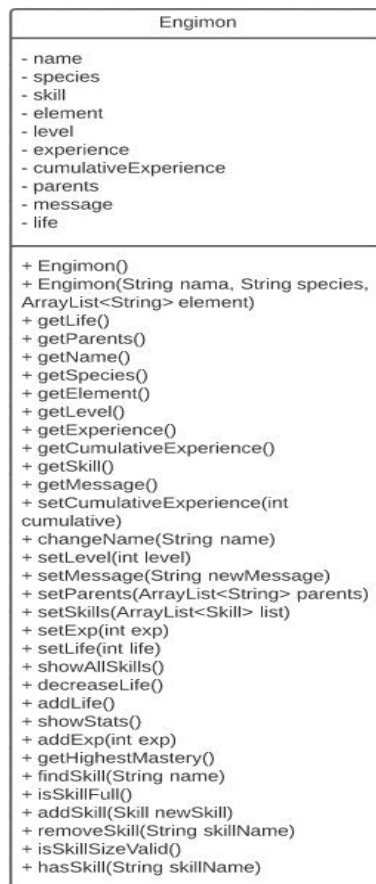


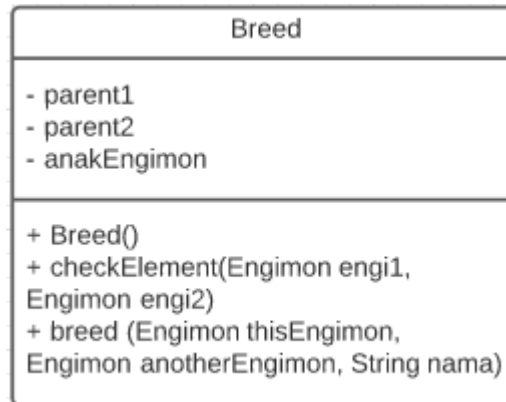
Kelas : 04
 Nama Kelompok : POO
 1. 13519180 / Karel Renaldi
 2. 13519185 / Richard Rivaldo
 3. 13519191 / Kevin Ryan
 4. 13519215 / Leonard Matheus
 5. 13519217 / Hughie Alghaniyyu Emiliano
 Asisten Pembimbing : Willy Santoso

1. Diagram Kelas



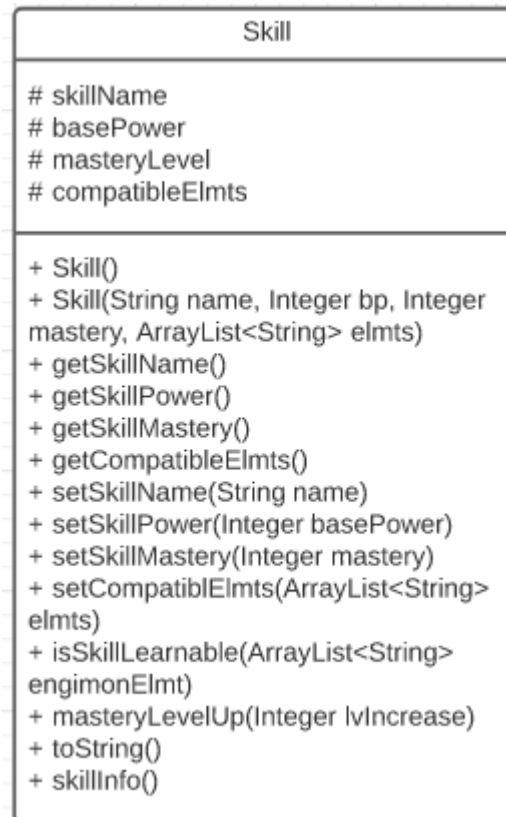
Engimon

Kelas Engimon dirancang sedemikian rupa untuk menyimpan informasi-informasi yang diperlukan objek Engimon. Di dalam kelas ini, disimpan atribut-atribut yang diperlukan Engimon dan dilengkapi dengan methods yang dapat digunakan untuk melakukan operasi-operasi tertentu yang berhubungan dengan Engimon dan segala atributnya.



Breed

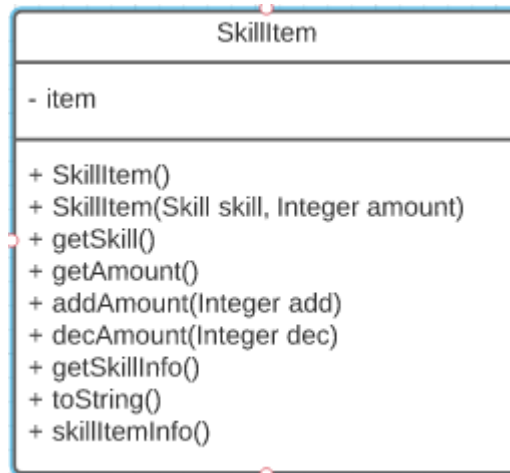
Kelas breed dirancang sedemikian rupa untuk menyimpan atribut berupa objek Engimon parent1, parent2, dan anakEngimon. Objek breed ini dilengkapi dengan methods yang dapat digunakan untuk melakukan breeding antara parent1 dan parent2, dan memberikan hasil berupa engimon yang dimasukkan statsnya ke dalam anakEngimon.



Skill

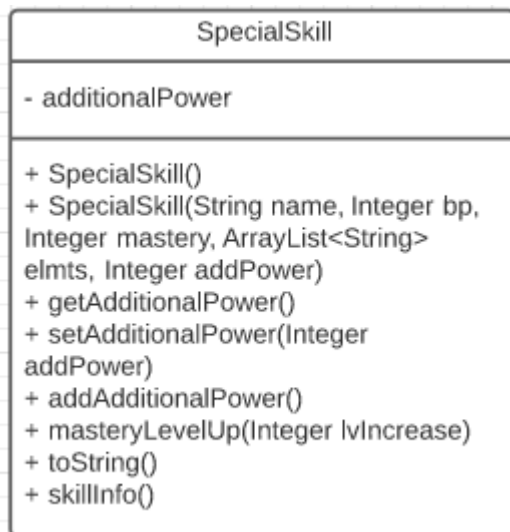
Kelas skill dirancang sedemikian rupa untuk menyimpan atribut-atribut skill yang diperlukan untuk keberjalanan eksekusi program. Objek skill dilengkapi dengan methods yang dapat digunakan untuk melakukan operasi yang berhubungan atau melibatkan skill. Selain itu, kelas *Skill* nantinya juga akan menjadi *parent class* untuk kategori-kategori *Skill* lain sehingga *code* menjadi *reusable*.

SkillItem



Kelas `SkillItem` dirancang sedemikian rupa untuk menyimpan atribut item yang bertipe `Pair<Skill, int>`. Setiap item merupakan sebuah skill yang dapat dipelajari oleh engimon. Integer pada pair merepresentasikan jumlah item dalam inventory player. Kelas ini juga dilengkapi dengan methods yang dapat digunakan untuk melakukan operasi yang melibatkan `SkillItem`. Dengan menyimpan atribut jumlah tersebut, maka *inventory* hanya perlu menangani *Skill Item* sebagai *black box* dan tidak perlu berurusan secara langsung dengan jumlah dari setiap *Skill Item*.

SpecialSkill



Kelas `SpecialSkill` merupakan kelas turunan dari `Skill`. Kelas `SpecialSkill` memiliki atribut `additionalPower` yang berisi power lebih dari suatu `Skill`. Kelas ini dilengkapi dengan methods yang dapat digunakan untuk menjalankan operasi yang berkaitan dengan `SpecialSkill`. Kelas ini menerapkan perilaku *method overriding* dari kelas *parent* sehingga penyesuaian perilaku kelas ini bisa dilakukan dengan benar.

UniqueSkill

Kelas UniqueSkill merupakan kelas turunan dari Skill. Kelas ini berisi uniqueSpecies atau spesies engimon yang dapat mempelajari skill ini. Kelas ini dilengkapi dengan methods yang bertujuan untuk melakukan operasi yang berkaitan dengan UniqueSkill. Sama seperti SpecialSkill, kelas ini menerapkan perilaku *method overriding* dari kelas *parent* sehingga penyesuaian perilaku kelas ini bisa dilakukan dengan benar.

UniqueSkill
- uniqueSpecies
+ UniqueSkill() + UniqueSkill(String name, Integer bp, Integer mastery, ArrayList<String> elmts, String species) + getUniqueSpecies() + setUniqueSpecies(String species) + isSkillLearnable(ArrayList<String> engimonElmt, String species) + toString() + skillInfo()

Battle

Kelas battle merupakan suatu objek yang atributnya merupakan data-data yang diperlukan pada suatu battle. Kelas ini dilengkapi dengan methods yang dapat dieksekusi untuk melakukan operasi yang diperlukan untuk menjalankan battle antar engimon.

Battle
- player - engimonPlayer - engimonWild - engiPlayerPower - engiWildPower - playerMult - wildMult - playerSkillPower - wildSkillPower
+ Battle(Player player, Engimon engimonWild) + checkMultiplier(String elemen1, String elemen2) + setMultiplier() + setSkillPower() + setpower() + showEngimonPower() + doBattle() + cancelBattle() + startBattle()

Player

Kelas player memuat atribut informasi mengenai player pada game, dimulai dari activeEngimon, hingga playerName. Kelas ini merupakan objek player yang akan ditampilkan pada program. Kelas ini juga dilengkapi dengan methods yang dapat melakukan operasi yang melibatkan player pada game. Kelas Player akan berinteraksi langsung dengan kelas Play karena kedua kelas saling berkaitan dan terhubung melalui *Graphical User Interface* yang dibangun. Kelas ini dapat dikatakan sebagai kelas yang menjadi sentral semua proses permainan sehingga tidak heran apabila di dalamnya banyak terkandung prinsip komposisi dengan objek-objek dari kelas lain.

Player
<ul style="list-style-type: none"> - speed - velocity - collisionLayer - s - a - w - d - play - answer - orientation - animationTime - activeEngimon - maxCapacity - engimonInvent - skillInvent - playerName
<pre> + Player(Animation<TextureRegion> s, Animation<TextureRegion> a, Animation<TextureRegion> w, Animation<TextureRegion> d, TiledMapTileLayer collisionLayer, int x, int y, Play play, Engimon active) + getSpeed() + setSpeed(float speed) + getGravity() + setGravity(float gravity) + setVelocity(Vector2 velocity) + getCollisionLayer(TiledMapTileLayer) + getOrientation() + xpos() + ypos() + draw(Batch batch) + isCollision(float x, float y) + move(int x, int y) + move(float delta) + notOutOfBound(float idx) + tileNotExists(float tileX, float tileY, TiledMapTileLayer map) + keyDown(int keycode) + keyUp(int keycode) + keyTyped(char character) + touchDown(int screenX, int screenY, int pointer, int button) + touchUp(int screenX, int screenY, int pointer, int button) + touchDragged(int screenX, int screenY, int pointer) + mouseMoved(int screenX, int screenY) + scrolled(float amountX, float amountY) + changePlayerName(String newName) + getActiveEngimon() + showHelp() + setActiveEngimon(Engimon active) + swapActiveEngimon(String newEngi) + getPlayerName() + getEngimonInvent() + getSkillInvent() + getTotalItems() + isStillEmpty() + isLosing() + addEngimon(Engimon newEngimon) + isAlreadyInInvent(String skillName) + addSkillItem(SkillItem newSkillItem) + showAllOwnedEngi() + showEngiInfo(String engiName) + showActiveEngi() + interactWithEngi() + showAllOwnedItem() + isAlreadyOwned(String engiName) + decOrRemove(int index) + learnNewSkill(String skillName, String engiName) + sortingItemInventory() + sortingEngimonInventory() + breed(String engi1, String engi2, String namaAnak) + setBattle(Battle battle) + doBattle(Battle battle, String answer) </pre>

Inventory
array_inventory
+ Inventory() + add(T inventory_item) + remove(int index) + set(int index, T element) + size() + getItemList() + isEmptyInvent()

Inventory

Kelas ini memiliki atribut array_inventory, yaitu sebuah ArrayList yang dapat diisi dengan elemen-elemen. Kelas ini juga dilengkapi dengan methods yang dapat digunakan untuk melakukan operasi yang berkaitan dengan inventory player

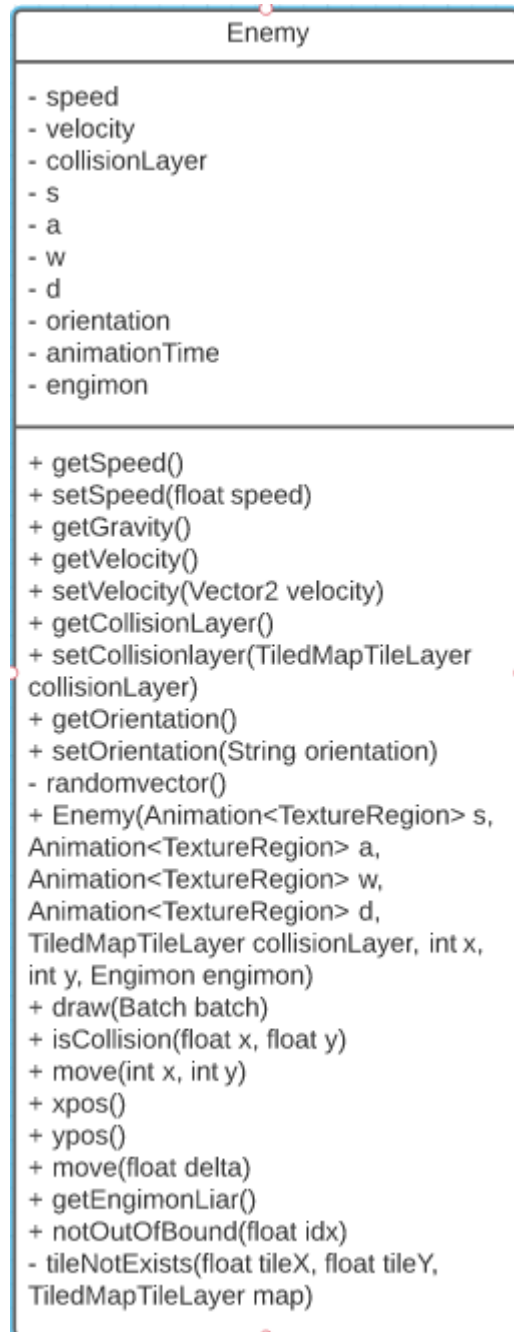
IndexOutOfBoundsException
- errorMessage
+ IndexOutOfBoundsException() + getErrorMessage()

IndexOutOfBoundsException

Kelas ini berisi atribut String errorMessage yang akan ditampilkan dengan method getErrorMessage(). Kelas ini digunakan untuk handle exception index out of bound.

Enemy

Kelas ini berisi atribut-atribut yang diperlukan untuk menset enemy atau wild engimon pada game, seperti kecepatan gerakannya. Kelas ini juga dilengkapi dengan methods yang dapat digunakan untuk melakukan operasi yang melibatkan enemy pada game.



EnemyList

EnemyList
<ul style="list-style-type: none"> - enemylist - maxEnemy - map - renderer - camera - enemyAtlas - s - a - w - d
<ul style="list-style-type: none"> + getMaxEnemy() + setMaxEnemy(int maxEnemy) + getEnemylist() + setEnemylist(Array<Enemy> enemylist) EnemyList(int maxEnemy, TiledMap map, OrthogonalTiledMapRenderer renderer, OrthographicCamera camera) + addEnemy + getEnemyTerdekat

Kelas ini berisi atribut-atribut yang berhubungan dengan enemy secara keseluruhan dalam permainan, misalnya nilai maksimum enemy yang diperbolehkan dalam permainan, dll. Kelas ini juga dilengkapi dengan methods yang dapat digunakan untuk beroperasi pada kelas EnemyList.

Skills

Skills
<ul style="list-style-type: none"> + fireSkills() : ArrayList<Skill> + waterSkills() : ArrayList<Skill> + electricSkills() : ArrayList<Skill> + iceSkills() : ArrayList<Skill> + groundSkills() : ArrayList<Skill> + createSkillItem(ArrayList<Skill> skills) : ArrayList<SkillItem> + main(String[] args)

Skills merupakan kelas bantuan kosong yang memiliki banyak *method* statik yang digunakan untuk melakukan *generation Skill* dan *SkillItem*. Dengan desain demikian, maka kelas Skill dan SkillItem tidak perlu lagi menangani pembuatan *'database'* Skill dan SkillItem yang nantinya akan digunakan di dalam permainan.

Kelas-Kelas Lain

Option
<ul style="list-style-type: none"> - selectedIdx - List<Image> arrows - List<Label> options - Table uiContainer
<ul style="list-style-type: none"> + addOption(String newOption) + moveDown() + moveUp() + getSelectedOptionIndex() + changeArrow() + resetOption()

Selain kelas-kelas di atas yang menjadi entitas permainan, juga terdapat beberapa kelas lain yang digunakan sebagai komponen permainan. Kelas-kelas berikut ini merupakan komponen permainan yang dibangun dengan menggunakan LibGDX dan Scene2D serta Box2D. Berikut merupakan salah satu komponen kelas tersebut, yaitu Option, yang digunakan untuk menampilkan *Option Box* pada permainan.

```
public class Play implements Screen {
    // Tiled Map
    private TiledMap map;
    // Renderer
    private OrthogonalTiledMapRenderer renderer;
    // Camera
    private OrthographicCamera camera;
    // Player
    private Player player;
    // enemylist
    private EnemyList enemyList;
    // Player Atlas
```

Kelas Play merupakan komponen permainan yang berinteraksi langsung dengan *backend* permainan, yaitu kelas Player, dan kelas ini mengatur *screen* dan *input process* dari pengguna. Kelas PopUp, sama seperti kelas Option, merupakan komponen yang digunakan untuk menampilkan *Pop Up* baru di dalam permainan. Terakhir, kelas POO merupakan kelas yang nantinya akan menggunakan kelas Play untuk memulai atau melaksanakan permainan.

```
public class POO extends Game {

    @Override
    public void create () { setScreen(new Play()); }
}
```

2. Penerapan Konsep OOP

2.1. Polymorphism dan Inheritance

Polymorphism dan Inheritance yang diimplementasikan pada tugas kali ini terdapat pada kelas Skill. Kelas tersebut diinherit dan digunakan oleh kelas SpecialSkill dan kelas UniqueSkill. Alasan penggunaan polymorphism dan inheritance pada kelas ini karena kelas SpecialSkill dan UniqueSkill merupakan turunan dari kelas Skill yang memiliki karakteristik yang hampir sama.

Keuntungan dari menggunakan polymorphism dalam pengimplementasian Skill adalah kode yang dapat digunakan kembali dengan menerapkan konsep polymorphism. Selain itu, polymorphism mensupport nama variabel yang sama untuk berbagai tipe data dan mengurangi coupling yang terjadi di antara berbagai fungsionalitas.

Selain itu, kelas SkillItem dibangun dengan menggunakan konsep komposisi dan di dalamnya mengandung Skill yang berkorespondensi dengan item tersebut. Dengan demikian, SkillItem memiliki hubungan *has-a* dengan Skill dan SkillItem juga mencatat jumlah item tersebut di dalam *inventory*.

```
import java.util.*;

public class Skill {
    protected String skillName;
    protected Integer basePower;
    protected Integer masteryLevel;
    protected ArrayList<String> compatibleElmts = new ArrayList<String>();

    // Default constructor
    public Skill() {
        skillName = "sKILL";
        basePower = 0;
        masteryLevel = 1;
    }

    // User-Defined Constructor
    public Skill(String name, Integer bp, Integer mastery, ArrayList<String> elmts) {
        this.skillName = name;
        this.basePower = bp;
        this.masteryLevel = mastery;
        this.compatibleElmts = elmts;
    }
}
```

```
import java.util.*;

public class SpecialSkill extends Skill {
    private Integer additionalPower;

    // Default constructor
    public SpecialSkill() {
        super();
        this.additionalPower = 0;
    }

    // User-Defined Constructor
    public SpecialSkill(String name, Integer bp, Integer mastery, ArrayList<String> elmts, Integer addPower) {
        super(name, bp, mastery, elmts);
        this.additionalPower = addPower;
    }
}
```

2.2. Abstract Class

Abstract class yang diimplementasikan pada tugas kali ini terdapat pada kelas Engimon. Seluruh methods dari kelas Engimon diabstraksikan terlebih dahulu sebelum diimplementasikan. Hal ini bertujuan untuk memberikan gambaran kepada programmer lain yang akan menggunakan kelas engimon untuk mengimplementasikan kelas lainnya. Alasan abstract class diimplementasikan pada Engimon dikarenakan kelas Engimon banyak digunakan pada kelas-kelas lain selain kelas Engimon.

Keuntungan menggunakan abstract class dalam penulisan program ini adalah dapat membuat fungsionalitas yang dapat diimplementasikan atau dioverride oleh subclasses.

```
import java.util.*;

abstract class AbstractEngimon {
    abstract public String getName();
    abstract public String getSpecies();
    abstract public ArrayList<String> getElement();
    abstract public int getLevel();
    abstract public int getExperience();
    abstract public int getCumulativeExperience();
    abstract public ArrayList<Skill> getSkill();
    abstract public String getMessage();
    abstract public void setLevel(int level);
    abstract public void setMessage(String newMessage);
    abstract public void showStats();
    abstract public void addExp(int exp);
}

public class Engimon extends AbstractEngimon {
    private String name;
    private String species;
    private ArrayList<Skill> skill;
    private ArrayList<String> element;
    private int level;
    private int experience;
    private int cumulativeExperience;
    private ArrayList<String> parents;
    private String message;
    private int life;
```

2.3. Interface

Interface yang diimplementasikan pada tugas kali ini terdapat pada kelas Breed. Metode breeding pada kelas ini dituliskan dengan interface dan pengimplementasian kelas ini dituliskan dengan kata kunci implements. Hal ini bertujuan agar penggunaan kelas breeding akan menjadi lebih mudah ketika akan digunakan oleh kelas lain. Alasan interface diimplementasikan pada kelas Breed adalah untuk memberikan gambaran kepada programmer lain yang akan menggunakan kelas breeding dalam mengimplementasikan kelas lain.

Keuntungan menggunakan interface dalam penulisan program ini adalah kemampuan untuk mendefinisikan fungsionalitas sehingga dapat lebih mudah dipahami oleh orang lain yang akan menggunakan kelas Breed.

```
import java.util.*;

interface Breeding {
    public Engimon breed(Engimon thisEngimon, Engimon anotherEngimon, String nama);
    public Boolean checkElement(Engimon engi1, Engimon engi2);
}

public class Breed implements Breeding {
    Engimon parent1;
    Engimon parent2;
    Engimon anakEngimon;

    public static boolean containsSkill(ArrayList<Skill> listSkill, String skill) {
        for (Skill elmt : listSkill) {
            if (elmt.getSkillName() == skill) {
                return true;
            }
        }
        return false;
    }

    public Breed() {
        this.parent1 = null;
        this.parent2 = null;
        this.anakEngimon = null;
    }
}
```

2.4. Generic Type & Wildcards

Generic type dan wildcards yang diimplementasikan pada tugas kali ini terdapat pada kelas Inventory. Alasan penggunaan generic type dan wildcards dalam kelas ini adalah metode-metode pada kelas ini banyak yang menggunakan tipe data yang mungkin berbeda. Oleh karena itu, dengan pengimplementasian generic type dan wildcards, akan mempermudah penulisan program dan meningkatkan efektifitas dan efisiensi penulisan dan eksekusi program.

Keuntungan menggunakan generic type dan wildcards dalam penulisan program ini adalah kemampuan untuk tidak mendefinisikan tipe data parameter terlebih dahulu dalam pendefinisian kelas, interface, dan methods.

```
import java.util.*;

public class Inventory<T> {
    protected ArrayList<T> array_inventory;

    // Constructor
    public Inventory() {
        this.array_inventory = new ArrayList<>();
    }

    // Add item
    public void add(T inventory_item) {
        this.array_inventory.add(inventory_item);
    }
}
```

2.5. Exception Handling

Exception handling yang diimplementasikan pada tugas kali ini terdapat pada kelas inventory. Exception handling dalam kelas ini digunakan untuk handle error index out of bounds. Alasan penggunaan exception handling dalam kelas ini adalah untuk mengatasi kemungkinan munculnya error pada kelas ini.

Keuntungan menggunakan exception handling dalam penulisan program ini adalah kemampuan untuk handle error yang mungkin terjadi dalam program, yaitu index out of bounds.

```
public void remove(int index) throws IndexOutOfBoundsException {  
    if (index - 1 > this.array_inventory.size())  
        throw new IndexOutOfBoundsException();  
    this.array_inventory.remove(index);  
}  
  
// Set Item  
public void set(int index, T element) throws IndexOutOfBoundsException {  
    if (index - 1 > this.array_inventory.size())  
        throw new IndexOutOfBoundsException();  
    this.array_inventory.set(index, element);  
}
```

2.6. Java Collection

Beberapa java collection banyak digunakan dalam penulisan program, misalnya ArrayList, HashMap dan Vector. ArrayList banyak digunakan pada kelas-kelas seperti kelas Engimon menggunakan ArrayList untuk mengisi Skills. Vector digunakan pada kelas Map untuk beberapa hal, seperti penyimpanan posisi engimon liar, dll. HashMap digunakan untuk membantu pemrogram dalam melakukan *grouping* Item dan Engimon dalam algoritma *sorting* yang digunakan untuk menyortir item di dalam *Inventory* pemain.

Keuntungan menggunakan java collection dalam penulisan program ini adalah kemampuan untuk menggunakan java collection untuk memudahkan penulisan dan pembuatan program dalam mewujudkan beberapa fungsionalitas, misalnya untuk menyimpan data, dapat diwujudkan dengan menggunakan ArrayList. Koleksi lain seperti HashMap sangat membantu dalam proses penyortiran dikarenakan struktur ini memiliki efisiensi waktu yang sangat baik.

```

public class Engimon extends AbstractEngimon {
    private String name;
    private String species;
    private ArrayList<Skill> skill;
    private ArrayList<String> element;
    private int level;
    private int experience;
    private int cumulativeExperience;
    private ArrayList<String> parents;
    private String message;
    private int life;

    public void sortingEngimonInventory() {
        HashMap<String, Inventory<Engimon>> groupEngimon = new HashMap<String, Inventory<Engimon>>();

        for (Engimon engi : this.engimonInvent.getItemList()) {
            if (engi.getElement().size() == 1) {
                if (groupEngimon.get(engi.getElement().get(0)) != null) {
                    groupEngimon.get(engi.getElement().get(0)).add(engi);
                } else {
                    Inventory<Engimon> inventEngi = new Inventory<Engimon>();
                    inventEngi.add(engi);
                    groupEngimon.put(engi.getElement().get(0), inventEngi);
                }
            }
        }
    }
}

```

2.7. Java API

Java API yang digunakan dalam penulisan program ini beberapa di antaranya adalah comparator dan exception, beserta API-API pada LibGDX, yaitu Screens, InputProcessor, Game. API yang digunakan bertujuan untuk mempermudah penyelesaian permasalahan yang muncul pada program ini. Misalnya, comparator digunakan untuk melakukan pengurutan, dan API-API LibGDX untuk mempermudah dalam pembuatan Graphical User Interface pada program. Selain itu, Java API lain yang digunakan adalah *Exception*.

Keuntungan menggunakan Java API dalam penulisan program ini adalah peningkatan efisiensi dalam penulisan program dan juga eksekusi program dikarenakan hasil dari Java API dapat digunakan secara langsung dan dapat digunakan oleh setiap channel.


```

class SortSkill implements Comparator<SkillItem> {
    public int compare(SkillItem skillItem1, SkillItem skillItem2) {
        return skillItem2.getSkill().getSkillPower() - skillItem1.getSkill().getSkillPower();
    }
}

class SortEngimon implements Comparator<Engimon> {
    public int compare(Engimon engi1, Engimon engi2) { return engi2.getPosition().x - engi1.getPosition().x; }
}

public class Player extends Sprite implements InputProcessor {
    // Gravity and speed of the entity
    private float speed = 60 * 2, gravity = 60 * 1.8f;
    // Velocity of the player entity
    private Vector2 velocity = new Vector2();
    // Layer of blocks with collisions

    public class Play implements Screen {
        // Tiled Map
        private TiledMap map;
        // Renderer
    }
}

```

2.8. Lain-Lain

Selain konsep-konsep OOP wajib yang telah dispesifikasikan, terdapat beberapa konsep OOP lain yang diimplementasikan untuk mempermudah penulisan program dan meningkatkan efisiensi program.

2.8.1. Pair

Pair yang digunakan pada Map dan SkillItems digunakan untuk menggabungkan 2 hal, misalnya engimon dengan posisi. Di dalam program, *Pair* yang dipakai adalah *Pair* buatan sendiri, karena Java sendiri hanya mempunyai *Pair* di JavaFX. Dengan menggunakan *Pair*, maka kita bisa menggabungkan dua hal yang berkaitan satu sama lain, misalnya *skill* dan jumlah item yang ditampung di dalam kelas *Skill Items*.

```

public class Map {
    private int xmax;
    private int ymax;
    private int maxEngimonLiar;
    private int setCapital; /* X >= setCapital (Huruf Besar), X < setCapital (Huruf Kecil)*/
    private int dimensiWater;
    private Position playerPosition;
    private Position activeEngimonPosition;
    ArrayList<ArrayList<String>> peta = new ArrayList<ArrayList<String>>() ;
    ArrayList<Pair<Position, Engimon>> engimonLiar = new ArrayList<Pair<Position, Engimon>>();
}

```

2.8.2. String Builder

String builder digunakan untuk membangun string. Konsep ini diimplementasikan pada kelas Skill untuk menghasilkan String yang nantinya digunakan untuk menampilkan informasi mengenai objek suatu kelas. Dengan *String Builder* ini, maka nantinya string yang dibuat menjadi lebih cepat dan tidak memakan banyak memori yang dihasilkan karena sifat String yang *immutable*.

```

public String toString() {
    StringBuilder sb = new StringBuilder();
    if (this.compatibleElmts.isEmpty()) {
        sb.append("-----");
    }
    else {
        for (String elmts : this.compatibleElmts) {
            sb.append("|");
            sb.append(elmts);
        }
        sb.append("|");
    }
    return String.format(
        "Skill Name          : %s\nBase Power          : %d\nMastery Level      : %d\nCompatible Elements : %s\n",
        this.skillName, this.basePower, this.masteryLevel, sb.toString());
}

```

2.8.3. Method Overriding

Method overriding digunakan untuk melakukan *overriding* suatu metode yang ada di *parent class* ke dalam kelas anak yang mewarisi kelas tersebut. Method overriding digunakan pada Skill dan SkillItem untuk melakukan penyesuaian terhadap perilaku kelas-kelas tersebut sesuai dengan kategori masing-masing kelas.

```
// Raise the mastery level
public void masteryLevelUp(Integer lvIncrease) {
    Integer currentLevel = this.getSkillMastery();
    if (this.getSkillMastery() + lvIncrease < 3) {
        this.setSkillMastery(this.getSkillMastery() + lvIncrease);
    } else {
        this.setSkillMastery(3);
    }
    Integer bound = this.getSkillMastery() - currentLevel;

    for(int i = 0; i < bound; i++){
        this.addAdditionalPower();
    }
}
```

```
// String for Skill Info
public String toString() {
    String skillInfo = super.toString();
    return String.format("%sSkill Grade      : Unique Skill\nUnique Species      : %s\n",
        skillInfo, this.uniqueSpecies);
}
```

Keuntungan menggunakan beberapa konsep OOP ini adalah peningkatan efisiensi penulisan program dikarenakan fungsionalitas konsep-konsep tersebut yang dapat digunakan untuk menyelesaikan beberapa permasalahan yang muncul pada saat penulisan program. Selain itu, beberapa konsep ini akan memudahkan struktur program, dan akan meningkatkan efisiensi memori dan waktu.

3. Bonus Yang dikerjakan

3.1. Bonus Kreasi Mandiri

Tampilan GUI yang dibuat menggunakan *LibGDX* memiliki desain peta yang dibuat dengan menggunakan *Tiled*. Dengan *Tiled*, maka berkas peta yang dihasilkan adalah berkas dengan format *.tmx*, dan komponen-komponen peta yang dibuat akan menggunakan berkas lain dengan format *.tsx*. Peta kemudian dirender ke dalam *LibGDX* dan ditampilkan pada saat program dimulai. Untuk tampilan pemain, dibuat berkas dengan format *.pack* untuk membangun animasi-animasi yang dibutuhkan pemain saat bergerak.

Setiap blok yang ada di dalam peta akan diberikan properti untuk menandakan bahwa blok tersebut bisa dilewati atau tidak. Dengan demikian, tercipta sistem *collision detection* sehingga pemain hanya bisa melewati blok-blok tertentu sesuai dengan desain peta. Selain itu, pada saat bergerak pemain akan memiliki tampilan *Sprite* yang berbeda sesuai dengan arah dan gerakan yang dilakukan pemain. Dengan demikian, gerakan yang dilakukan pemain akan mendapatkan animasi dan menjadi tidak kaku. Untuk bergerak, pemain bisa menggunakan WASD atau tanda panah.

Selain itu, sebagai konsekuensi dari spesifikasi program pada kelas *Skill*, maka pada *Skill* dibangun 3 buah kategori. Kategori pertama, yaitu *Skill*, merupakan kelas yang masuk ke dalam kategori *Basic Skill*. Kategori ini bisa dimiliki oleh Engimon manapun asalkan sesuai dengan elemen kompatibelnya. *Unique Skill* adalah kategori untuk *Skill* yang hanya bisa dimiliki oleh Engimon dengan spesies tertentu. Karena dimiliki sejak awal permainan, maka *Unique Skill* didesain dengan *base power* yang lebih rendah dari *skill* lainnya. Terakhir, *Special Skill* adalah *Skill* yang memiliki tambahan *base power* yang membuat *Skill* ini menjadi lebih kuat.

```
private enum EngimonLiarName
{
    Omnimon,
    Skull,
    Greymon,
    Piedmon,
    War_Greymon,
    MagnaAngemon,
    Garurumon,
    Devimon,
    Apocalypmon,
    Etemon,
    Agumon;

    /**
     * Pick a random value of the EngimonLiarName enum.
     * @return a random EngimonLiarName.
     */
    public static EngimonLiarName getRandomName() {
        Random random = new Random();
        return values()[random.nextInt(values().length)];
    }
}
```

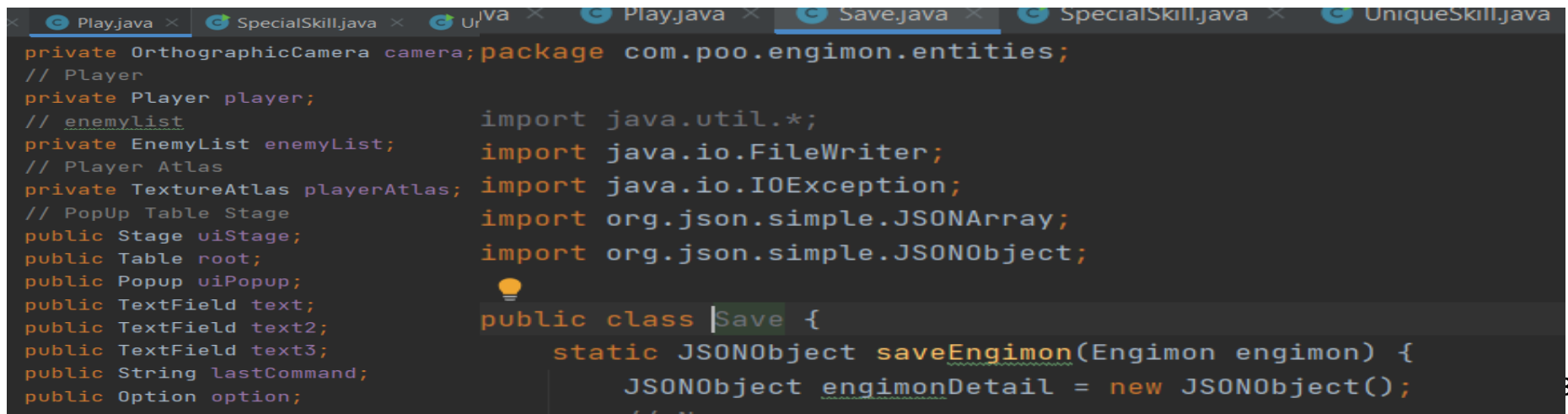
Wild Engimon akan dimasukkan dengan turn random dengan peluang tertentu menggunakan random generator. Selain itu, untuk penentuan species, nama, dan elemen, program akan generate posisi secara random juga. Apabila posisi yang dihasilkan berada pada area tertentu, maka hanya element yang eligible saja yang berada di tertentu. Untuk pemilihan spesies akan dilakukan secara random juga.

4. External Library

Pustaka eksternal yang digunakan untuk melakukan pembuatan GUI adalah *LibGDX*. LibGDX adalah sebuah pustaka yang memang khusus digunakan untuk pembuatan permainan, dan pemrogram bisa membuat program baik untuk *Desktop* maupun *Android*. Karena *LibGDX* mempunyai komponen permainan yang cukup banyak, maka *LibGDX* dipilih untuk mengimplementasikan GUI. Dengan menggunakan LibGDX, maka otomatis pemrogram juga menggunakan *library* LWJGL, karena LWJGL digunakan sebagai *backend* dari LibGDX.

Selain digunakan LibGDX, juga digunakan pustaka eksternal lain, yaitu Box2D dan Scene2D. Kedua *library* ini digunakan untuk menangani *User Interfaces* dan *Widgets* yang digunakan di dalam LibGDX. Beberapa komponen yang digunakan dari pustaka ini adalah *Stage*, *Text Field*, *Table*, *Skin*, *Texture*, dan *Texture Region* maupun *Drawable*. Komponen-komponen ini digunakan untuk menampilkan *output* ke permainan yang dibangun. Sayangnya, *LibGDX* belum mempunyai dokumentasi yang baik maupun tutorial yang memadai, karena kebanyakan tutorial yang ada sudah *outdated* dan *LibGDX* sendiri memiliki banyak fitur yang sudah mengalami perubahan-perubahan.

Untuk melakukan fungsionalitas *save* dan *load* program, akan digunakan pustaka eksternal berupa JSON *simple*. Dari *library* ini, digunakan dua buah kelas, yaitu JSON *Array* dan JSON *Object*. Kedua kelas ini berperan dalam melakukan *parsing* maupun konversi dari JSON ke dalam Java maupun dari Java ke dalam objek JSON. Dengan demikian, penyimpanan program dilakukan dengan menggunakan JSON.



```

package com.poo.engimon.entities;

import java.util.*;
import java.io.FileWriter;
import java.io.IOException;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

public class Save {
    static JSONObject saveEngimon(Engimon engimon) {
        JSONObject engimonDetail = new JSONObject();
    }
}

```

5. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
I.1. Engimon	13519185, 13519191, 13519180	13519185, 13519191, 13519180
I.2. Skill	13519185, 13519180	13519185, 13519180
I.3. Player	13519185, 13519191, 13519180, 13519217	13519185, 13519191, 13519180, 13519217
I.3.b Inventory	13519185, 13519180	13519185, 13519180
I.4. Battle	13519217, 13519185, 13519180	13519217, 13519185, 13519180
I.5. Breeding	13519191, 13519185	13519191, 13519185
I.6. Peta	13519185, 13519215	13519185, 13519215
II.1 Graphical User Interface	13519185, 13519180	13519185, 13519180
II.2 Save & Load Functionality	13519191	13519191
Bonus Kreasi Mandiri: <i>Player Movement</i> dan <i>Collision Detection</i> , Desain Map dan Pemain, <i>Skill</i>	13519185, 13519180	13519185, 13519180
Bonus Kreasi Mandiri: <i>Random Spawn Engimon</i>	13519215	13519215