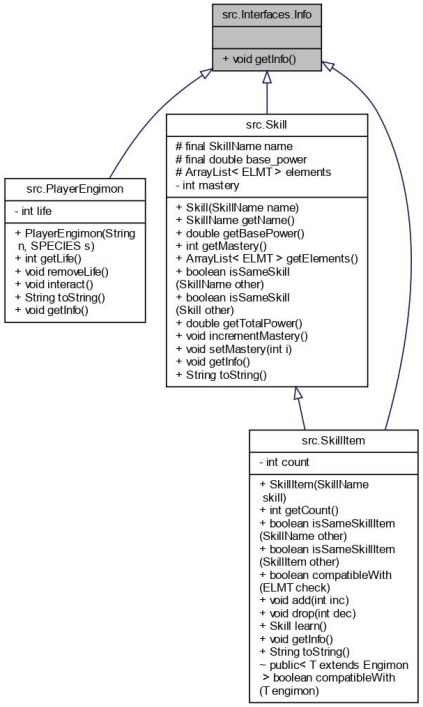
Kelas : 03
Nama Kelompok : yukkelarin
1. 13519111 / Febriawan Ghally Ar Rahman
2. 13519123 / Muhammad Rifky Muthahhari
3. 13519126 / Alvin Rizqi Alfisyahrin
4. 13519148 / Muhammad Atthaumar Rifqy
5. 13519154 / Rafi Raihansyah Munandar
6. 13519161 / Harith Fakhiri Setiawan
Asisten Pembimbing : Jan Meyer Saragih

1. Diagram Kelas

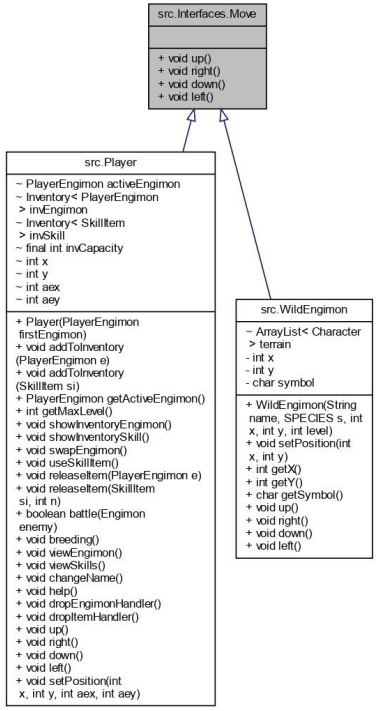
6.4 src.Interfaces.Info Interface Reference

Inheritance diagram for src.Interfaces.Info:



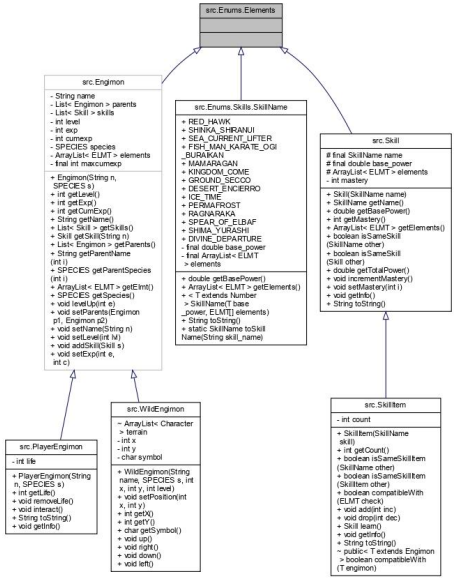
6.7 src.Interfaces.Move Interface Reference

Inheritance diagram for src.Interfaces.Move:



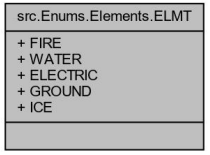
6.1 src.Enums.Elements Interface Reference

Inheritance diagram for src.Enums.Elements:



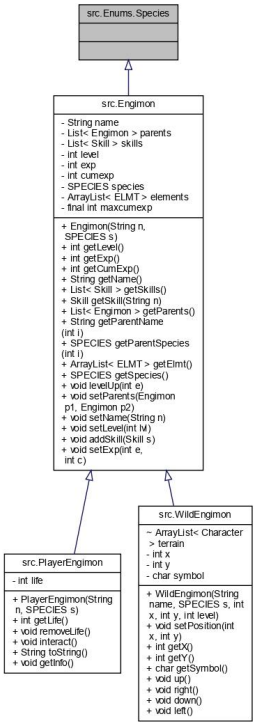
6.2 src.Enums.Elements.ELMT Enum Reference

Collaboration diagram for src.Enums.Elements.ELMT:



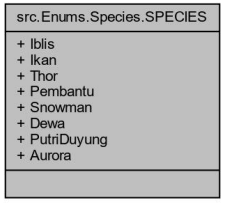
6.18 src.Enums.Species Interface Reference

Inheritance diagram for src.Enums.Species:



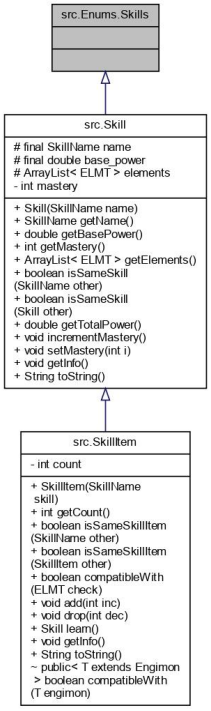
3.18 src.Enums.Species.SPECIES Enum Reference

Collaboration diagram for src.Enums.Species.SPECIES:

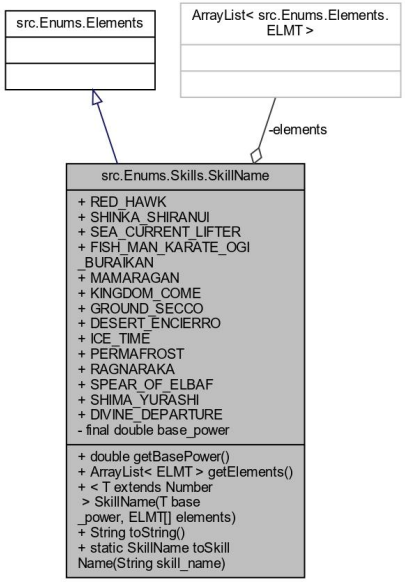


6.16 src.Enums.Skills Interface Reference

Inheritance diagram for src.Enums.Skills:

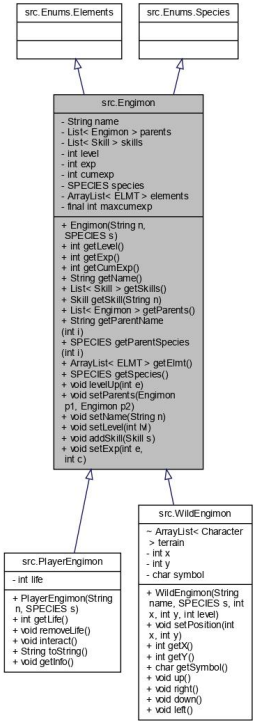


Collaboration diagram for src.Enums.Skills.SkillName:

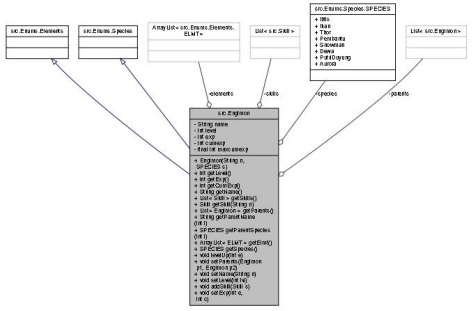


6.3 src.Engimon Class Reference

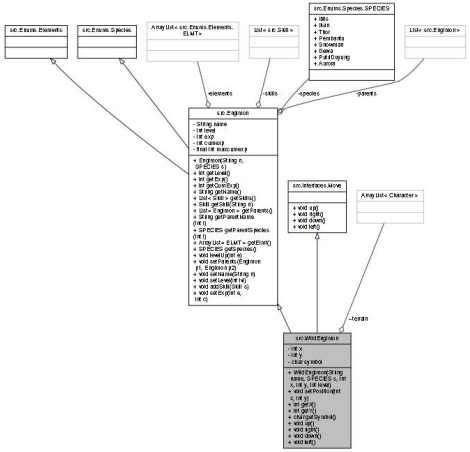
Inheritance diagram for src.Engimon:



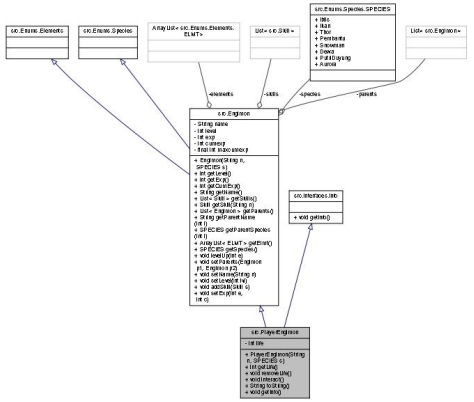
Collaboration diagram for src.Engimon:



Collaboration diagram for src.WildEngimon:

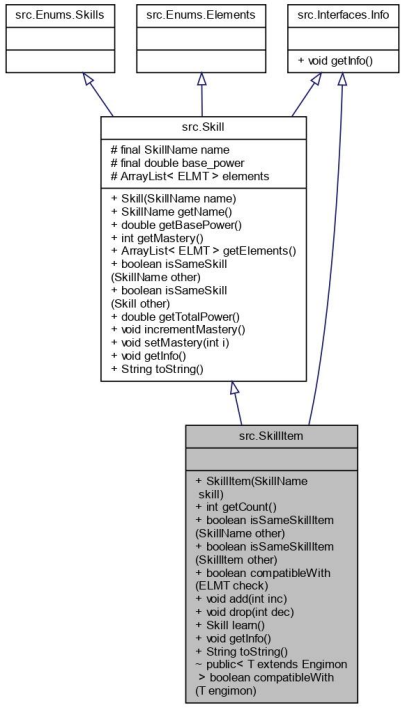


Collaboration diagram for src.PlayerEngimon:

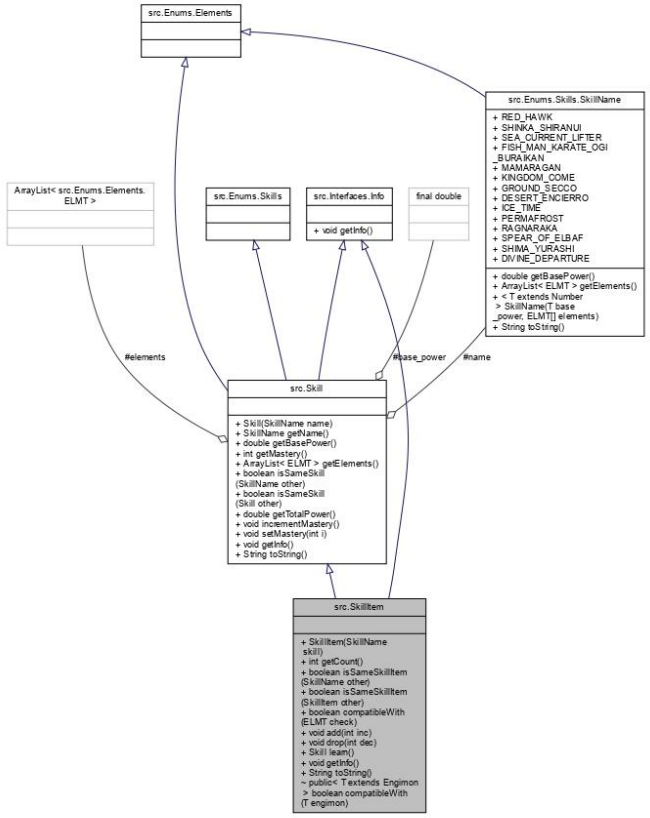


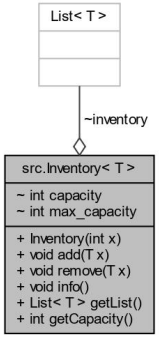
3.13 src.SkillItem Class Reference

Inheritance diagram for src.SkillItem:



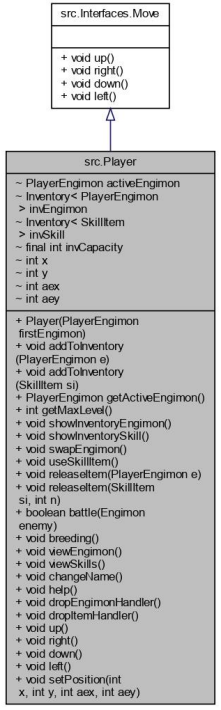
Collaboration diagram for src.SkillItem:



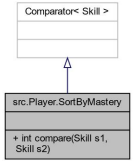

List< T >
~inventory
src.Inventory< T >
~ int capacity
~ int max_capacity
+ Inventory(int x)
+ void add(T x)
+ void remove(T x)
+ void info()
+ List< T > getList()
+ int getCapacity()

6.10 src.Player Class Reference

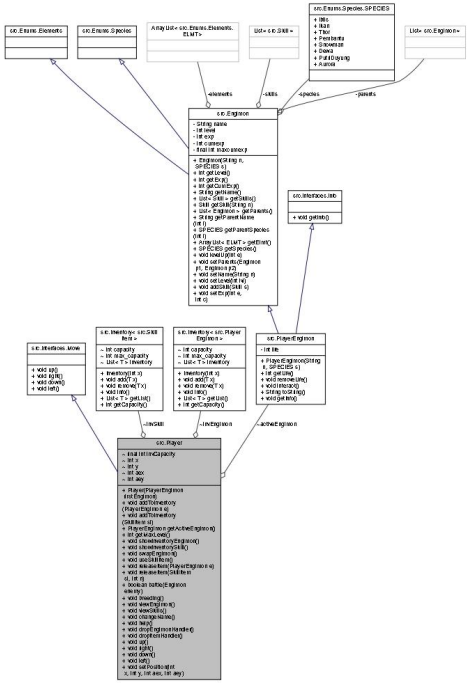
Inheritance diagram for src.Player:



Collaboration diagram for src.Player.SortByMastery:

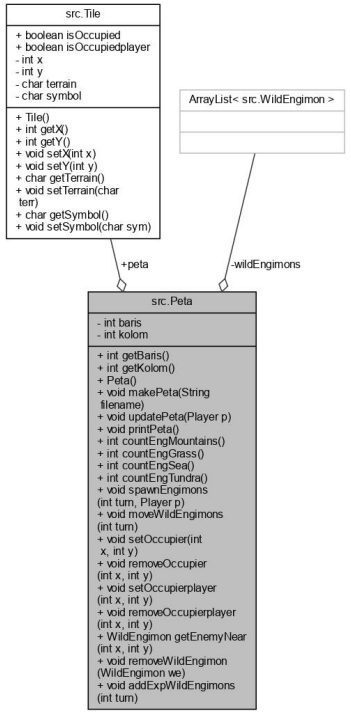


Collaboration diagram for src.Player:



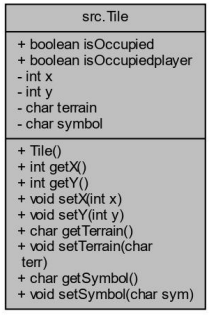
6.9 src.Peta Class Reference

Collaboration diagram for src.Peta:



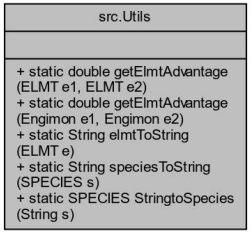
6.20 src.Tile Class Reference

Collaboration diagram for src.Tile:



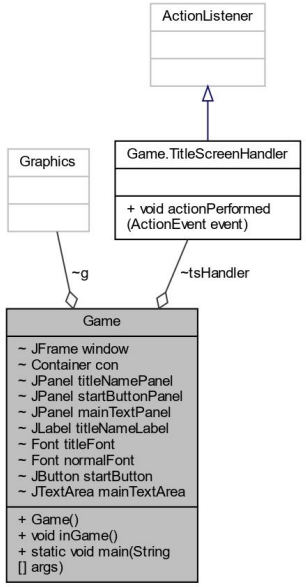
6.22 src.Utils Class Reference

Collaboration diagram for src.Utils:



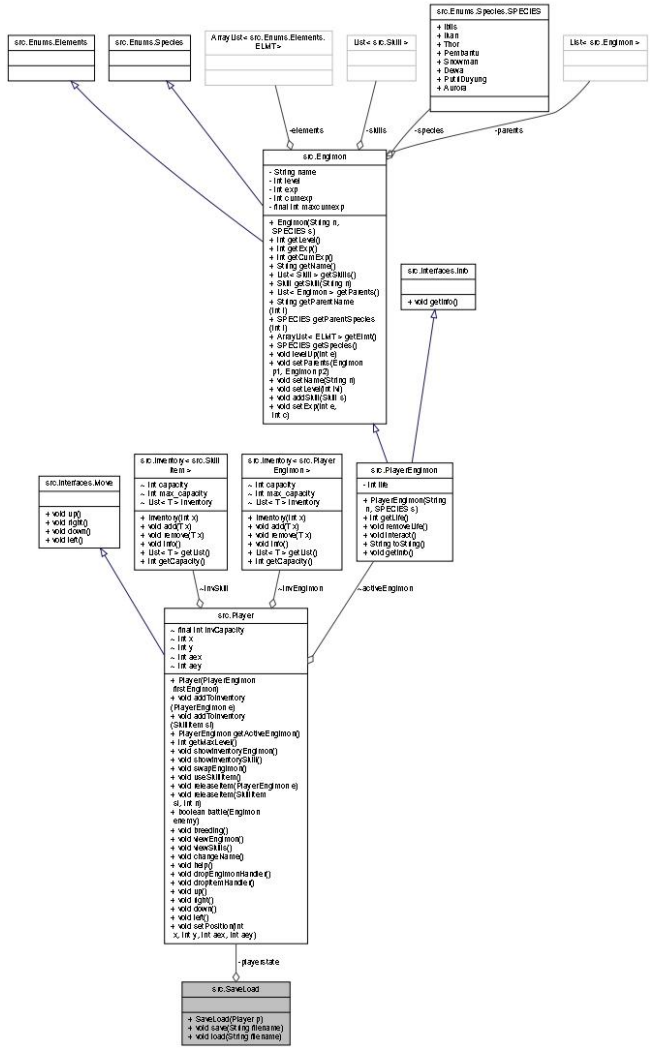
6.4 Game Class Reference

Collaboration diagram for Game:



6.12 src.SaveLoad Class Reference

Collaboration diagram for src.SaveLoad:



Pemilihan interface info dilakukan karna ada beberapa objek yang membutuhkan method yang sama yaitu getInfo(), yaitu PlayerEngimon, Skill, SkillItem. Selain itu, PlayerEngimon dan WildEngimon juga dapat bergerak, sehingga bisa dibuat interface Move. Untuk elements, kami menggunakan enum dalam pengaksesannya, kami juga menerapkan enums pada SkillName dan Species. Kendalanya adalah ada beberapa method harus mengecek satu-persatu dari konten enums untuk mengeluarkan output tertentu, Selain itu dalam method save, harus menuliskan string ke dalam textfile, akan tetapi banyak objek yang bukan merupakan string, oleh karena itu objek-objek tersebut harus diubah ke string terlebih dahulu dengan object Utils yang memiliki method toString(). Lalu, ada objek SkillItem yang merupakan inheritance dari Skill. Untuk Engimon, kami melakukan inheritance menjadi 2 anak yaitu WildEngimon dan PlayerEngimon, keduanya memiliki atribut Life yang berbeda, untuk penyimpanan Skill dan Species pada Engimon dapat dilakukan dengan Collection dari Java yaitu ArrayList. Objek Player memiliki atribut inventory Engimon beserta inventory Skill, lalu posisi dari Player tersebut dalam map.

Penerapan Konsep OOP

1.1. Polymorphism

```
public void addToInventory(PlayerEngimon e) {
    this.invEngimon.add(e);
}

public void addToInventory(SkillItem si) {

    // int idx = invSkill.getList().indexOf(si);
    if (this.invSkill.getList().stream().anyMatch(i -> i.isSameSkillItem(si))) {
        for (SkillItem skillItem : this.invSkill.getList()) {
            if (skillItem.isSameSkillItem(si)) {
                skillItem.add(1);
                break;
            }
        }
    } else {
        this.invSkill.add(si);
    }
}
```

```
public class Utils {
    public static double getElmtAdvantage(ELMT e1, ELMT e2) {···

    /*···
    public static double getElmtAdvantage(Engimon e1, Engimon e2) {···
```

Polymorphism memanfaatkan dua atau lebih *method* dengan nama yang sama, namun *parameter* yang berbeda. Dalam tugas ini, kami mengimplementasikan *polymorphism* pada method *addToInventory* dan *getElmtAdvantage*. Dengan memanfaatkan konsep ini pada method *addToInventory*, tidak perlu dibuat method berbeda untuk menambahkan suatu entitas (pada kasus ini yaitu *PlayerEngimon* dan *SkillItem*) ke

dalam inventory yang dimiliki player. Method *getElmtAdvantage* juga memanfaatkan parameter khusus untuk membandingkan elemen (ELMT) dan untuk membandingkan nilai advantage dari dua engimon.

1.2. Inheritance/Composition/Aggregation

```java
public class SkillItem extends Skill implements Info {

    private int count;

    public SkillItem(SkillName skill) {
        super(skill);
        this.count = 1;
    }

    public int getCount() {
        return count;
    }
```

```java
public class WildEngimon extends Engimon implements Move {
    private int x;
    private int y;
    private char symbol;
    ArrayList<Character> terrain;

    public WildEngimon(String name, SPECIES s, int x, int y
        super(name, s);
        this.x = x;
        this.y = y;
```

```
public class PlayerEngimon extends Engimon implements Info {
    private int life;

    public PlayerEngimon(String n, SPECIES s) {
        super(n, s);
        this.life = 3;
    }

    public int getLife() {
        return this.life;
    }

    public void removeLife() {
        this.life--;
    }
```

Inheritence digunakan untuk menerima struktur objek dari *superclass*-nya, sehingga objek *derived class*-nya memiliki *field* dan *method* yang sama dengan *superclass*-nya.

Untuk kasus *SkillItem* yang meng-*inherit* dari kelas *Skill*, hal ini diperuntukkan agar memudahkan penggunaan nanti di *inventory*. Kelas *SkillItem* memiliki properti yaitu *count* untuk mencatat jumlah *SkillItem* yang dimiliki oleh player.

Untuk kasus *PlayerEngimon* dan *WildEngimon*, keduanya meng-*inherit* dari kelas *Engimon*. Hal ini diperuntukkan agar kedua jenis kelas turunan *Engimon* tersebut selalu memiliki *base attribute/method* yang sama sehingga dapat digunakan secara universal. Hal ini juga mempermudah apabila ada method yang ingin memproses *Engimon*, baik *WildEngimon* maupun *PlayerEngimon*.

1.3. Abstract Class

```
abstract class Engimon implements Elements, Species {

    private String name;
    private List<Engimon> parents;
    private List<Skill> skills;
    private int level;
    private int exp;
    private int cumexp;
    private SPECIES species;
    private ArrayList<ELMT> elements;
    private final int maxcumexp = 10000;
```

Abstract class digunakan untuk di-*inherit* kemudian diimplementasikan method-methodnya. *Engimon* dibuat menjadi sebuah kelas abstrak, karena instansiasi pada permainan nantinya hanya bisa terdapat 2 jenis yaitu *PlayerEngimon* dan *WildEngimon*. Kelas abstract ini tujuannya untuk mencatat dan menginisiasi segala kebutuhan *Engimon* bertipe apapun yang nantinya akan di-*inherit* oleh kelas lain yang mengimplementasikan *Engimon*.

1.4. Interface

```java
package src.Interfaces;
public interface Info {
    /**
     * Menampilkan info dari sebuah objek
     * (dapat berupa engimon atau skill item)
     */
    void getInfo();
}
```

```java
package src.Interfaces;

import src.Peta;

public interface Move {
    public void up();

    public void right();

    public void down();

    public void left();
}
```

Interface digunakan untuk menginisiasi sebuah method yang nantinya dapat diimplementasikan oleh kelas-kelas yang membutuhkannya.

Untuk interface *Info*, method yang tersedia adalah *getInfo()*. Method ini dibuat *interface* karena akan ada beberapa kelas yang tidak berhubungan secara langsung memanfaatkan dan membutuhkan method untuk menampilkan informasi, antara lain *Skill, SkillItem,* dan *PlayerEngimon*.

Interface move memiliki 4 method untuk bergerak pada permainan nanti, yaitu *up, right, down,* dan *left*. Interface ini dibuat karena terdapat beberapa kelas yang akan mengimplementasikannya nanti, antara lain *Player* dan *WildEngimon*.

1.5. Generic Type & Wildcards

```java
public class Inventory<T>{
    int capacity;
    int max_capacity;
    List<T> inventory;

    public Inventory(int x){
        this.max_capacity=x;
        this.capacity=0;
        this.inventory= new ArrayList<T>();
        System.out.println("Inventory dengan kapasitas "+ this.max_capacity);
    }
```

```java
public<T extends Engimon> boolean compatibleWith(T engimon) {
    ArrayList<ELMT> engimon_elements = engimon.getElmt();
    for (ELMT elements : engimon_elements) {
        if (this.elements.contains(elements)) {
            return true;
        }
    }
    return false;
}
```

```java
<T extends Number> SkillName(T base_power, ELMT[] elements) {
```

Penggunaan generic pada kelas *Inventory* diperuntukkan agar suatu instansiasi *inventory* dapat menampung berbagai jenis objek. Pada implementasi ini, *Inventory* nantinya digunakan untuk menjadi properti pada *Player*, di mana player memiliki *Inventory of PlayerEngimon* (Inventory<PlayerEngimon>) dan *Inventory of SkillItem* (Inventory<SkillItem>).

WildCard yang digunakan terdapat pada method *compatibleWith* pada kelas *SkillItem*. Wildcard ini memastikan bahwa tipe yang menjadi parameter nantinya merupakan kelas atau turunan dari kelas *Engimon*. Pada method *SkillName* juga terdapat wildcard yang menerima tipe

berupa kelas atau turunan dari *Number*. Hal ini memudahkan untuk input nilai yang bisa berupa integer, double, float, dan lainnya yang merupakan bagian dari kelas *Number*.

1.6. Exception Handling

```
public void drop(int dec) throws Exception {
    if (this.count < dec) {
        throw new Exception();
    }
    this.count -= dec;
}
public Skill learn() throws Exception {
    if (this.count <= 0) {
        throw new Exception();
    }
    this.count--;
    return new Skill(this.name);
}
```

```
public void makePeta(String filename) throws FileNotFoundException {
    File file = new File(filename);
    Scanner scan = new Scanner(file);
    String lines = "";
    int i = 0;
    String[] read = new String[baris];
    char[][] map = new char[baris][kolom];
    while (scan.hasNextLine()) {
        lines = scan.nextLine();
        read[i] = lines;
        for (int j = 0; j < this.kolom; j++) {
            map[i] = read[i].toCharArray();
        }
        i++;
    }
    for (i = 0; i < this.baris; i++) {
        for (int j = 0; j < this.kolom; j++) {
            this.peta[i][j].setX(i);
            this.peta[i][j].setY(j);
            this.peta[i][j].setTerrain(map[i][j]);
            this.peta[i][j].setSymbol(map[i][j]);
        }
    }
}
```

```
ublic void swapEngimon() {
    System.out.println("Choose Engimon to swap");
    this.invEngimon.info();
    if (invEngimon.getCapacity() > 1) {
        System.out.format("[1-%d] (0 to exit): ", invEngimon.getCapacity());
    }

    // getting input from player
    int selected;
    Scanner reader = new Scanner(System.in);
    while (true) {
        try {
            if (reader.hasNextInt()) {
                selected = reader.nextInt();
                if (selected < 0 || selected > invEngimon.getCapacity()) {
                    throw new InputMismatchException();
```

```
case "d":
    try {
        if (!P.peta[p1.x][p1.y+1].isOccupied
        && !P.peta[p1.x][p1.y+1].isOccupiedplayer) {
        P.removeOccupierplayer(p1.x, p1.y);
        P.removeOccupierplayer(p1.aex, p1.aey);
        p1.right();
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("tidak bisa keluar map");
    }
    break;
```

Terdapat beberapa exception handling yang diterapkan. Tujuan dari seluruh exception handling adalah sama, yaitu untuk menangkap error agar program tidak keluar dari “rules” yang ditentukan seperti index array out of bounds, ioexception, dll.

1.7. Java Collection

```
abstract class Engimon implements Elements, Species {

    private String name;
    private List<Engimon> parents;
    private List<Skill> skills;
    private int level;
    private int exp;
    private int cumexp;
    private SPECIES species;
    private ArrayList<ELMT> elements;
    private final int maxcumexp = 10000;

    public Engimon(String n, SPECIES s){
        this.name = n;
        this.parents = new ArrayList<Engimon>();
        this.skills = new ArrayList<Skill>();
        this.elements = new ArrayList<ELMT>();
        this.level = 1;
        this.exp = 0;
        this.cumexp = 0;
        this.species = s;
```

Program kami mengimplementasikan ArrayList dari Java Collection Framework, ArrayList digunakan untuk mengakses object didalamnya berdasarkan indeks dan memiliki beberapa base method yang berguna dalam pengerjaan tugas.

1.8. Java API

```
public class SortByMastery implements Comparator<Skill> {
    public int compare(Skill s1, Skill s2) {
        return s1.getMastery() - s2.getMastery();
    }
}
```

```
<T extends Number> SkillName(T base_power, ELMT[] elements) {
    this.base_power = base_power.doubleValue();
    this.elements = Arrays.stream(elements)
            .collect(Collectors.toCollection(ArrayList::new));
}
```

Java API Comparator Interface digunakan untuk membuat custom sorting yang digunakan dalam method breeding. Selain itu, kami juga memanfaatkan API Stream dalam konstruktor enum SkillName untuk memanipulasi array ELMT dan mengumpulkan isinya dalam collection ArrayList.

2. External Library

Library yang digunakan adalah Java Swing yang digunakan dalam proses pembuatan Graphical User Interface.

3. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
I.1. Engimon	13519126, 13519154	13519126, 13519154, 13519161
I.2 Skill	13519148	13519148
I.3 Player	13519154, 13519123	13519154, 13519123
I.3.b Inventory	13519111	13519111
I.4 Battle	13519154	13519154
I.5 Breeding	13519123	13519123
I.6 Peta	13519161	13519161, 13519154
II.1 Graphical User Interface	13519111, 13519154, 13519161	13519111, 13519154
II.2 Save and Load	13519126	13519126