada Kelas TextLoader (src/main/java/org/bro/tubesoop2/state/TextLoader.java).
- Method save pada Kelas TextLoader (src/main/java/org/bro/tubesoop2/state/TextLoader.java).
- Fungsi getMessage pada BuyingFromShopNotSuccessfullException, ItemShopNotFoundException, ItemShopNotEqualException, ItemShopEmptyException, UangTidakCukupShopException, StockTidakCukupShopException, StockTidakCukupPlayer, BeliOutOfRange, PenyimpananTidakCukup (src/main/java/org/bro/tubesoop2/toko/TokoException.java).

5.5. Polymorphism

Tuliskan daftar penggunaan konsep ini di aplikasi, misalnya:

- Method dragDoneAction pada Card, CreatureCard, EmptyCard, ItemCard, ProductCard dimana masing masing memiliki behavior yang berbeda saat dilepas dimana akan di set menjadi empty card maupun handling jika terdapat catch suatu error
- Metode handleDrop menangani logika ketika sebuah kartu menerima kartu lain yang dijatuhkan di atasnya. Pada kelas Card, metode ini menyediakan logika umum untuk menangani drop. Pada CreatureCard, metode ini memeriksa apakah kartu makhluk dapat menerima kartu lain dan mengubah behaviornya berdasarkan kartu yang menjatuhkannya. Pada EmptyCard, metode ini mengubah status kartu dari kosong menjadi terisi dengan kartu yang valid. Pada ItemCard, metode ini memeriksa kompatibilitas item yang dijatuhkan dan mengubah behaviornya sesuai. Pada ProductCard, metode ini memastikan produk yang diterima valid dan mengubah status kartu berdasarkan produk yang diterima.
- Metode dropEnemy menangani logika ketika sebuah kartu didrop di ladang lawan. Pada CreatureCard, metode ini memeriksa apakah dapat diterima di ladang lawan dan mungkin mengembalikannya ke posisi semula jika tidak diizinkan.
- Metode eat menangani behavior hewan dalam memakan produk tertentu. Pada carnivore, metode ini memastikan bahwa hewan karnivora hanya dapat memakan daging atau produk hewani lainnya. Pada herbivore, metode ini memastikan hewan herbivora hanya memakan produk tanaman. Pada omnivore, metode ini memungkinkan hewan omnivora memakan baik daging maupun tanaman, dan behaviornya berubah berdasarkan jenis produk yang dikonsumsi.
- Metode ConsumeBy memberikan efek berbeda ketika dikonsumsi oleh Creature. Pada Accelerate, metode ini menambah umur pada tanaman dan weight pada hewan. Pada Delay, metode ini menunda umur atau weight. Pada Destroy, metode ini menghapus deck lawan. Pada InstantHarvest, metode ini memberikan efek panen instan dari produk atau hasil yang dihasilkan oleh hewan atau

tumbuhan. Pada Protect, metode ini memberikan perlindungan terhadap hewan yang mengonsumsinya dari bahaya atau serangan. Pada Trap, metode ini menjebak atau menonaktifkan hewan yang mengonsumsinya, membatasi gerakan atau kemampuan mereka.

5.6. Java API Collection

- Method getFlattened pada Kelas Grid (src/main/java/org/bro/tubesoop2/grid/Grid.java) memanfaatkan java API Collection.
- Konstruktor Kelas Location (src/main/java/org/bro/tubesoop2/grid/Location.java) memanfaatkan java API StringBuilder..
- Method getFormattedName pada Kelas Resource (src/main/java/org/bro/tubesoop2/resource/Resource.java) memanfaatkan java API StringBuilder.
- Method generateAffectedIndex pada Kelas SeranganBeruang (src/main/java/org/bro/tubesoop2/seranganberuang/SeranganBeruang.java) memanfaatkan java API Collection.
- Kelas StateLoader (src/main/java/org/bro/tubesoop2/state/StateLoader.java) memanfaatkan java API ServiceLoader.

5.7. SOLID

- S: Kelas StateLoader (src/main/java/org/bro/tubesoop2/state/StateLoader.java) hanya berperan untuk menulis dan memuat state dari file luaran, sedangkan State dalam program diatur oleh Kelas GameState.
- S: Kelas Controller berperan untuk mengatur fungsionalitas GUI, sedangkan keadaan aplikasi sudah diawasi oleh Kelas GameState (src/main/java/org/bro/tubesoop2/state/GameState.java).

- O: Kelas Protect, Trap, InstantHarvest, Acceleration, Delay, Destroy (src/main/java/org/bro/tubesoop2/item) yang menginherit kelas Item menggunakan prinsip *open-closed principle* (module `org.bro.tubesoop2.item.Item`) karena apabila ada penambahan suatu item baru, kode tidak perlu memodifikasi kode yang menghandle consume item.
- L: Kelas Animal akan menggantikan Kelas Herbivore, Omnivore, dan Carnivore dalam implementasi item (src/main/java/org/bro/tubesoop2/animal).
- I: Kelas TextLoader mengimplementasikan interface yang sudah disegregasikan menjadi StatePlugin (src/main/java/org/bro/tubesoop2/state).
- D: Kelas StateLoader menggunakan Interface StatePlugin untuk melakukan dependensinya kepada Kelas TextLoader (src/main/java/org/bro/tubesoop2/state).

5.8. Design Pattern

1. Factory Pattern

ResourceFactory (src/main/java/org/bro/tubesoop2/resource/ResourceFactory.java) sebagai kelas factory berperan untuk menghasilkan Kelas Resource (src/main/java/org/bro/tubesoop2/resource/Resource.java) dengan tujuan untuk memudahkan penggantian dan perluasan kode. Hal ini memastikan kekonsistennan objek yang dibuat untuk seluruh program. Apabila tidak digunakan, pemanggilan konstruktor dari objek harus ditulis atributnya dan harus memerhatikan apabila terjadi perubahan dalam penggunaan kelas tersebut.



ResourceFactory.java

```
public class ResourceFactory {
    Map<String, Supplier<Resource>> resourceMap = new HashMap<>();

    public ResourceFactory(){
        // Product
        resourceMap.put("SIRIP_HIU", () -> new ProductAnimal("SIRIP_HIU", 12, 500));
        resourceMap.put("SUSU"      , () -> new ProductAnimal("SUSU", 4, 100));
        resourceMap.put("DAGING_DOMBA", () -> new ProductAnimal("DAGING_DOMBA", 6, 120));
        resourceMap.put("DAGING_KUDA", () -> new ProductAnimal("DAGING_KUDA", 8, 150));
        resourceMap.put("TELUR", () -> new ProductAnimal("TELUR", 2, 50));
        resourceMap.put("DAGING_BERUANG", () -> new ProductAnimal("DAGING_BERUANG", 12, 500));
        resourceMap.put("JAGUNG", () -> new ProductPlant("JAGUNG", 3, 150));
        resourceMap.put("LABU", () -> new ProductPlant("LABU", 10, 500));
        resourceMap.put("STROBERI", () -> new ProductPlant("STROBERI", 5, 350));

        // Animal
        resourceMap.put("HIU_DARAT", () -> new Carnivore("HIU_DARAT", 25, (Product)get("SIRIP_HIU")));
        resourceMap.put("SAPI", () -> new Herbivore("SAPI", 10, (Product)get("SUSU")));
        resourceMap.put("DOMBA", () -> new Herbivore("DOMBA", 12, (Product)get("DAGING_DOMBA")));
        resourceMap.put("KUDA", () -> new Herbivore("KUDA", 14, (Product)get("DAGING_KUDA")));
        resourceMap.put("AYAM", () -> new Omnivore("AYAM", 5, (Product)get("TELUR")));
        resourceMap.put("BERUANG", () -> new Omnivore("BERUANG", 25, (Product)get("DAGING_BERUANG")));

        // Tanaman
        resourceMap.put("BIJI_LABU", () -> new Plant(3, "BIJI_LABU", (Product)get("LABU")));
        resourceMap.put("BIJI_JAGUNG", () -> new Plant(5, "BIJI_JAGUNG", (Product)get("JAGUNG")));
        resourceMap.put("BIJI_STROBERI", () -> new Plant(4, "BIJI_STROBERI", (Product)get("STROBERI")));
    }
}
```

```
resourceMap.put("ACCELERATE", Accelerate::new);
resourceMap.put("DELAY", Delay::new);
resourceMap.put("INSTANT_HARVEST", InstantHarvest::new);
resourceMap.put("DESTROY", Destroy::new);
resourceMap.put("PROTECT", Protect::new);
resourceMap.put("TRAP", Trap::new);

}

public Resource get(String key){
    System.out.println(key);
    Supplier<Resource> s = resourceMap.get(key);
    return s.get();
}

public Set<String> getAllResourceKeys() {
    return resourceMap.keySet();
}
}
```

2. Observer Pattern

Action (*src/main/java/org/bro/tubesoop2/action/Action.java*) sebagai kelas publisher berperan sebagai publisher yang akan menotifikasi Consumer atas event yang terjadi padanya. Observer Pattern memungkinkan subject untuk memberi tahu observer tentang perubahan tanpa mengetahui siapa observer tersebut atau apa yang mereka lakukan. Hal ini mengakibatkan adanya peningkatan modularitas dan mengurangi ketergantungan antara komponen.



Action.java

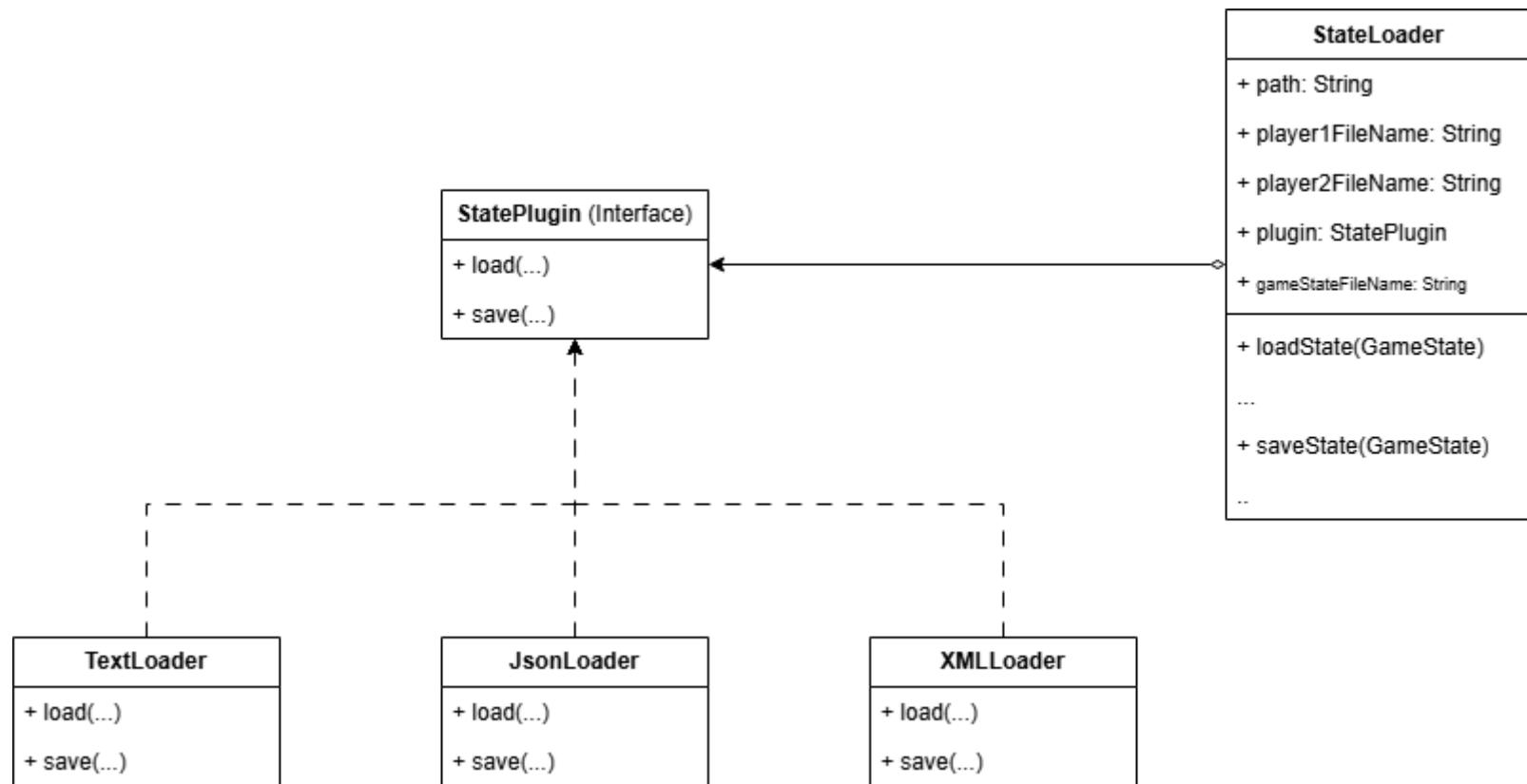
```
public class Action<T> {
    public ArrayList<Consumer<T>> list;
    public Action(){
        list = new ArrayList<>();
    }
    public void AddListener(Consumer<T> r){
        list.add(r);
    }
    public void RemoveListener(Consumer<T> r){
        list.remove(r);
    }

    public void Notify(T t){
        list.forEach(c -> {
            c.accept(t);
        });
    }
}
```

```
}
```

3. Adapter Pattern

StateLoader (*src/main/java/org/bro/tubesoop2/state/StateLoader.java*) berperan sebagai client yang akan menggunakan interface StatePlugin (*src/main/java/org/bro/tubesoop2/state/StatePlugin.java*) sebagai adapter dalam membaca file yang berformat berbeda - beda. Hal ini memungkin program untuk melakukan load file untuk berbagai macam jenis file. Adapter Pattern ini diterapkan dalam membuat plugin. Untuk implementasi, silahkan rujuk bab *Plugin & Class Loader*.



5.9. Reflection

- Method loadPluginService pada Kelas StateLoader (*src/main/java/org/bro/tubesoop2/state/StateLoader.java*) menggunakan reflection ServiceLoader untuk mengakses konten JAR secara dinamis.

5.10. Threading

- Threading CountdownTimer (*src/main/java/org/bro/tubesoop2/countdowntimer/CountdownTimer.java*) untuk timer pada serangan beruang agar pada saat serangan beruang pemain dapat menggunakan program.

6. Pembagian Tugas

NIM	Nama	Tugas
13522014	Raden Rafly Hanggaraksa Budiarto	1. Mengurus Kelas Shop & Shop Controller. 2. Mengurus GUI enemy dan my Ladang & Deck. 3. Setup javaFX. 4. Mengurus Kelas SeranganBeruang. 5. Mengurus Kelas Dasar. 6. Laporan. 7. Mengurus design pattern. 8. Mengurus load & save. 9. Mengurus musik.
13522057	Moh Fairuz Alauddin Yahya	
13522066	Nyoman Ganadipa Narayana	
13522084	Dhafin Fawwaz Ikramullah	
13522095	Rayhan Fadhlwan Azka	

7. Foto Kelompok



Link Form Asistensi :

https://docs.google.com/document/d/1OqOZGCazgoSmkBTxx1wmmg__etcUFn8E/edit?usp=sharing&ouid=105151447910772998653&rtpof=true&sd=true