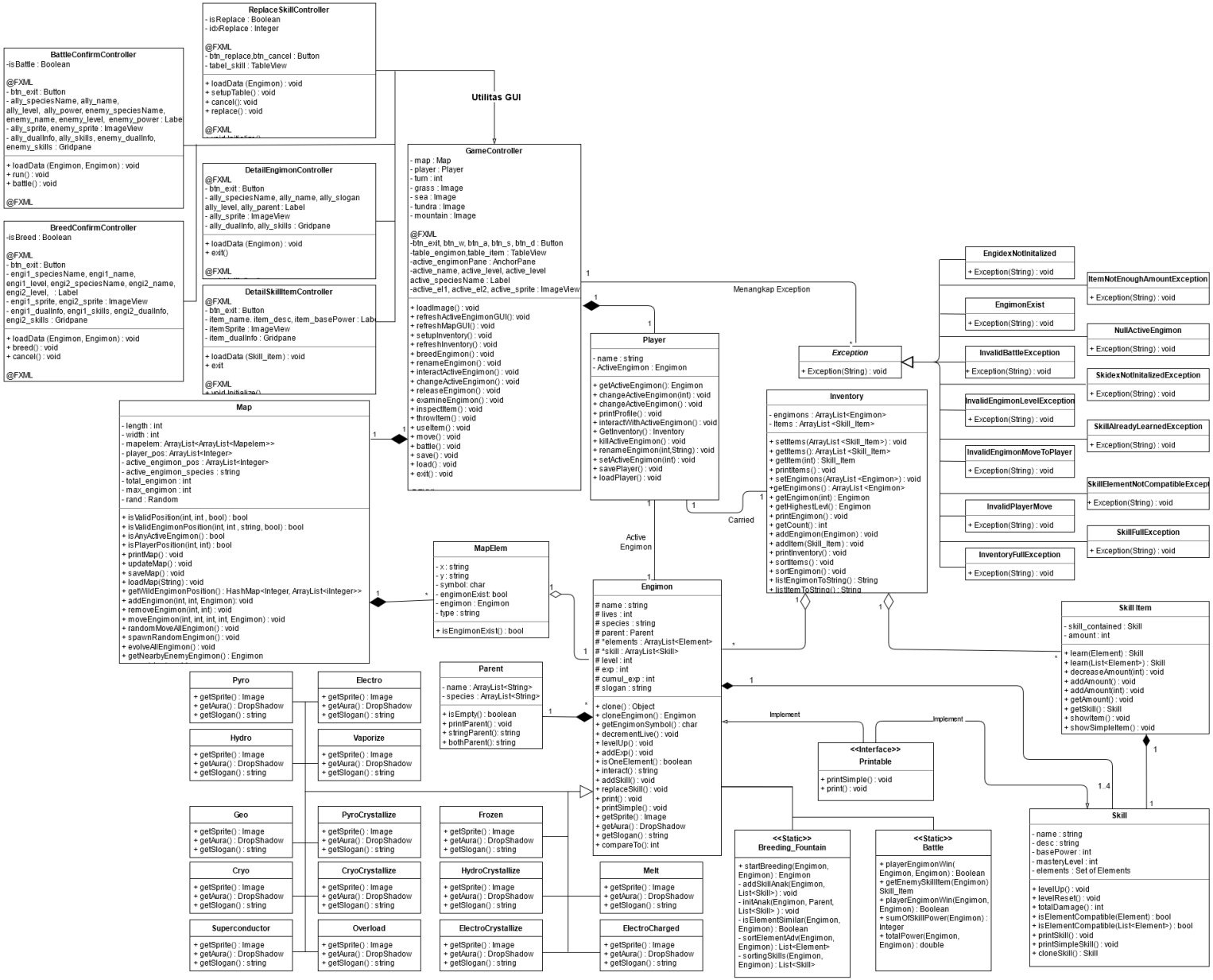
Kelas : 01
Nama Kelompok : Tutturuuu~!
1. 13519008 / Ronggur Mahendra Widya Putra
2. 13519014 / Mahameru Ds
3. 13519020 / Muhammad Azhar Faturahman
4. 13519026 / Gde Anantha Priharsena
5. 13519036 / Andrew
6. 13519043 / Reihan Andhika Putra
Asisten Pembimbing : Adyaksa Wisanggeni

1. Diagram Kelas

Pada desain Diagram Kelas yang kami buat, terdapat beberapa kelas dan asosiasi yang menghubungkan antar kelas bersangkutan. Kelas yang kami buat diantaranya Engimon, Parent, Skill, Skill_Item, Inventory, Player, Map, MapElem, GameState, ShortInput, dan WriteException. Desain ini kami pilih dengan alasan agar setiap entitas yang mungkin ada pada permainan Engimon dipisah menjadi kelasnya masing-masing. Setiap kelas memiliki tanggung jawabnya masing-masing dan terpisah dari tanggung jawab kelas lain. Hubungan antar kelas menyatakan keterhubungan seperti *association, aggregation, composition,* dan *inheritance*. Kelebihan dari desain yang kami buat adalah berhasil menciptakan konsep *isolation*, yang mana setiap kelas tidak perlu tau bagaimana kelas lain bekerja, hanya cukup tau layanan apa saja yang dapat diberikan oleh kelas tersebut. Sehingga dalam membuat kelas hanya perlu memikirkan layanan apa yang dapat diberikan oleh kelas tersebut dan menjadi bebas dalam menentukan struktur internal kelas. Dengan demikian, pembuatan kelas bisa lebih cepat, lebih mudah, dan lebih teratur. Selain itu pada proses *maintenance* dan *debugging* pun akan lebih mudah untuk mencari *bug* yang ada dalam program. Akan tetapi, kelemahan dari desain yang kami buat adalah setiap spesies Engimon perlu dibuat kelasnya masing-masing, akan tetapi hal ini tidak melanggar konsep DRY karena setiap spesies berbeda dan memiliki skill dan nama yang unik. Secara umum, pengerjaan dan pembuatan desain OOP berjalan lancar, akan tetapi pada awalnya kami mengalami kesulitan dalam menentukan hubungan antar kelas, dan terdapat kesalahan perspektif dalam menganalisis spesifikasi tugas besar, sehingga terjadi ketidakseimbangan dalam pembagian tugas, namun hal tersebut dapat kami atasi dengan komunikasi yang baik antar anggota kelompok.

Note : Untuk memperkecil dan menyederhanakan class diagram, kami tidak menuliskan empat sekawan, getter, dan setter.


ReplaceSkillController
- isReplace : Boolean
- idxReplace : Integer
@FXML
- btn_replace,btn_cancel : Button
- tabel_skill : TableView
+ loadData (Engimon) : void
+ setupTable() : void
+ cancel(): void
+ replace() : void
@FXML
BattleConfirmController
-isBattle : Boolean
@FXML
- btn_exit : Button
- ally_speciesName, ally_name, ally_level, ally_power, enemy_speciesName, enemy_name, enemy_level, enemy_power : Label
- ally_sprite, enemy_sprite : ImageView
- ally_dualInfo, ally_skills, enemy_dualInfo, enemy_skills : Gridpane
+ loadData (Engimon, Engimon) : void
+ run() : void
+ battle() : void
@FXML
Utilitas GUI
DetailEngimonController
@FXML
- btn_exit : Button
- ally_speciesName, ally_name, ally_slogan
ally_level, ally_parent : Label
- ally_sprite : ImageView
- ally_dualInfo, ally_skills : Gridpane
+ loadData (Engimon) : void
+ exit()
@FXML
BreedConfirmController
-isBreed : Boolean
@FXML
- btn_exit : Button
- engi1_speciesName, engi1_name, engi1_level, engi2_speciesName, engi2_name, engi2_level, : Label
- engi1_sprite, engi2_sprite : ImageView
- engi1_dualInfo, engi1_skills, engi2_dualInfo, engi2_skills : Gridpane
+ loadData (Engimon, Engimon) : void
+ breed() : void
+ cancel() : void
@FXML
DetailSkillItemController
@FXML
- btn_exit : Button
- item_name, item_desc, item_basePower : Label
- itemSprite : ImageView
- item_dualInfo : Gridpane
+ loadData (Skill_Item) : void
+ exit
@FXML
GameController
- map : Map
- player : Player
- turn : int
- grass : Image
- sea : Image
- tundra : Image
- mountain : Image
@FXML
-btn_exit, btn_w, btn_a, btn_s, btn_d : Button
-table_engimon,table_item : TableView
-active_engimonPane : AnchorPane
-active_name, active_level, active_level
active_speciesName : Label
-active_el1, active_el2, active_sprite : ImageView
+ loadImage() : void
+ refreshActiveEngimonGUI(): void
+ refreshMapGUI() : void
+ setupInventory() : void
+ refreshInventory() : void
+ breedEngimon() : void
+ renameEngimon() : void
+ interactActiveEngimon() : void
+ changeActiveEngimon() : void
+ releaseEngimon() : void
+ examineEngimon() : void
+ inspectItem() : void
+ throwItem() : void
+ useItem() : void
+ move() : void
+ battle() : void
+ save() : void
+ load() : void
+ exit() : void
Menangkap Exception
Player
- name : string
- ActiveEngimon : Engimon
+ getActiveEngimon(): Engimon
+ changeActiveEngimon(int) : void
+ changeActiveEngimon() : void
+ printProfile() : void
+ interactWithActiveEngimon() : void
+ GetInventory() : Inventory
+ killActiveEngimon() : void
+ renameEngimon(int,String) : void
+ setActiveEngimon(int) : void
+ savePlayer() : void
+ loadPlayer() : void
Exception
+ Exception(String) : void
EngidexNotInitalized
+ Exception(String) : void
ItemNotEnoughAmountException
+ Exception(String) : void
EngimonExist
+ Exception(String) : void
NullActiveEngimon
+ Exception(String) : void
InvalidBattleException
+ Exception(String) : void
SkidexNotInitalizedException
+ Exception(String) : void
InvalidEngimonLevelException
+ Exception(String) : void
SkillAlreadyLearnedException
+ Exception(String) : void
InvalidEngimonMoveToPlayer
+ Exception(String) : void
SkillElementNotCompatibleExcept
+ Exception(String) : void
InvalidPlayerMove
+ Exception(String) : void
SkillFullException
+ Exception(String) : void
InventoryFullException
+ Exception(String) : void
Inventory
- engimons : ArrayList<Engimon>
- Items : ArrayList <Skill_Item>
+ setItems(ArrayList <Skill_Item>) : void
+ getItems(): ArrayList <Skill_Item>
+ getItem(int) : Skill_Item
+ printItems() : void
+ setEngimons(ArrayList <Engimon>) : void
+getEngimons() : ArrayList <Engimon>
+ getEngimon(int): Engimon
+ getHighestLevl() : Engimon
+ printEngimon() : void
+ getCount() : int
+ addEngimon(Engimon) : void
+ addItem(Skill_Item) : void
+ printInventory() : void
+ sortItems() : void
+ sortEngimon() : void
+ listEngimonToString() : String
+ listItemToString() : String
Carried
Active Engimon
Map
- length : int
- width : int
- mapelem: ArrayList<ArrayList<Mapelem>>
- player_pos: ArrayList<Integer>
- active_engimon_pos : ArrayList<Integer>
- active_engimon_species : string
- total_engimon : int
- max_engimon : int
- rand : Random
+ isValidPosition(int, int , bool) : bool
+ isValidEngimonPosition(int, int , string, bool) : bool
+ isAnyActiveEngimon() : bool
+ isPlayerPosition(int, int) : bool
+ printMap() : void
+ updateMap() : void
+ saveMap() : void
+ loadMap(String) : void
+ getWildEngimonPosition() : HashMap<Integer, ArrayList<Integer>>
+ addEngimon(int, int, Engimon): void
+ removeEngimon(int, int) : void
+ moveEngimon(int, int, int, Engimon) : void
+ randomMoveAllEngimon() : void
+ spawnRandomEngimon() : void
+ evolveAllEngimon() : void
+ getNearbyEnemyEngimon() : Engimon
MapElem
- x : string
- y : string
- symbol: char
- engimonExist: bool
- engimon : Engimon
- type : string
+ isEngimonExist() : bool
Engimon
name : string
lives : int
species : string
parent : Parent
*elements : ArrayList<Element>
*skill : ArrayList<Skill>
level : int
exp : int
cumul_exp : int
slogan : string
+ clone() : Object
+ cloneEngimon() : Engimon
+ getEngimonSymbol() : char
+ decrementLive() : void
+ levelUp() : void
+ addExp() : void
+ isOneElement() : boolean
+ interact() : string
+ addSkill() : void
+ replaceSkill() : void
+ print() : void
+ printSimple() : void
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
+ compareTo() : int
Parent
- name : ArrayList<String>
- species : ArrayList<String>
+ isEmpty() : boolean
+ printParent() : void
+ stringParent(): string
+ bothParent(): string
Pyro
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Electro
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Hydro
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Vaporize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Geo
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
PyroCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Frozen
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Cryo
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
CryoCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
HydroCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Melt
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Superconductor
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Overload
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
ElectroCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
ElectroCharged
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Implement
<<Interface>>
Printable
+ printSimple() : void
+ print() : void
<<Static>>
Breeding_Fountain
+ startBreeding(Engimon, Engimon) : Engimon
- addSkillAnak(Engimon, List<Skill>) : void
- initAnak(Engimon, Parent, List<Skill>): void
- isElementSimilar(Engimon, Engimon) : Boolean
- sortElementAdv(Engimon, Engimon) : List<Element>
- sortingSkills(Engimon, Engimon) : List<Skill>
<<Static>>
Battle
+ playerEngimonWin(Engimon, Engimon) : Boolean
+ getEnemySkillItem(Engimon) Skill_Item
+ playerEngimonWin(Engimon, Engimon) : Boolean
+ sumOfSkillPower(Engimon) : Integer
+ totalPower(Engimon, Engimon) : double
Skill Item
- skill_contained : Skill
- amount : int
+ learn(Element) : Skill
+ learn(List<Element>) : Skill
+ decreaseAmount(int) : void
+ addAmount() : void
+ addAmount(int) : void
+ getAmount() : void
+ getSkill() : Skill
+ showItem() : void
+ showSimpleItem() : void
Skill
- name : string
- desc : string
- basePower : int
- masteryLevel : int
- elements : Set of Elements
+ levelUp() : void
+ levelReset() : void
+ totalDamage() : int
+ isElementCompatible(Element) : bool
+ isElementCompatible(List<Element>) : bool
+ printSkill() : void
+ printSimpleSkill() : void
+ cloneSkill() : Skill

2. Penerapan Konsep OOP

2.1. Polymorphism

Polymorphism adalah kemampuan sebuah object untuk memiliki banyak bentuk. Hal ini berarti sebuah class dapat diinstansiasikan dengan menggunakan *subclassnya*. Dengan penggunaan *polymorphism*, programmer dapat lebih mudah dalam melakukan manajemen generalisasi object tergantung dengan *subclass*-nya. *Polymorphism* juga meningkatkan *code reusability* dan *maintainability*.

Cuplikan kode yang menerapkan konsep *Polymorphism* :

Class / Method	Alasan

`Engimon e1 = new Cryo("Cryo sp.", true);`
`Engimon e2 = new CryoCrystallize("CyroCrystallize sp.", 1);`
`Engimon e3 = new Electro("Electro sp.", 1);`
`Engimon e4 = new ElectroCharged("ElectroCharged sp.", 1);`
`Engimon e5 = new ElectroCrystallize("ElectroCrystallize sp.", 1);`
`Engimon e6 = new Frozen("Frozen sp.", 1);`
`Engimon e7 = new Geo("Geo sp.", 1);`
`Engimon e8 = new Hydro("Hydro sp.", 1);`
`Engimon e9 = new HydroCrystallize("HydroCrystallize sp.", 1);`
`Engimon e10 = new Melt("Melt sp.", 1);`
`Engimon e11 = new Overload("Overload sp.", 1);`
`Engimon e12 = new Pyro("Pyro sp.", 1);`
`Engimon e13 = new PyroCrystallize("PyroCrystallize sp.", 1);`
`Engimon e14 = new Superconductor("Superconductor sp.", 1);`
`Engimon e15 = new Vaporize("Vaporize sp.", 1);`
	Berikut adalah penerapan konsep *polymorphism* dalam program. Instansiasi type objek e1 hingga e15 dilakukan pada kelas Engimon, namun dengan menggunakan konstruktor dari spesies Engimon. Hal ini sesuai dengan konsep *polymorphism*, karena setiap spesies dapat berperilaku sebagai engimon. Dengan demikian, dapat memudahkan program untuk melakukan operasi-operasi lain tanpa harus membuat kasus untuk setiap spesies Engimon.

2.2. Inheritance/Composition/Aggregation

Inheritance adalah salah satu cara untuk mencapai *abstraction. Inheritance* adalah cara untuk membuat suatu kelas yang mendapatkan atribut dan *method* dari kelas lain. Dengan menggunakan *inheritance*, programmer bisa menghindari penulisan ulang *attribute* dan *method* dari suatu kelas yang sebenarnya sudah dipunyai oleh kelas lain. Selain itu, menggunakan *inheritance* akan membuat programmer lebih mudah

dalam manajemen kelas-kelas yang ada dan meningkatkan *code reusability* apabila program yang dibangun semakin besar dan semakin kompleks.

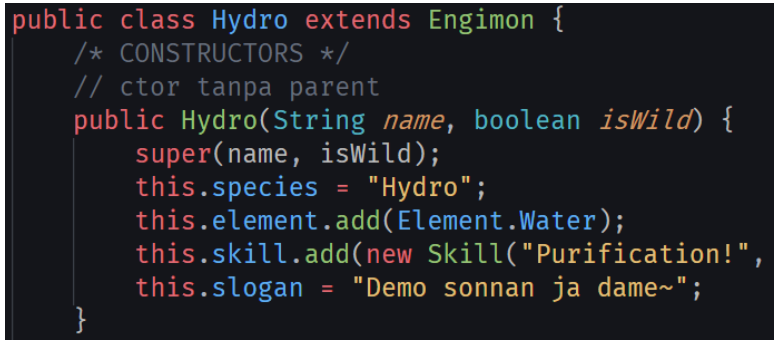
Composition adalah cara supaya suatu kelas dapat mempunyai *attribute* berupa kelas lain. C*omposition* membuat suatu kelas dapat mempunyai beberapa *behaviour* yang sama dengan kelas lain, namun tidak harus semuanya sama persis. *Composition* adalah relasi *has-a* dimana suatu objek yang rumit bisa terdiri dari beberapa objek lainnya yang lebih sederhana dan saling berhubungan namun objek yang lebih rumit tersebut tidak harus bisa melakukan semua tindakan yang bisa dilakukan oleh objek yang lebih kecil tersebut. Perlu diperhatikan, pada *composition,* objek yang lebih rumit tersebut tidak dapat bertahan tanpa adanya objek yang lebih sederhana. Dengan menggunakan *composition,* programmer dapat membuat kelas yang lebih kompleks dari beberapa kelas yang lebih kecil. Hal ini akan mengurangi kompleksitas program yang dibuat dan membuat *debugging* semakin mudah.

Aggregation adalah cara supaya suatu kelas dapat mempunyai *attribute* berupa kelas lain. *Aggregation* membuat suatu kelas dapat mempunyai beberapa *behaviour* yang sama dengan kelas lain, namun tidak harus semuanya sama persis. Konsep dari *aggregation* mirip dengan *composition,* bedanya adalah pada *composition* objek yang lebih rumit tidak bisa berdiri tanpa adanya objek yang lebih sederhana, pada *aggregation* objek yang lebih rumit bisa berdiri bisa berdiri tanpa adanya objek yang lebih sederhana. Dengan menggunakan *aggregation,* programmer dapat membuat kelas yang lebih kompleks dari beberapa kelas yang lebih kecil. Hal ini akan mengurangi kompleksitas program yang dibuat dan membuat *debugging* semakin mudah.

Cuplikan kode yang menerapkan konsep *Inheritance/Composition/Aggregation* :

Class / Method	Alasan

`public class Pyro extends Engimon {`
`  /* CONSTRUCTORS */`
`  // ctor tanpa parent`
`  public Pyro(String name, boolean isWild) {`
`  super(name, isWild);`
`  this.species = "Pyro";`
`  this.element.add(Element.Fire);`
`  this.skill.add(new Skill("Explosion!",`
`  this.slogan = "Se no!~";`
`  }`
	Pyro merupakan salah satu spesies Engimon. Penerapan spesies Pyro adalah *subclass* atau kelas anak dari *class* Engimon. Penerapan ini baik digunakan karena setiap spesies memiliki *skill* bawaan dan slogan yang unik. Dengan konsep *inheritance*, instansiasi setiap spesies akan lebih mudah dilakukan .

	Hydro merupakan salah satu spesies Engimon. Penerapan spesies Hydro adalah *subclass* atau kelas anak dari *class* Engimon. Penerapan ini baik digunakan karena setiap spesies memiliki *skill* bawaan dan slogan yang unik. Dengan konsep *inheritance*, instansiasi setiap spesies akan lebih mudah dilakukan.
	Pada kasus disamping, kita memerlukan sebuah kelas yang merepresentasikan sebuah *Skill* dan banyaknya *Skill* tersebut disimpan. Kelas tersebut tidak perlu mewarisi semua *atribut* dan *method* yang dimiliki kelas *Skill* karena kelas tersebut hanya menjadi kelas penengah untuk mempermudah penyimpanan suatu *Skill. Composition* menjadi solusi yang lebih tepat untuk permasalahan ini karena dengan *composition* kelas yang ingin dibuat tidak perlu mewarisi semua *atribut* dan *method* namun masih bisa berperilaku layaknya kelas yang dikomposisikan.

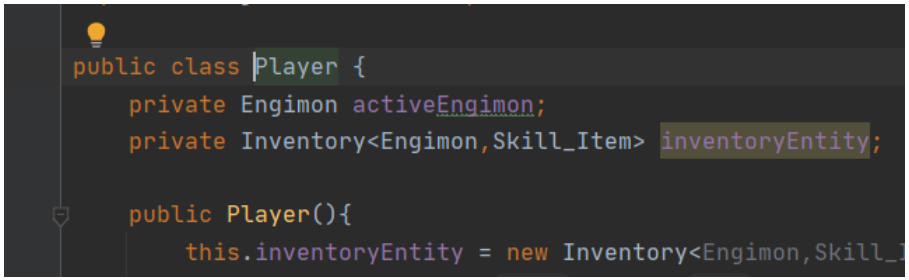
```java
public abstract class Engimon implements Cloneable, Compar
    /* FINAL ATTRIBUTES */
    private final int DEFAULT_LIVES = 3;
    private final int DEFAULT_WILD_LIVES = 1;

    /* FIELDS */
    protected String name;
    protected int lives;
    protected String species;
    protected Parent parent;
    protected List<Element> element = new ArrayList<>();
    protected List<Skill> skill = new ArrayList<>();
    protected int level;
    protected int exp;
    protected int cumul_exp;
    protected String slogan;
```

Setiap Engimon memiliki atribut yang mewakili Parentnya. Engimon dan Parent ini memiliki keterhubungan yang bersifat *aggregation* karena objek Parent bisa tetap ada walaupun Engimon tersebut sudah tidak ada. Hal ini sesuai dengan konsep jika Parent bisa dimiliki oleh beberapa Engimon yang berbeda.

```java
12  public class Inventory<E extends Engimon,I extends Skil
13      private List<I > ListItem;
14      private List<E> ListEngimon;
15      public Inventory(){
16          this.ListItem = new ArrayList<I>();
17          this.ListEngimon = new ArrayList<E>();
18      }
19
20      /** modify data**/
21      public void addEngimon(E Engimon)throws InventoryFul
22          if(this.getAmount() >100){
23              throw new InventoryFullException();
24          }
25          this.ListEngimon.add(Engimon);
```

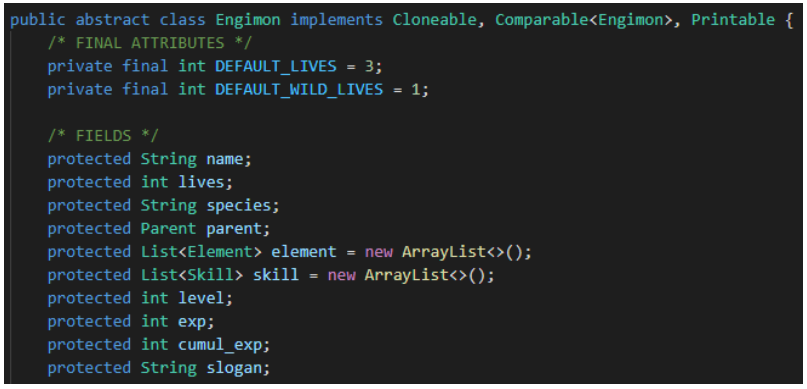
Class inventory men-composite extension dari class Engimon dan extension dari class Skill_item class Inventory menyimpan list Engimon dan list Skill_item yang terdapat dalam inventory.

	Class player men-composite kelas Engimon sebagai activeEngimon untuk merepresentasikan engimon mana yang tengah digunakan player. Class player juga mencomposite kelas Inventory untuk merepresentasikan inventory yang dimiliki oleh player.

2.3. Abstract Class

Abstract Class merupakan sebuah kelas yang dideklarasikan abstract. Jika sebuah kelas dideklarasikan sebagai abstract, maka kelas tersebut tidak dapat diinstansiasikan sebagai sebuah object. Abstract class dapat memiliki abstract method maupun tidak. Abstract method adalah sebuah method yang dideklarasikan abstract dan tidak memiliki implementasi. Jika sebuah method dideklarasikan abstract, maka method tersebut harus diimplementasikan oleh subclassnya. Jika sebuah class memiliki abstract method, maka kelas tersebut harus dideklarasikan sebagai abstract class.

Cuplikan kode yang menerapkan konsep *Abstract Class* :

Abstract Class	Alasan
	*Class* Engimon diimplementasikan sebagai *Abstract Class* untuk memudahkan generalisasi spesies pada Engimon. *Class* Engimon berisi atribut-atribut yang digunakan oleh setiap spesies. Kemudian, *class* Engimon akan digunakan sebagai *object polymorphism* setiap spesiesnya.

2.4. Interface

Interface adalah salah satu cara untuk mencapai *abstraction*. Sebuah object berkomunikasi dengan dunia luar melalui method-methodnya. Interface adalah sekumpulan method yang berhubungan namun tidak memiliki implementasi. Dalam bahasa pemrograman Java, sebuah kelas hanya dapat memiliki satu superclass. Bahasa Java tidak memperbolehkan “multiple inheritance”. Dengan penggunaan interface, maka sebuah kelas dapat mencapai “multiple inheritance”.

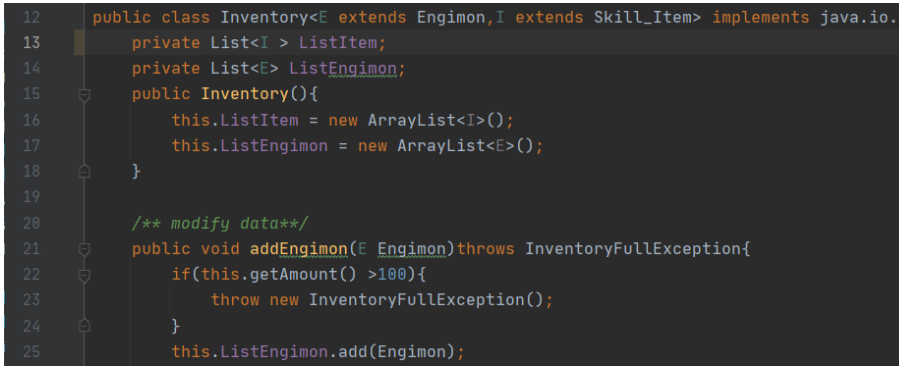
Cuplikan kode yang menerapkan konsep *Interface* :

Interface	Alasan
`public interface Printable { public void print(); public void printSimple(); }` `/** Printer **/ public void print() { System.out.println("-- " + name + " --"); System.out.println(desc); System.out.println("BasePower : " + basePower); System.out.println("MasteryLevel : " + masteryLevel); System.out.print("Elements : "); showSkillElements(); } public void printSimple() { System.out.println("Skill : " + name); System.out.println("BP/ML : " + basePower + "/" + masteryLevel); System.out.print("EL : "); showSkillElements(); }`	Interface ini digunakan untuk membedakan suatu kelas yang memiliki metode print seperti Engimon dan Skill.

2.5. Generic Type & Wildcards

Generic Type adalah suatu kemampuan dalam mendefinisikan suatu tipe data pada field class pada pemanggilan constructor kelas. Keuntungan menggunakan Generic diantaranya meningkatkan *expressive power*, meningkatkan *type safety*, dan mengekplisitkan parameter tipe dan mengimplisitkan *type casting*. Generic type pada suatu kelas dapat diberi boundary antara super(lower boundary) dan upper boundary(upper boundary), seperti pada cuplikan kode dibawah ini memanfaatkan upper boundary.

Cuplikan kode yang menerapkan konsep *Generic* :

Class / Method	Alasan
	Class Inventory diimplementasi sebagai class generic agar membuat Inventory bisa dengan mudah menyimpan banyak jenis class turunan dari class engimon dan skill_item.

```
12  public class Inventory<E extends Engimon,I extends Skill_Item> implements java.io.
13      private List<I > ListItem;
14      private List<E> ListEngimon;
15      public Inventory(){
16          this.ListItem = new ArrayList<I>();
17          this.ListEngimon = new ArrayList<E>();
18      }
19
20      /** modify data**/
21      public void addEngimon(E Engimon)throws InventoryFullException{
22          if(this.getAmount() >100){
23              throw new InventoryFullException();
24          }
25          this.ListEngimon.add(Engimon);
```

2.6. Exception Handling

Sebuah program tidak selalu berjalan sesuai dengan *flow* yang diharapkan. Terkadang dapat terjadi suatu *error* atau *unexpected behaviour* dalam sebuah program. Untuk menangani hal ini, kita menggunakan *exception*. *Exception* adalah kemampuan suatu object untuk merespons terhadap *error* atau *unexpected behaviour* yang terjadi dalam program. *Exception* akan mengubah arah berjalannya program sehingga setiap *error* dan *unexpected behaviour* dapat ditangani dengan baik tanpa merusak program atau memberhentikan program dengan paksa. Untuk menggunakan *exception*, dapat digunakan blok *try* and *catch*. Di dalam blok *try*, *caller method* akan mencoba menjalankan program sesuai dengan *flow* yang diharapkan. Jika terdapat suatu *error* atau *unexpected behaviour*, maka *caller method* akan melemparkan sebuah *exception* yang nantinya akan di-*handle* oleh *caller method*.

Berikut merupakan contoh penerapan *exception handling* pada tugas besar ini, antara lain :

Exception	Alasan

`package main.java.exception;`
 	
`public class InvalidEngimonMoveToPlayer extends Exception {`
`  static final long serialVersionUID = 1L; // Idk if this is nescessary`
 	
`  public InvalidEngimonMoveToPlayer(){`
`  super("Engimon liar tidak bisa bergerak ke posisi Player. Melakukan reposisi engimon liar");`
`  }`
`  public InvalidEngimonMoveToPlayer(String errorMessage){`
`  super(errorMessage);`
`  }`
`}`
 	
`// Move setiap engimon di map`
`for(Integer id : wildEngimon.keySet()){`
`  int number = rand.nextInt(5);`
`  int i = wildEngimon.get(id).get(0);`
`  int j = wildEngimon.get(id).get(1);`
`  if(number==1){`
`  // Engimon bergerak ke atas`
`  // Cek apakah tiles valid atau tiles yang dituju adalah tiles player`
`  if(isValidPosition(i-1, j, false)){`
`  this.moveEngimon(i, j, i-1, j, this.mapelem.get(i).get(j).get_engimon());`
`  } else if(this.isPlayerPosition(i-1, j)){`
`  throw new InvalidEngimonMoveToPlayer();`
`  }`
`  }else if(number==2){`
`  // Engimon bergerak ke kiri`
`  // Cek apakah tiles valid atau tiles yang dituju adalah tiles player`
`  if(isValidPosition(i, j-1, false)){`
`  this.moveEngimon(i, j, i, j-1, this.mapelem.get(i).get(j).get_engimon());`
`  }else if(this.isPlayerPosition(i, j-1)){`
`  throw new InvalidEngimonMoveToPlayer();`
`  }`
	Pada bagian ini, *exception* berguna untuk melakukan penanganan terhadap kasus *Unexpected Behaviour* ketika menggerakan *wild engimon* dalam peta. *Unexpected Behaviour* yang dimaksud pada *method* ini adalah ketika *wild engimon* ingin bergerak ke suatu posisi yang juga merupakan posisi dari player. Kemudian Exception ini akan ditangkap di main program saat berjalan agar dapat ditangani lebih lanjut, pada kasus ini hanya menampilkan pesan bahwa Engimon liar tidak bisa bergerak ke posisi player dan melakukan reposisi engimon liar ke posisi semula.

<table>
<tr>
<td>

```java
package main.java.exception;

public class InventoryFullException extends Exception{
    public InventoryFullException(){
        super("Inventory Penuh");
    }
}
```

```java
/** modify data**/
public void addEngimon(E Engimon)throws InventoryFullException{
    if(this.getAmount() >100){
        throw new InventoryFullException();
    }
    this.ListEngimon.add(Engimon);
    this.sortEngimons();
}
```

</td>
<td>Pada bagian ini, exception berguna untuk melakukan penanganan terhadap kasus Unexpected Behaviour ketika menambahkan engimon atau skill item ke inventory/ Unexpected Behaviour yang dimaksud pada method ini adalah ketika player ingin menambahkan engimon atau skill item saat inventory sudah memiliki 100 item (engimon dan skill item). Kemudian Exception ini akan ditangkap di main program saat berjalan agar dapat ditangani lebih lanjut, pada kasus ini hanya menampilkan pesan bahwa inventory penuh.</td>
</tr>
<tr>
<td>

```java
package main.java.exception;

public class SkillElementNotCompatibleException extends Exception {
    public SkillElementNotCompatibleException(){
        super("Element Skill tidak cocok dengan element Engimon");
    }
}
```

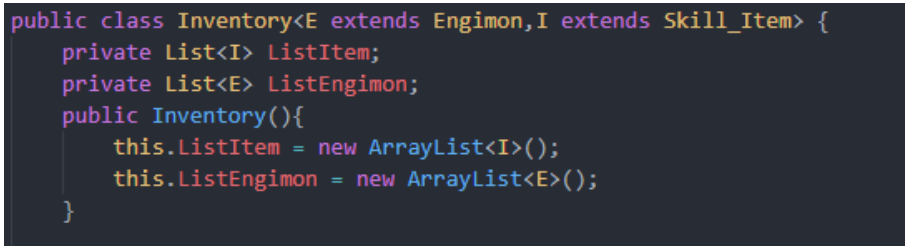
</td>
<td>Pada bagian ini, exception berguna untuk melakukan penanganan terhadap kasus Unexpected Behaviour ketika engimon yang dimiliki player menggunakan skill item untuk mempelajari skill terkait. Unexpected Behaviour yang dimaksud pada method ini adalah ketika skill yang ingin dipelajari tidak cocok dengan element yang dimiliki oleh engimon tersebut. Kemudian Exception ini akan ditangkap di main program saat berjalan agar dapat ditangani lebih lanjut, pada kasus ini hanya menampilkan pesan bahwa elemen dari skill tersebut tidak cocok dengan elemen engimon..</td>
</tr>
</table>

	
dan lain-lain..	

2.7. Java Collection Framework

Java Collection Framework adalah arsitektur yang merepresentasikan dan memanipulasikan Collection. Java Collection Framework terdiri atas Collection Interfaces, Collection Implementations, dan Collection Algorithms. Collection Interface adalah struktur data abstrak yang merepresentasikan Collection. Collection Implementation adalah implementasi konkrit terhadap Collection Interface. Collection Algorithms berisi algoritma-algoritma yang sangat berguna untuk object-object yang mengimplementasikan Collection Interface. Java Collection Framework digunakan agar kita tidak perlu membuat struktur data yang sama dan meningkatkan performance program.

Berikut merupakan contoh penerapan Java Collection Framework pada tugas besar ini, antara lain :

Java Collection Framework	Alasan
	**JCF** : List (Collection Interfaces) dan ArrayList (Collection Implementations) JCF List digunakan untuk mengimplementasikan sebuah “List” yang jumlah elemennya dinamis. Karena isi dan ukuran dari Inventory dapat berubah ubah sesuai dengan berjalannya permainan. Selain itu digunakan juga ArrayList untuk merepresentasikan koordinat elemen-elemen peta, seperti posisi engimon liar, posisi player, dan posisi engimon aktif.

<table>
<tr>
<td>

```
public class Map {
    private int length; // Map Length
    private int width; // Map Width
    private ArrayList<ArrayList<Mapelem>> mapelem ; // Array of MapElem
    private ArrayList<Integer> player_pos; // Array of Integer with size
    private ArrayList<Integer> active_engimon_pos; // Array of Integer wi
    private String active_engimon_species;  // Current Active Engimon Spe
    private int total_engimon; // Total Wild Engimon that exist on Map
    private final int max_engimon = 20; // Maximum Engimon Exist on Map
```

</td>
<td></td>
</tr>
<tr>
<td>

```
public class Engidex {
    private static Map<String, Engimon> Engidex  = new HashMap<String, Engimon>();
```

```
private static List<Element> sortElementAdv(Engimon parentA, Engimon parentB) {
    List<Element> elParentA = parentA.getElement();
    List<Element> elParentB = parentB.getElement();
    Map<Element,Double> elChild = new HashMap<>();

    double elAdv;
    // Hitung Element Advantage parent A
    for (Element elA : elParentA) {
        elAdv = Element.advantage(elA, elParentB);
        elChild.put(elA, elAdv);
    }

    // Hitung Element Advantage parent B
    for (Element elB : elParentB) {
        elAdv = Element.advantage(elB, elParentA);
        if (elChild.containsKey(elB) && elAdv > elChild.get(elB)) {
            elChild.replace(elB, elAdv);
        } else {
            elChild.put(elB, elAdv);
        }
    }
```

</td>
<td>

JCF : Map (Collection Interfaces) dan HashMap (Collection Implementations)

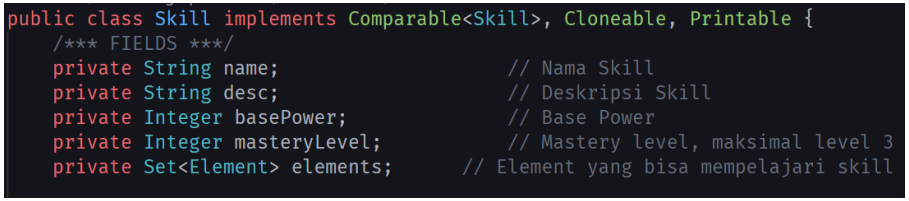
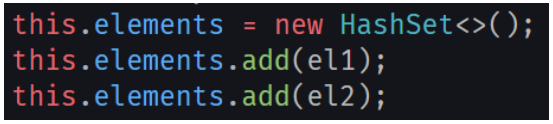
JCF untuk struktur data map dibutuhkan untuk mengimplementasikan sebuah struktur data yang memiliki *value* yang unik berdasarkan *key* yang merujuknya. Hal ini dibutuhkan untuk beberapa kasus seperti untuk menyimpan database engimon (Engidex) dan untuk mengetahui posisi dari engimon yang ada di Peta. Untuk kasus database engimon (Engidex), engimon yang dirujuk akan unik untuk setiap *key* yang direpresentasikan oleh spesiesnya. Sementara, untuk mengetahui posisi dari engimon yang ada di peta, posisi engimon yang dirujuk akan unik untuk setiap yang direpresentasikan oleh spesiesnya.

</td>
</tr>
<tr>
<td></td>
<td>

JCF : Set (Collection Interfaces) dan HashSet (Collection Implementations)

JCF untuk struktur data set digunakan untuk merepresentasikan

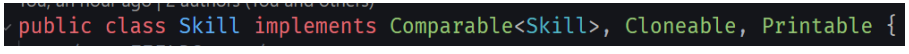
</td>
</tr>
</table>

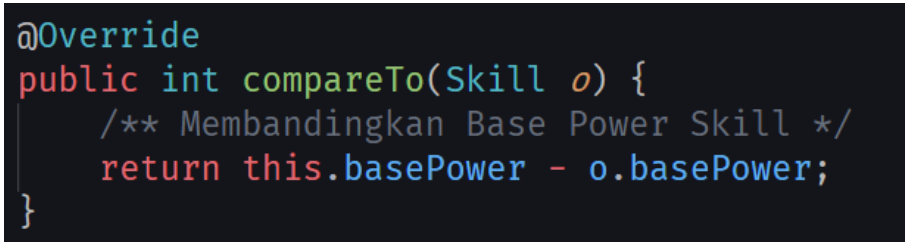
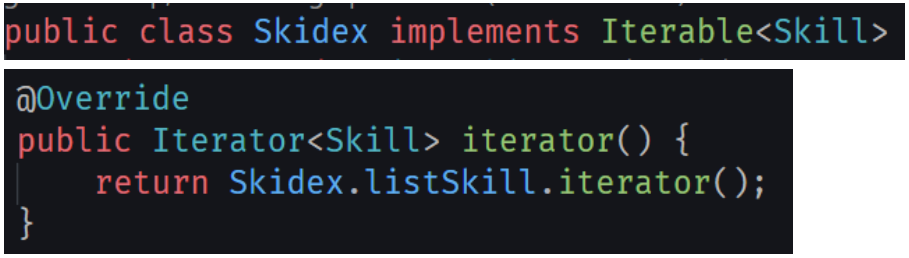
 	suatu data yang diharapkan tidak memiliki replikasi, sehingga ini sangat cocok digunakan untuk merepresentasikan elemen yang bisa mempelajari suatu skill.
dan lain-lain..	

2.8. Java API

Java API terdiri dari semua class yang merupakan bagian dari Java Development Kit (JDK), termasuk semua package, class, dan interface, bersama dengan attribute, dan method masing-masing. Sejak Java 8, terdapat sebuah API baru yaitu Stream API. Java Stream API adalah sekumpulan proses yang dilakukan terhadap suatu *collection of objects*.

Cuplikan kode yang menerapkan konsep Java API :

Java API	Alasan
Java Stream API 	Java Stream API memungkinkan untuk melakukan iterasi secara internal dalam suatu Collection pada Java. Hal ini sangat berguna sekali untuk mempersingkat suatu iterasi yang panjang menjadi hanya beberapa baris saja
Comparable 	Suatu skill dapat dibandingkan agar pada saat pengurutan skill berdasarkan basePower nya dapat dilakukan dengan langsung, sehingga dapat mengimplementasikan interface Comparable yang

	sudah disediakan oleh Java
Clonable 	Engimon merupakan suatu class abstract, sehingga untuk menyalin secara deep copy, diperlukan implementasi interface Clonable. Hal ini sangat penting karena semua spesies Engimon terdaftar pada suatu Engidex dan untuk membuat objek Engimon yang serupa diperlukan deep copy dari Engidex ke objek Engimon baru.
Iterable 	Iterable digunakan agar dalam menggunakan Skidex dapat langsung melakukan iterasi secara foreach.

3. Bonus Yang dikerjakan

3.1. Bonus Kreasi Mandiri

3.1.1. Engidex (Database Engimon) + 15 Jenis Engimon

Engidex adalah sebuah ensiklopedia Engimon. Ensiklopedia ini akan memuat Engimon dan informasi terkait Engimon tersebut. Engidex diterapkan sebagai STL Map dengan key berupa string spesies dan value berupa pointer to Engimon. Selain itu, kelompok kami juga membuat kombinasi *dual-element* untuk semua elemen yang ada. Beberapa engimon dengan elemen yang berlawanan tidak bisa didapatkan melalui *breeding,* namun harus ditangkap dengan mengalahkan engimonnya di peta secara langsung.

```
public class Engidex {
    private static Map<String, Engimon> Engidex  = new HashMap<String, Engimon>();
    /**
     * Initializing the first instance of Engidex. Unless initialized, the engidex
     */
    public static void initEngidex(){
        Engimon e1 = new Cryo("Cryo sp.", true);
        Engimon e2 = new CryoCrystallize("CyroCrystallize sp.", 1);
        Engimon e3 = new Electro("Electro sp.", 1);
        Engimon e4 = new ElectroCharged("ElectroCharged sp.", 1);
        Engimon e5 = new ElectroCrystallize("ElectroCrystallize sp.", 1);
        Engimon e6 = new Frozen("Frozen sp.", 1);
        Engimon e7 = new Geo("Geo sp.", 1);
        Engimon e8 = new Hydro("Hydro sp.", 1);
        Engimon e9 = new HydroCrystallize("HydroCrystallize sp.", 1);
        Engimon e10 = new Melt("Melt sp.", 1);
        Engimon e11 = new Overload("Overload sp.", 1);
        Engimon e12 = new Pyro("Pyro sp.", 1);
        Engimon e13 = new PyroCrystallize("PyroCrystallize sp.", 1);
        Engimon e14 = new Superconductor("Superconductor sp.", 1);
        Engimon e15 = new Vaporize("Vaporize sp.", 1);
```

3.1.2. Skidex (Database Skill) + Skill MultiElement

Database skill adalah kumpulan skill yang disimpan dalam vector (list) yang berperan sebagai rujukan saat suatu skill akan dibuat, sehingga tidak perlu secara manual membuat skill, jadi skill dapat diambil secara langsung dari *database* yang sudah ada. Skill yang dibuat dapat terdiri atas banyak elemen.

```
public static void initSkill() {
    listSkill.clear();

    List<Element> allElements = new ArrayList<>();      You, 2 weeks ago • add Skidex
    /* All Elements Skill */
    listSkill.add(new Skill("Punch", "Pukulan Maut!",10,allElements));
    listSkill.add(new Skill("Kick", "Sikat Miring!",10,allElements));
    listSkill.add(new Skill("ONE-PUNCH", "Hiyaa jurus si botak", 20, allElements));

    /* 1 Element Skill */
    listSkill.add(new Skill("Fire Breath", "Hah Naga!", 20, Element.Fire));
    listSkill.add(new Skill("Zoroaster", "Api keabadian", 30, Element.Fire));
    listSkill.add(new Skill("Gush", "Ciuhh!", 20, Element.Water));
    listSkill.add(new Skill("Ravage", "Tentakel + Anime", 30, Element.Water));
    listSkill.add(new Skill("Ball Lighting", "Lookin for me?", 20, Element.Electric));
    listSkill.add(new Skill("Static Storm", "I'm ecstatic!",30 ,Element.Electric));
    listSkill.add(new Skill("Fissure", "Feel the earth shake!", 20, Element.Ground));
    listSkill.add(new Skill("Echo Slam", "1 Million Dollar Slam", 30, Element.Ground));
    listSkill.add(new Skill("Ice Path", "Beku broh", 20, Element.Ice));
    listSkill.add(new Skill("Freezing Field", "Badai salju", 30,Element.Ice));

    /* 2 Element Skill */
    listSkill.add(new Skill("Rupture", "Bocor Bocor", 30, Element.Fire, Element.Water));
    listSkill.add(new Skill("Flamming Lasso", "Kena lo", 30, Element.Fire, Element.Electric));
    listSkill.add(new Skill("Life Break", "Loncat Indah", 30, Element.Fire, Element.Ground));
    listSkill.add(new Skill("Hand of God", "Tangan Madonna", 30, Element.Fire, Element.Ice));
    listSkill.add(new Skill("Chronosphere", "Zawarudo", 30, Element.Water, Element.Electric));
    listSkill.add(new Skill("Feast", "Laperrr",30,Element.Water,Element.Ground));
    listSkill.add(new Skill("Flux", "Ababil", 30, Element.Water, Element.Ice));
    listSkill.add(new Skill("Mana Void", "Duarr", 30, Element.Electric, Element.Ground));
```

3.1.3. Dual Element Breeding

Untuk kasus pertama, yaitu elemen kedua *parent* sama, untuk Engimon Parent dengan *Dual Element* akan memasuki kasus ini jika ada salah satu elemen *parent* yang sama (*similiar*) dengan elemen dari *paren*t lainnya. Kemudian penentuan spesies anak dilakukan secara random antara mengikuti spesies parent A atau spesies parent B.

```
if (isElementSimilar(parentA, parentB)) {
/* Jika elemen kedua parent sama, anak akan memiliki elemen yang sama dengan kedua par
    Spesies anak dipilih dari parent A atau parent B secara bebas (boleh random atau a
    spesifik tertentu). */
    int choice = rand.nextInt(2);
    if (choice == 0) {
        Engimon anak = Engidex.getEngimonBySpecies(parentA.getSpecies()).cloneEngimon(
        initAnak(anak, ortu, calonSkill);
        return anak;
    } else {
        Engimon anak = Engidex.getEngimonBySpecies(parentB.getSpecies()).cloneEngimon(
        initAnak(anak, ortu, calonSkill);
        return anak;
    }
```

Untuk kasus kedua, yaitu elemen kedua *parent* berbeda, akan dihitung terlebih dahulu total *Element Advantage* kedua parent mirip dengan menghitung *Element Advantage* pada Battle. Spesies anak akan mengikuti *parent* yang memiliki *Element Advantage* paling besar diantara keduanya.

```
} else {
/* Jika elemen kedua parent berbeda maka anak akan memiliki elemen dan spesies dari elemen
    yang memiliki element advantage yang lebih tinggi. */
    double elAdvA = Element.advantage(parentA.getElement(), parentB.getElement());
    double elAdvB = Element.advantage(parentB.getElement(), parentA.getElement());
    if (elAdvA > elAdvB) {
        Engimon anak = Engidex.getEngimonBySpecies(parentA.getSpecies()).cloneEngimon();
        initAnak(anak, ortu, calonSkill);
        return anak;
    } else if (elAdvA < elAdvB) {
        Engimon anak = Engidex.getEngimonBySpecies(parentB.getSpecies()).cloneEngimon();
        initAnak(anak, ortu, calonSkill);
        return anak;
    } else {
    /* eleAdv_a == eleAdv_b */
    /* Jika elemen kedua parent berbeda dan kedua elemen memiliki element advantage yang sama,
        maka anak akan memiliki spesies berbeda dari kedua parent yang memiliki kedua elemen pare
        (boleh dipilih random atau hardcoded). */
        List<Element> listEl = sortElementAdv(parentA, parentB);

        Engimon anak = Engidex.getEngimonByElement(listEl.get(0), listEl.get(1)).cloneEngimon();
        initAnak(anak, ortu, calonSkill);
        return anak;

    }
```

Untuk kasus ketiga, yaitu ketika kedua *parent* memiliki elemen yang berbeda dan *Element Advantage* kedua *parent* sama, maka akan dilakukan pendaftaran (*listing*) semua elemen yang dimiliki oleh kedua *parent* menjadi sebuah list. Kemudian list tersebut akan diurutkan berdasarkan *Element Advantage* nya dengan cara mirip seperti perhitungan nilai *Element Advantage* pada Battle Engimon. Setiap elemen pada Engimon *Parent* A akan “diadukan” dengan setiap elemen pada Engimon *Parent* B dan diambil nilai *Element Advantage* paling besar. List akan terurut berdasarkan nilai *Element Advantage* ini. Kemudian akan dipilih spesies anak yang memiliki elemen berupa 2 elemen dengan *Element Advantage* terbesar pada list tersebut. Pemilihan spesies anak ini dilakukan secara random dengan mengambil spesies dari *database* Engimon.

3.1.4. Purely Random Wild Engimon Generation

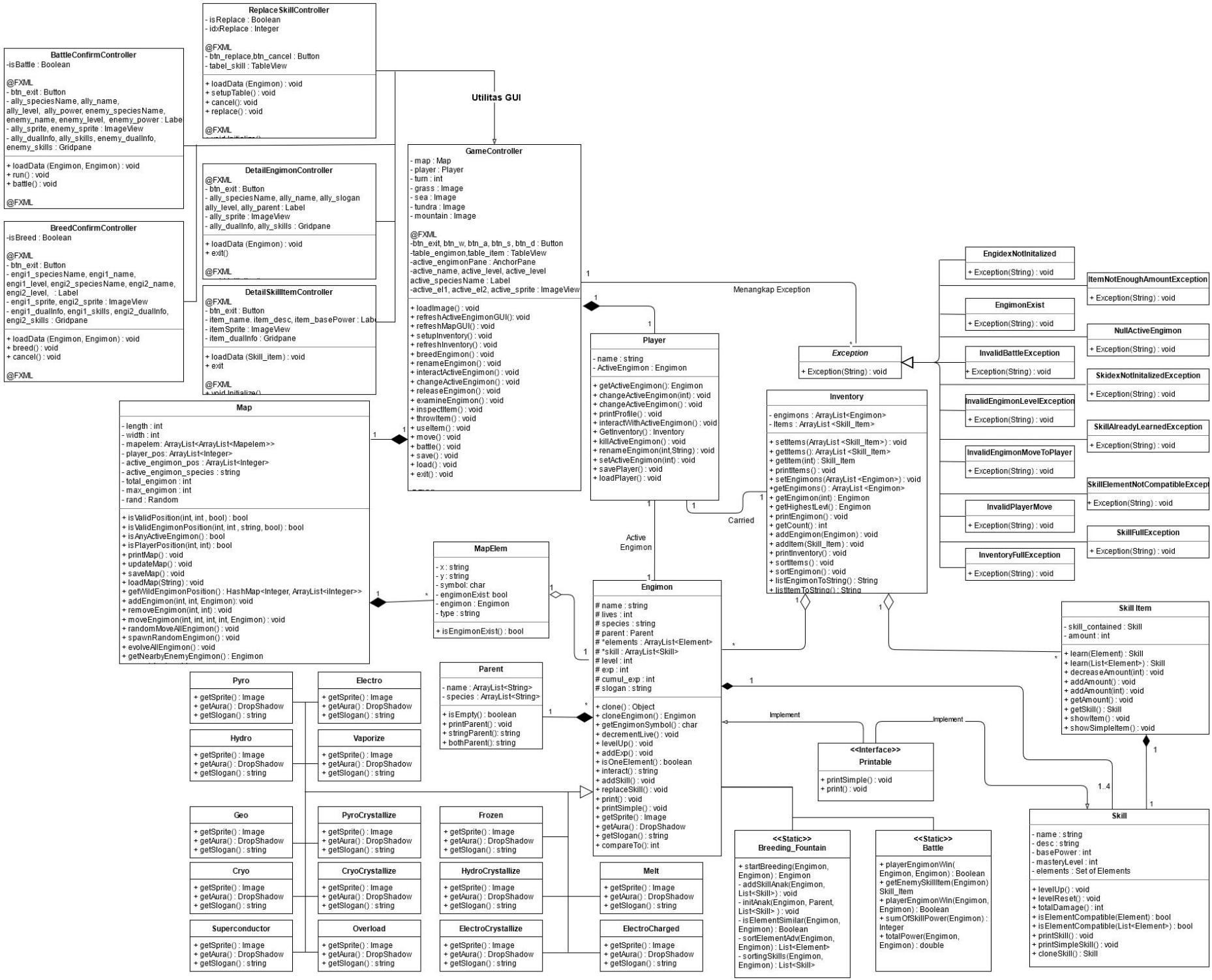
Wild Engimon Generation memanfaatkan batasan dari file map.txt yang sudah disediakan untuk memaksimalkan efektivitas generasi Engimon liar. Fungsi memiliki konstanta berupa batasan-batasan area *sea* untuk membatasi penambahan engimon dengan aturan : Engimon berhabitat *sea* hanya akan ditambahkan pada habitat *sea*, begitu pula untuk engimon berhabitat *grassland*. Fungsi ini juga memiliki vector string berisi nama-nama spesies engimon. *Container* ini berfungsi sebagai tabel frekuensi untuk meningkatkan kemungkinan kemunculan *Wild Engimon* yang hanya memiliki satu tipe dari *Wild Engimon* yang bertipe campuran dengan perbandingan 3:1. Engimon akan ditambahkan kedalam map secara acak dengan memanfaatkan fungsi rand dari C standard library, dan apabila terjadi *conflict* atau *exception*, maka akan dicoba untuk menambahkan posisi acak lain hingga tidak lagi terjadi *exception*.

4. External Library

Kelompok kami menggunakan library JavaFX dalam membuat GUI. Kami menggunakan JavaFX karena JavaFX merupakan library GUI yang lebih modern daripada JavaSwing dan JavaAtt sehingga komponen GUI yang dihasilkan lebih elegan daripada kedua library tersebut. Selain itu, JavaFX mempunyai SceneBuilder yang dapat mempermudah dalam mendesain GUI.

5. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
I.1. Engimon	13519014, 13519036	13519014, 13519036
I.2. Skill	13519020	13519020
I.3. Player	13519008, 13519020	13519008, 13519020
I.4. Battle	13519020, 13519043	13519020, 13519043
I.5. Breeding	13519020, 13519043	13519020, 13519043
I.6. Peta	13519026	13519026
II.1. Graphical User Interface	13519043	13519043
II.2. Save and Load Functionality	13519008, 13519026	13519008, 13519026


ReplaceSkillController
- isReplace : Boolean
- idxReplace : Integer
@FXML
- btn_replace,btn_cancel : Button
- tabel_skill : TableView
+ loadData (Engimon) : void
+ setupTable() : void
+ cancel(): void
+ replace() : void
@FXML
BattleConfirmController
-isBattle : Boolean
@FXML
- btn_exit : Button
- ally_speciesName, ally_name,
ally_level, ally_power, enemy_speciesName,
enemy_name, enemy_level, enemy_power : Label
- ally_sprite, enemy_sprite : ImageView
- ally_dualInfo, ally_skills, enemy_dualInfo,
enemy_skills : Gridpane
+ loadData (Engimon, Engimon) : void
+ run() : void
+ battle() : void
@FXML
Utilitas GUI
DetailEngimonController
@FXML
- btn_exit : Button
- ally_speciesName, ally_name, ally_slogan
ally_level, ally_parent : Label
- ally_sprite : ImageView
- ally_dualInfo, ally_skills : Gridpane
+ loadData (Engimon) : void
+ exit()
@FXML
BreedConfirmController
-isBreed : Boolean
@FXML
- btn_exit : Button
- engi1_speciesName, engi1_name,
engi1_level, engi2_speciesName, engi2_name,
engi2_level, : Label
- engi1_sprite, engi2_sprite : ImageView
- engi1_dualInfo, engi1_skills, engi2_dualInfo,
engi2_skills : Gridpane
+ loadData (Engimon, Engimon) : void
+ breed() : void
+ cancel() : void
@FXML
DetailSkillItemController
@FXML
- btn_exit : Button
- item_name. item_desc, item_basePower : Label
- itemSprite : ImageView
- item_dualInfo : Gridpane
+ loadData (Skill_item) : void
+ exit
@FXML
GameController
- map : Map
- player : Player
- turn : int
- grass : Image
- sea : Image
- tundra : Image
- mountain : Image
@FXML
-btn_exit, btn_w, btn_a, btn_s, btn_d : Button
-table_engimon,table_item : TableView
-active_engimonPane : AnchorPane
-active_name, active_level, active_level
active_speciesName : Label
-active_el1, active_el2, active_sprite : ImageView
+ loadImage() : void
+ refreshActiveEngimonGUI(): void
+ refreshMapGUI() : void
+ setupInventory() : void
+ refreshInventory() : void
+ breedEngimon() : void
+ renameEngimon() : void
+ interactActiveEngimon() : void
+ changeActiveEngimon() : void
+ releaseEngimon() : void
+ examineEngimon() : void
+ inspectItem() : void
+ throwItem() : void
+ useItem() : void
+ move() : void
+ battle() : void
+ save() : void
+ load() : void
+ exit() : void
Menangkap Exception
Player
- name : string
- ActiveEngimon : Engimon
+ getActiveEngimon(): Engimon
+ changeActiveEngimon(int) : void
+ changeActiveEngimon() : void
+ printProfile() : void
+ interactWithActiveEngimon() : void
+ GetInventory() : Inventory
+ killActiveEngimon() : void
+ renameEngimon(int,String) : void
+ setActiveEngimon(int) : void
+ savePlayer() : void
+ loadPlayer() : void
Exception
+ Exception(String) : void
EngidexNotInitalized
+ Exception(String) : void
ItemNotEnoughAmountException
+ Exception(String) : void
EngimonExist
+ Exception(String) : void
NullActiveEngimon
+ Exception(String) : void
InvalidBattleException
+ Exception(String) : void
SkidexNotInitalizedException
+ Exception(String) : void
InvalidEngimonLevelException
+ Exception(String) : void
SkillAlreadyLearnedException
+ Exception(String) : void
InvalidEngimonMoveToPlayer
+ Exception(String) : void
SkillElementNotCompatibleExcept
+ Exception(String) : void
InvalidPlayerMove
+ Exception(String) : void
SkillFullException
+ Exception(String) : void
InventoryFullException
+ Exception(String) : void
Inventory
- engimons : ArrayList<Engimon>
- Items : ArrayList <Skill_Item>
+ setItems(ArrayList <Skill_Item>) : void
+ getItems(): ArrayList <Skill_Item>
+ getItem(int) : Skill_Item
+ printItems() : void
+ setEngimons(ArrayList <Engimon>) : void
+getEngimons() : ArrayList <Engimon>
+ getEngimon(int) : Engimon
+ getHighestLevl() : Engimon
+ printEngimon() : void
+ getCount() : int
+ addEngimon(Engimon) : void
+ addItem(Skill_Item) : void
+ printInventory() : void
+ sortItems() : void
+ sortEngimon() : void
+ listEngimonToString() : String
+ listItemToString() : String
Carried
Active
Engimon
Map
- length : int
- width : int
- mapelem: ArrayList<ArrayList<Mapelem>>
- player_pos: ArrayList<Integer>
- active_engimon_pos : ArrayList<Integer>
- active_engimon_species : string
- total_engimon : int
- max_engimon : int
- rand : Random
+ isValidPosition(int, int , bool) : bool
+ isValidEngimonPosition(int, int , string, bool) : bool
+ isAnyActiveEngimon() : bool
+ isPlayerPosition(int, int) : bool
+ printMap() : void
+ updateMap() : void
+ saveMap() : void
+ loadMap(String) : void
+ getWildEngimonPosition() : HashMap<Integer, ArrayList<iInteger>>
+ addEngimon(int, int, Engimon): void
+ removeEngimon(int, int) : void
+ moveEngimon(int, int, int, int, Engimon) : void
+ randomMoveAllEngimon() : void
+ spawnRandomEngimon() : void
+ evolveAllEngimon() : void
+ getNearbyEnemyEngimon() : Engimon
MapElem
- x : string
- y : string
- symbol: char
- engimonExist: bool
- engimon : Engimon
- type : string
+ isEngimonExist() : bool
Engimon
name : string
lives : int
species : string
parent : Parent
*elements : ArrayList<Element>
*skill : ArrayList<Skill>
level : int
exp : int
cumul_exp : int
slogan : string
+ clone() : Object
+ cloneEngimon() : Engimon
+ getEngimonSymbol() : char
+ decrementLive() : void
+ levelUp() : void
+ addExp() : void
+ isOneElement() : boolean
+ interact() : string
+ addSkill() : void
+ replaceSkill() : void
+ print() : void
+ printSimple() : void
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
+ compareTo(): int
Parent
- name : ArrayList<String>
- species : ArrayList<String>
+ isEmpty() : boolean
+ printParent() : void
+ stringParent(): string
+ bothParent(): string
Skill Item
- skill_contained : Skill
- amount : int
+ learn(Element) : Skill
+ learn(List<Element>) : Skill
+ decreaseAmount(int) : void
+ addAmount() : void
+ addAmount(int) : void
+ getAmount() : void
+ getSkill() : Skill
+ showItem() : void
+ showSimpleItem() : void
Implement
<<Interface>>
Printable
+ printSimple() : void
+ print() : void
Implement
Skill
- name : string
- desc : string
- basePower : int
- masteryLevel : int
- elements : Set of Elements
+ levelUp() : void
+ levelReset() : void
+ totalDamage() : int
+ isElementCompatible(Element) : bool
+ isElementCompatible(List<Element>) : bool
+ printSkill() : void
+ printSimpleSkill() : void
+ cloneSkill() : Skill
<<Static>>
Breeding_Fountain
+ startBreeding(Engimon, Engimon) : Engimon
- addSkillAnak(Engimon, List<Skill>) : void
- initAnak(Engimon, Parent, List<Skill>) : void
- isElementSimilar(Engimon, Engimon) : Boolean
- sortElementAdv(Engimon, Engimon) : List<Element>
- sortingSkills(Engimon, Engimon) : List<Skill>
<<Static>>
Battle
+ playerEngimonWin(Engimon, Engimon) : Boolean
+ getEnemySkillItem(Engimon) Skill_Item
+ playerEngimonWin(Engimon, Engimon) : Boolean
+ sumOfSkillPower(Engimon) : Integer
+ totalPower(Engimon, Engimon) : double
Pyro
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Electro
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Hydro
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Vaporize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Geo
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
PyroCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Frozen
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Cryo
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
CryoCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
HydroCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Melt
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Superconductor
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
Overload
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
ElectroCrystallize
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string
ElectroCharged
+ getSprite() : Image
+ getAura() : DropShadow
+ getSlogan() : string