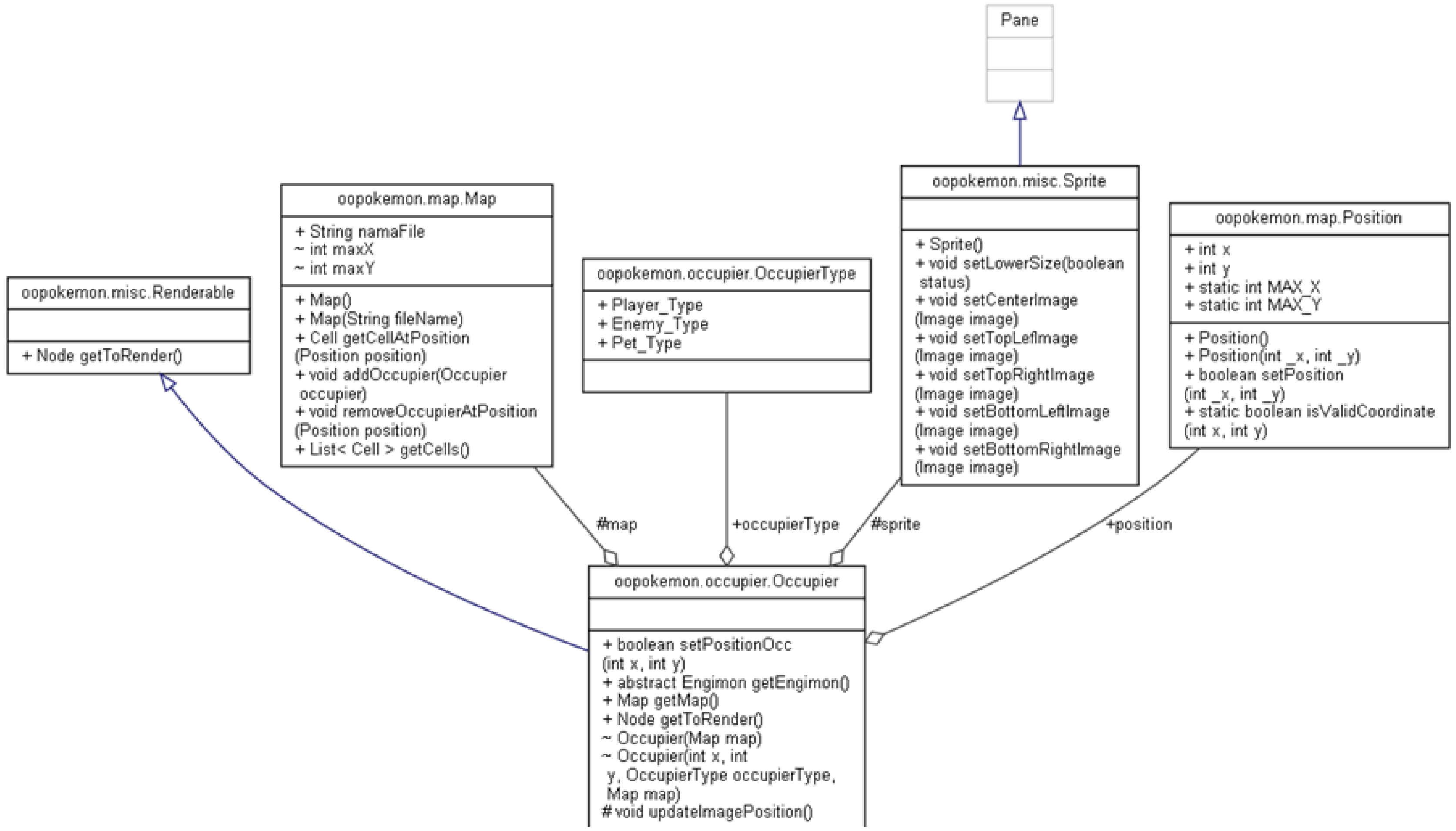
Kelas : 02
Nama Kelompok : OOPokemon

1. 13518014 / Ignatius David Partogi
2. 13519066 / Almeiza Arvin Muzaki
3. 13519075 / Juan Louis Rombetasik
4. 13519102 / Leonardus Brandon Luwianto
5. 13519107 / Daffa Ananda Pratama Resyaly
6. 13519109 / Christian Tobing Alexandro

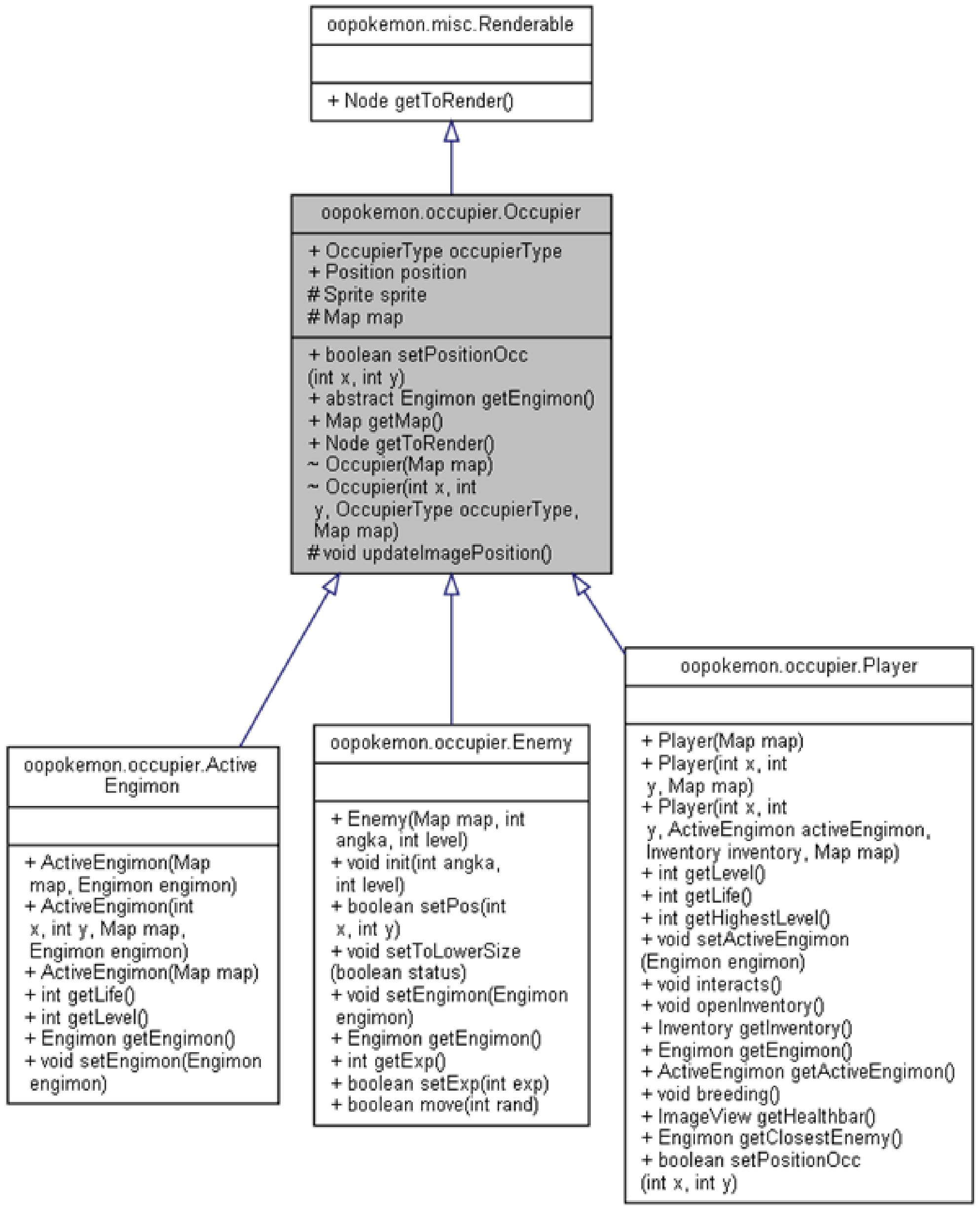
Asisten Pembimbing : Jan Meyer Saragih

## 1. Diagram Kelas

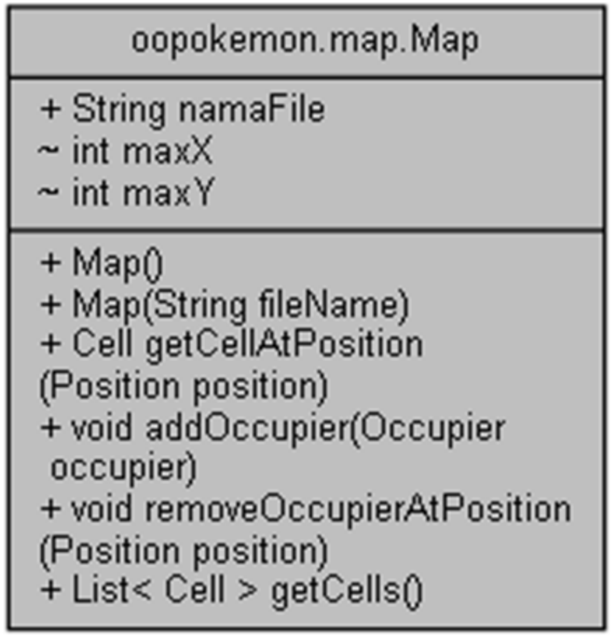
Occupier

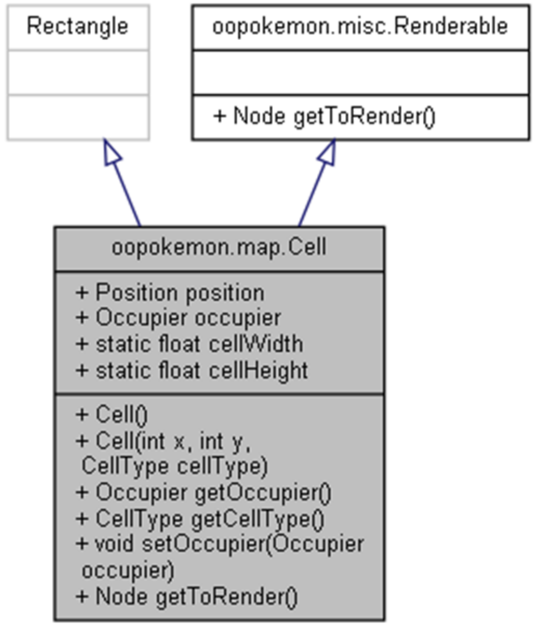

Pane
oopokemon.misc.Renderable
+ Node getToRender()
oopokemon.map.Map
+ String namaFile
~ int maxX
~ int maxY
+ Map()
+ Map(String fileName)
+ Cell getCellAtPosition (Position position)
+ void addOccupier(Occupier occupier)
+ void removeOccupierAtPosition (Position position)
+ List< Cell > getCells()
oopokemon.occupier.OccupierType
+ Player_Type
+ Enemy_Type
+ Pet_Type
oopokemon.misc.Sprite
+ Sprite()
+ void setLowerSize(boolean status)
+ void setCenterImage (Image image)
+ void setTopLeftImage (Image image)
+ void setTopRightImage (Image image)
+ void setBottomLeftImage (Image image)
+ void setBottomRightImage (Image image)
oopokemon.map.Position
+ int x
+ int y
+ static int MAX_X
+ static int MAX_Y
+ Position()
+ Position(int _x, int _y)
+ boolean setPosition (int _x, int _y)
+ static boolean isValidCoordinate (int x, int y)
#map
+occupierType
#sprite
+position
oopokemon.occupier.Occupier
+ boolean setPositionOcc (int x, int y)
+ abstract Engimon getEngimon()
+ Map getMap()
+ Node getToRender()
~ Occupier(Map map)
~ Occupier(int x, int y, OccupierType occupierType, Map map)
# void updateImagePosition()

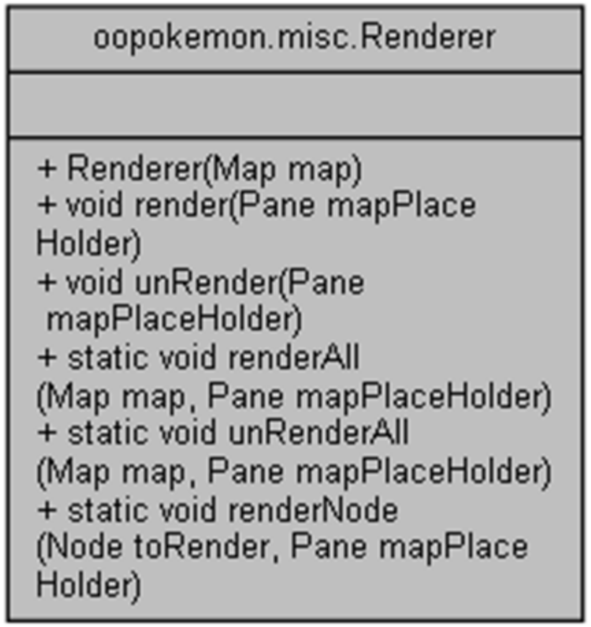
Turunan kelas Occupier


oopokemon.misc.Renderable
+ Node getToRender()
oopokemon.occupier.Occupier
+ OccupierType occupierType
+ Position position
# Sprite sprite
# Map map
+ boolean setPositionOcc (int x, int y)
+ abstract Engimon getEngimon()
+ Map getMap()
+ Node getToRender()
~ Occupier(Map map)
~ Occupier(int x, int y, OccupierType occupierType, Map map)
# void updateImagePosition()
oopokemon.occupier.ActiveEngimon
+ ActiveEngimon(Map map, Engimon engimon)
+ ActiveEngimon(int x, int y, Map map, Engimon engimon)
+ ActiveEngimon(Map map)
+ int getLife()
+ int getLevel()
+ Engimon getEngimon()
+ void setEngimon(Engimon engimon)
oopokemon.occupier.Enemy
+ Enemy(Map map, int angka, int level)
+ void init(int angka, int level)
+ boolean setPos(int x, int y)
+ void setToLowerSize (boolean status)
+ void setEngimon(Engimon engimon)
+ Engimon getEngimon()
+ int getExp()
+ boolean setExp(int exp)
+ boolean move(int rand)
oopokemon.occupier.Player
+ Player(Map map)
+ Player(int x, int y, Map map)
+ Player(int x, int y, ActiveEngimon activeEngimon, Inventory inventory, Map map)
+ int getLevel()
+ int getLife()
+ int getHighestLevel()
+ void setActiveEngimon (Engimon engimon)
+ void interacts()
+ void openInventory()
+ Inventory getInventory()
+ Engimon getEngimon()
+ ActiveEngimon getActiveEngimon()
+ void breeding()
+ ImageView getHealthbar()
+ Engimon getClosestEnemy()
+ boolean setPositionOcc (int x, int y)

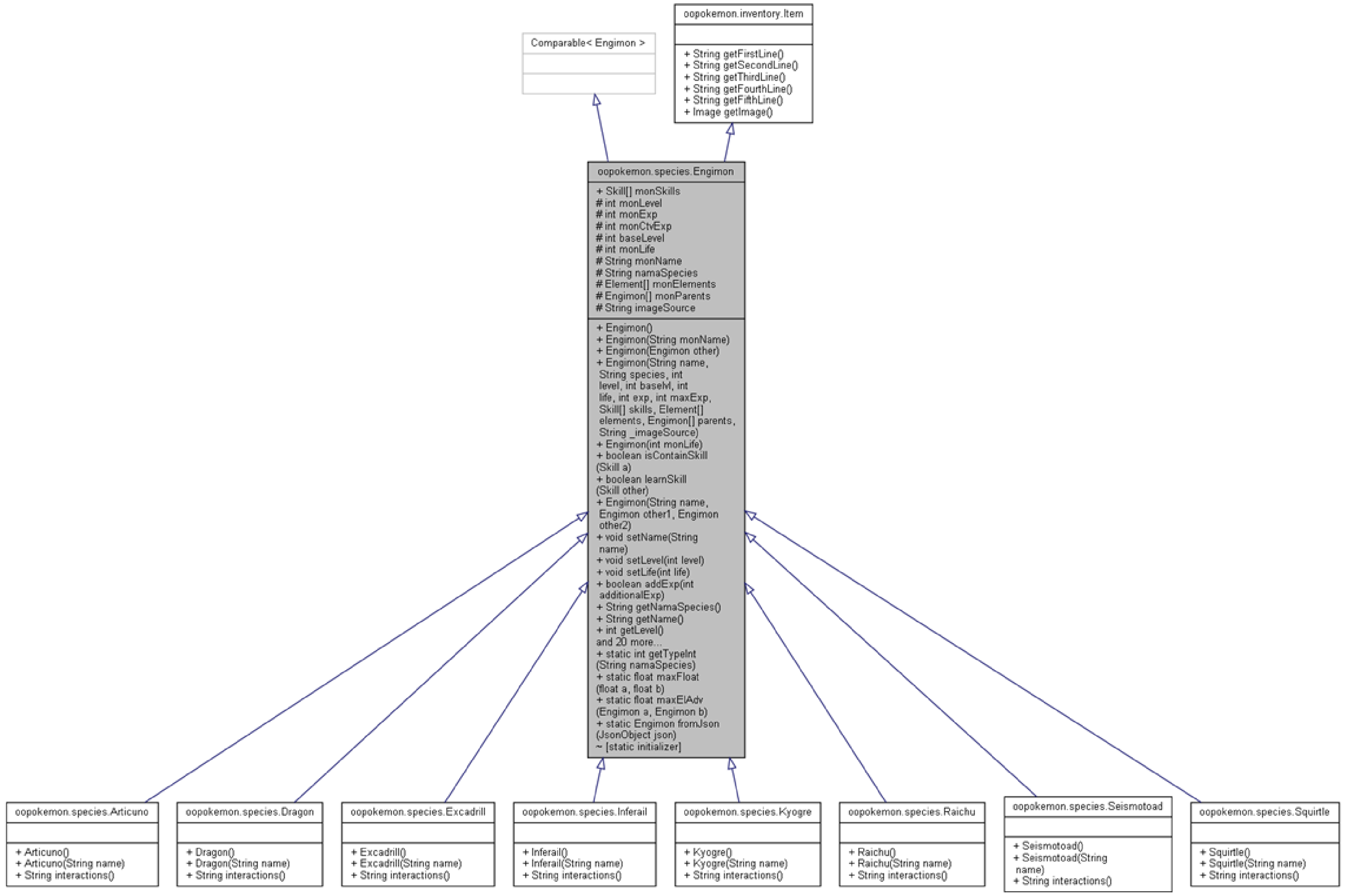
Kelas Map, Cell dan Renderer

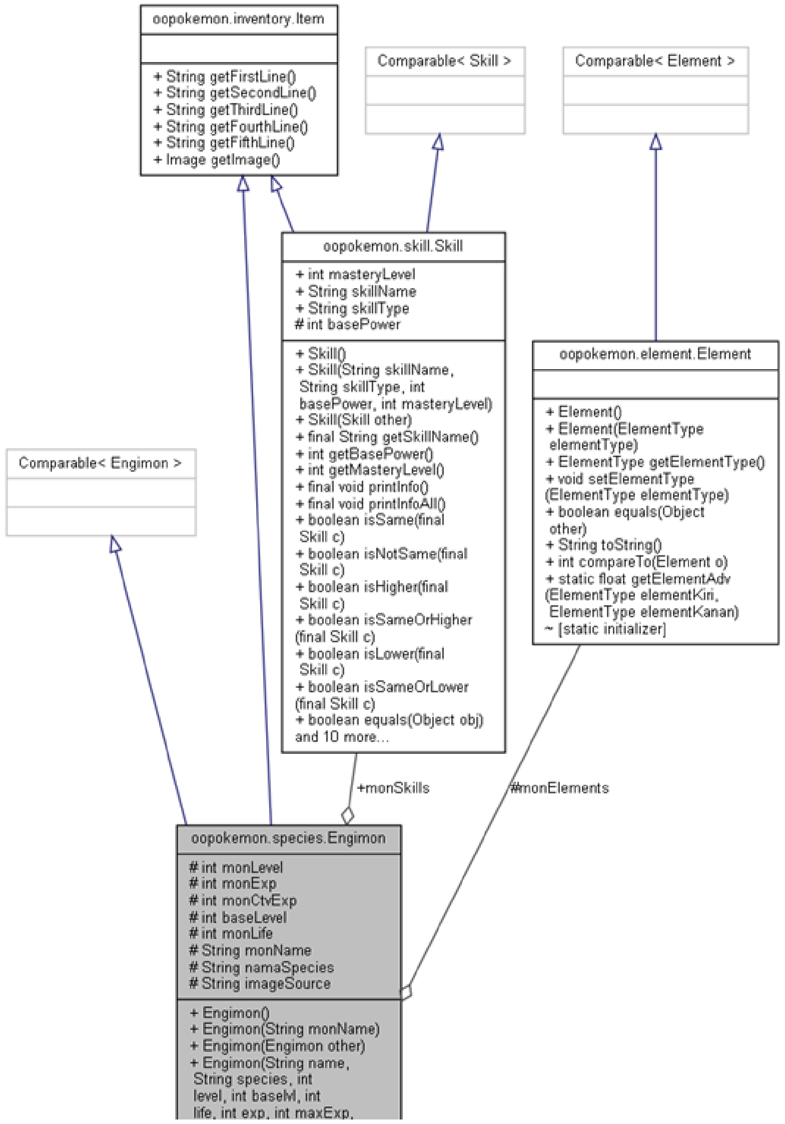

oopokemon.map.Map
+ String namaFile
~ int maxX
~ int maxY
+ Map()
+ Map(String fileName)
+ Cell getCellAtPosition (Position position)
+ void addOccupier(Occupier occupier)
+ void removeOccupierAtPosition (Position position)
+ List< Cell > getCells()



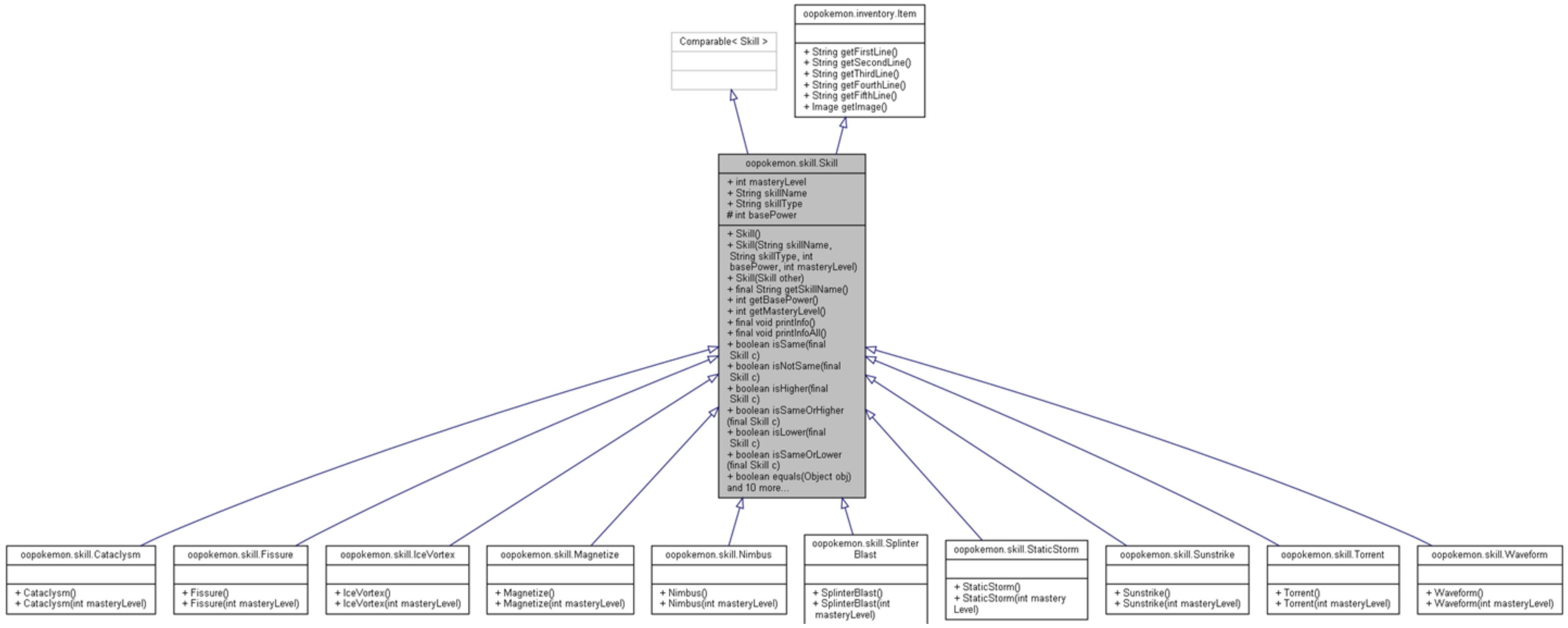

oopokemon.misc.Renderer
+ Renderer(Map map)
+ void render(Pane mapPlaceHolder)
+ void unRender(Pane mapPlaceHolder)
+ static void renderAll (Map map, Pane mapPlaceHolder)
+ static void unRenderAll (Map map, Pane mapPlaceHolder)
+ static void renderNode (Node toRender, Pane mapPlaceHolder)

## Kelas Engimon

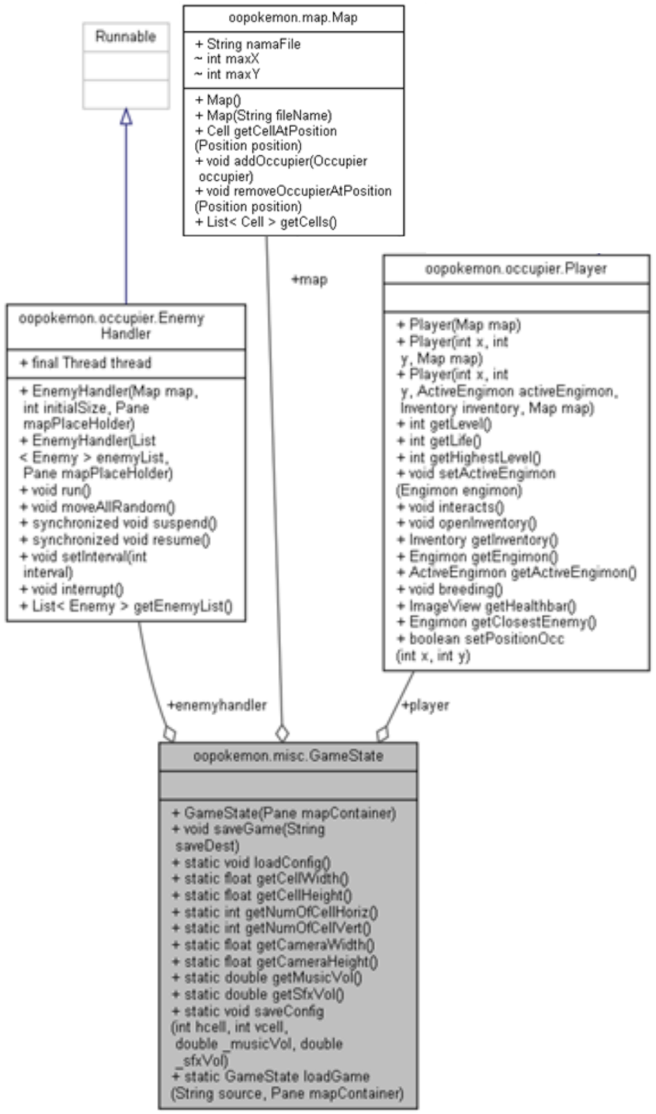



oopokemon.inventory.Item
+ String getFirstLine()
+ String getSecondLine()
+ String getThirdLine()
+ String getFourthLine()
+ String getFifthLine()
+ Image getImage()
Comparable< Skill >
Comparable< Element >
oopokemon.skill.Skill
+ int masteryLevel
+ String skillName
+ String skillType
# int basePower
+ Skill()
+ Skill(String skillName, String skillType, int basePower, int masteryLevel)
+ Skill(Skill other)
+ final String getSkillName()
+ int getBasePower()
+ int getMasteryLevel()
+ final void printInfo()
+ final void printInfoAll()
+ boolean isSame(final Skill c)
+ boolean isNotSame(final Skill c)
+ boolean isHigher(final Skill c)
+ boolean isSameOrHigher (final Skill c)
+ boolean isLower(final Skill c)
+ boolean isSameOrLower (final Skill c)
+ boolean equals(Object obj)
and 10 more...
oopokemon.element.Element
+ Element()
+ Element(ElementType elementType)
+ ElementType getElementType()
+ void setElementType (ElementType elementType)
+ boolean equals(Object other)
+ String toString()
+ int compareTo(Element o)
+ static float getElementAdv (ElementType elementKiri, ElementType elementKanan)
~ [static initializer]
Comparable< Engimon >
+monSkills
#monElements
oopokemon.species.Engimon
# int monLevel
# int monExp
# int monCtvExp
# int baseLevel
# int monLife
# String monName
# String namaSpecies
# String imageSource
+ Engimon()
+ Engimon(String monName)
+ Engimon(Engimon other)
+ Engimon(String name, String species, int level, int baseLvl, int life, int exp, int maxExp,

## Kelas Skill


Comparable< Skill >
oopokemon.inventory.Item
+ String getFirstLine()
+ String getSecondLine()
+ String getThirdLine()
+ String getFourthLine()
+ String getFifthLine()
+ Image getImage()
oopokemon.skill.Skill
+ int masteryLevel
+ String skillName
+ String skillType
# int basePower
+ Skill()
+ Skill(String skillName, String skillType, int basePower, int masteryLevel)
+ Skill(Skill other)
+ final String getSkillName()
+ int getBasePower()
+ int getMasteryLevel()
+ final void printInfo()
+ final void printInfoAll()
+ boolean isSame(final Skill c)
+ boolean isNotSame(final Skill c)
+ boolean isHigher(final Skill c)
+ boolean isSameOrHigher (final Skill c)
+ boolean isLower(final Skill c)
+ boolean isSameOrLower (final Skill c)
+ boolean equals(Object obj)
and 10 more...
oopokemon.skill.Cataclysm
+ Cataclysm()
+ Cataclysm(int masteryLevel)
oopokemon.skill.Fissure
+ Fissure()
+ Fissure(int masteryLevel)
oopokemon.skill.IceVortex
+ IceVortex()
+ IceVortex(int masteryLevel)
oopokemon.skill.Magnetize
+ Magnetize()
+ Magnetize(int masteryLevel)
oopokemon.skill.Nimbus
+ Nimbus()
+ Nimbus(int masteryLevel)
oopokemon.skill.Splinter Blast
+ SplinterBlast()
+ SplinterBlast(int masteryLevel)
oopokemon.skill.StaticStorm
+ StaticStorm()
+ StaticStorm(int mastery Level)
oopokemon.skill.Sunstrike
+ Sunstrike()
+ Sunstrike(int masteryLevel)
oopokemon.skill.Torrent
+ Torrent()
+ Torrent(int masteryLevel)
oopokemon.skill.Waveform
+ Waveform()
+ Waveform(int masteryLevel)

## Kelas GameState


Runnable
oopokemon.map.Map
+ String namaFile
~ int maxX
~ int maxY
+ Map()
+ Map(String fileName)
+ Cell getCellAtPosition (Position position)
+ void addOccupier(Occupier occupier)
+ void removeOccupierAtPosition (Position position)
+ List< Cell > getCells()
+map
oopokemon.occupier.Player
+ Player(Map map)
+ Player(int x, int y, Map map)
+ Player(int x, int y, ActiveEngimon activeEngimon, Inventory inventory, Map map)
+ int getLevel()
+ int getLife()
+ int getHighestLevel()
+ void setActiveEngimon (Engimon engimon)
+ void interacts()
+ void openInventory()
+ Inventory getInventory()
+ Engimon getEngimon()
+ ActiveEngimon getActiveEngimon()
+ void breeding()
+ ImageView getHealthbar()
+ Engimon getClosestEnemy()
+ boolean setPositionOcc (int x, int y)
oopokemon.occupier.Enemy Handler
+ final Thread thread
+ EnemyHandler(Map map, int initialSize, Pane mapPlaceHolder)
+ EnemyHandler(List < Enemy > enemyList, Pane mapPlaceHolder)
+ void run()
+ void moveAllRandom()
+ synchronized void suspend()
+ synchronized void resume()
+ void setInterval(int interval)
+ void interrupt()
+ List< Enemy > getEnemyList()
+enemyhandler
+player
oopokemon.misc.GameState
+ GameState(Pane mapContainer)
+ void saveGame(String saveDest)
+ static void loadConfig()
+ static float getCellWidth()
+ static float getCellHeight()
+ static int getNumOfCellHoriz()
+ static int getNumOfCellVert()
+ static float getCameraWidth()
+ static float getCameraHeight()
+ static double getMusicVol()
+ static double getSfxVol()
+ static void saveConfig (int hcell, int vcell, double _musicVol, double _sfxVol)
+ static GameState loadGame (String source, Pane mapContainer)

# 2. Penerapan Konsep OOP

## 2.1 Polymorphism

Konsep *polymorphism* digunakan pada kelas abstrak Occupier karena objek yang menempati cell dapat berupa Player, Wild Engimon atau Engimon sendiri.

```
public abstract class Occupier implements Renderable {
    protected Sprite sprite;
    public OccupierType occupierType;
    public Position position;
    protected Map map;

    private static final float cellHeight = Cell.cellHeight;
    private static final float cellWidth = Cell.cellWidth;

    Occupier(Map map) throws NotInitializedException {}

    Occupier(int x, int y, OccupierType occupierType, Map map) throws NotInitializedException {}

    protected void updateImagePosition() {}

    public boolean setPositionOcc(int x, int y) {}

    public Map getMap() {}

    @Override
    public Node getToRender() {}
}
```

## 2.2 Inheritance/Composition/Aggregation

### 2.2.1 Inheritance

Konsep pewarisan (*inheritance*) digunakan pada kelas Occupier, Engimon, dan Skill. Untuk *superclass* Occupier, terdapat tiga kelas turunannya yaitu *subclass* Player, ActiveEngimon, dan Enemy. Kelas Occupier juga merupakan *abstract class,* yang dijelaskan pada bagian 2.3 Abstract Class. Berikut cuplikan kode dari *subclass* Player yang mewarisi *subclass* Occupier:

```
public class Player extends Occupier {
    private final Inventory inventory = new Inventory();
    private final ActiveEngimon activeEngimon;

    public ImageView healthbar = new ImageView(new Image("assets/life3.png"));

    public Player(Map map) throws NotInitializedException {}

    ...
}
```

Untuk *superclass* Engimon, terdapat delapan kelas turunannya yaitu *subclass* Articuno, Dragon, Excadrill, Raichu, Squirtle, Inferail, Kyogre, dan Seismotoad. Kelas-kelas Species banyak memiliki kesamaan informasi seperti nama, nama parent, skill, level, experience — sehingga akan lebih baik jika kesamaan tersebut disimpan pada superclassnya, Engimon, dan diakses melalui inheritance. Adapun perbedaan informasi seperti tipe elemen akan diimplementasikan di kelas Spesies masing-masing. Berikut cuplikan kode dari *subclass* Squirtle yang mewarisi *subclass* Engimon:

```
public class Squirtle extends Engimon {
    private void InitComp() {
        namaSpecies = "Squirtle";
        monElements[0].setElementType(Water);
        monSkills[0] = new Torrent();
        imageSource = "assets/squirtle.png";
    }
    public Squirtle() {
        super();
        InitComp();
    }

    public Squirtle(String name) {
        super(name);
        InitComp();
    }
}
```

Untuk *superclass* Skill, terdapat sepuluh kelas turunannya yaitu *subclass* Cataclysm, Fissure, IceVortex, Magnetize, Nimbus, SplinterBlast, StaticStorm, Sunstrike, Torrent, dan Waveform. Hampir serupa seperti poin Engimon sebelumnya, kelas-kelas Skill memiliki atribut yang hampir serupa, hanya terdapat perbedaan atribut mengenai jenis spesies yang *eligible*. Oleh karena itu, lebih dirasa efisien apabila setiap Skill yang spesifik dibuat turunan dari *superclass* Skill. Berikut cuplikan kode dari *subclass* IceVortex yang mewarisi *subclass* Skill:

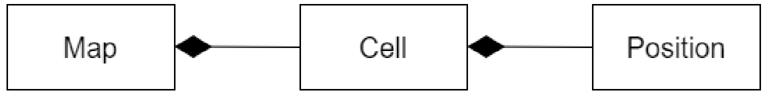
```
public class IceVortex extends Skill {
    private final String species;

    public IceVortex() {
        super("Ice Vortex", "Ice", 13, 1);
        this.species = "Articuno";
    }
    public IceVortex(String species, int masteryLevel) {
        super("Ice Vortex", "Ice", 13, masteryLevel);
        this.species = species;
    }
}
```

*Subclass* Cell juga mewarisi *superclass* Rectangle yang merupakan kelas bawaan JavaFX.

### 2.2.2 Composition

Composition merupakan *part-of relationship* sehingga sebuah entitas tidak mungkin terbentuk tanpa keberadaan entitas lain. Berikut salah satu contoh hubungan *composition* antara entitas Map, Cell, dan Position dalam notasi UML:



Jika Map di-*destroy*, Cell juga ikut di-*destroy*, akibatnya Position juga ikut di-*destroy*. Namun, Map tetap ada apabila Cell dan/atau Position dihapus.

```java
// Map.java
public class Map {
    private final List<Cell> cells;
    ...
    public Map() {
        namaFile = "";
        cells = new ArrayList<>();
        for (int i = 0; i < maxX * maxY; i++) {
            Cell cell = new Cell(i % maxX, i / maxX, CellType.Grassland_Cell);
            cells.add(cell);
        }
    }

    public Cell getCellAtPosition(Position position) {
        return cells.get(position.x + position.y * maxX);
    }

    ...
    public List<Cell> getCells(){
        return cells;
    }
}
```

```java
// Cell.java
public class Cell extends Rectangle implements Renderable {
    public Position position;
    ...

    public Cell() {
        super(0,0, cellWidth, cellHeight);
        position = new Position();
        ...
    }

    Cell(int x, int y, CellType cellType) {
        super(x * cellWidth, y * cellHeight, cellWidth, cellHeight);
        position = new Position(x,y);
        ...
    }
    ...
}
```

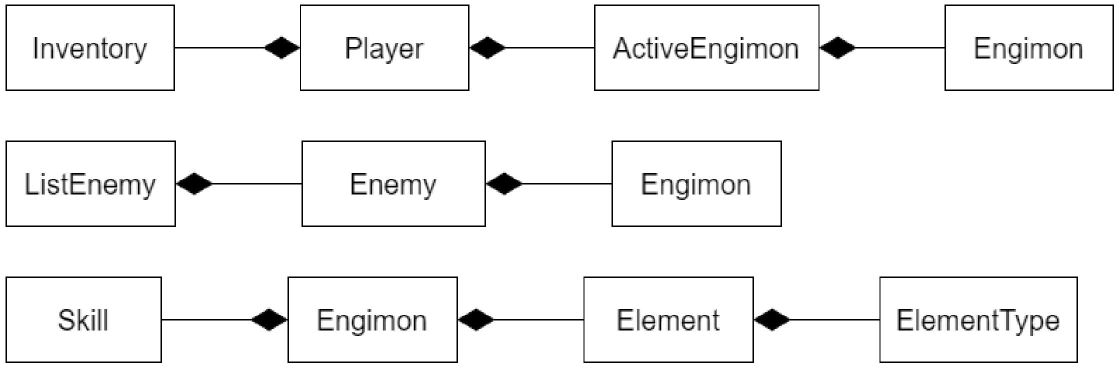
```java
// Position.java
public class Position {
    public int x;
    public int y;
    ...

    public Position() { x = 0; y = 0; }
    public Position(int _x, int _y) { x = _x; y = _y; }

    public boolean setPosition(int _x, int _y)
    {
        if (isValidCoordinate(_x, _y)){
            x = _x;
            y = _y;
            return true;
        }
        return false;
    }

    ...
}
```

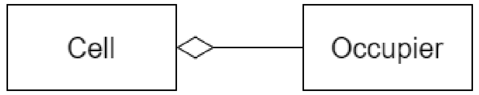
Hubungan *composition* pada entitas-entitas lain:



dan masih banyak lagi.

### 2.2.3 Aggregation

Aggregation merupakan *has-a relationship*. Jika suatu entitas dihapus, tidak akan memengaruhi entitas yang lain. Berikut salah satu contoh hubungan *aggregation* antara entitas Cell dan Occupier dalam notasi UML:



Cell bisa mempunyai Occupier, tetapi sebaliknya tidak bisa.

```
// Cell.java
public class Cell extends Rectangle implements Renderable {
    public Occupier occupier;

    ...

    public void setOccupier(Occupier occupier) {
        this.occupier = occupier;
    }

    ...
}
```

## 2.3 Abstract Class

*Abstract base class* digunakan pada kelas Occupier karena diturunkan menjadi kelas-kelas yang masih berhubungan satu sama lain. Kelas abstrak Occupier diturunkan menjadi kelas Player, ActiveEngimon, dan Enemy.

```
public abstract class Occupier implements Renderable {
    protected Sprite sprite;
```

```java
    public OccupierType occupierType;
    public Position position;
    protected Map map;

    private static final float cellHeight = Cell.cellHeight;
    private static final float cellWidth = Cell.cellWidth;

    Occupier(Map map) throws NotInitializedException {}

    Occupier(int x, int y, OccupierType occupierType, Map map) throws NotInitializedException {}

    protected void updateImagePosition() {}

    public boolean setPositionOcc(int x, int y) {}

    public Map getMap() {}

    public abstract Engimon getEngimon();

    @Override
    public Node getToRender() {}
}
```

## 2.4 Interface

Terdapat tiga *interface* yang kami buat yaitu Renderable, Exceptions, dan Item. *Interface* Renderable diimplementasikan oleh kelas Cell dan kelas Occupier yang membutuhkan *rendering*. Dapat dilihat pada cuplikan kode di bawah, kelas Cell juga merupakan turunan dari kelas Rectangle dan seperti yang diketahui Java tidak memperbolehkan *multiple inheritance.* Namun, sebuah kelas dapat mengimplementasikan *interface* lebih dari satu sehingga kelas Cell mengimplementasikan *interface* Renderable.

```java
// Renderable.java
```

```java
package oopokemon.misc;

import javafx.scene.Node;

public interface Renderable {
    Node getToRender();
}
```

```java
// Cell.java
import oopokemon.misc.Renderable;
import javafx.scene.Node;
...

public class Cell extends Rectangle implements Renderable {
    ...

    @Override
    public Node getToRender() {
        return this;
    }
}
```

Interface Item guna keperluan menampilkan informasi Item (Engimon dan Skill)

```java
// Item.Java
public interface Item {
    String getFirstLine();
    String getSecondLine();
    String getThirdLine();
    String getFourthLine();
    String getFifthLine();
    Image getImage();
}
```

Adapun *interface* yang kami gunakan merupakan bawaan dari Java API yaitu Comparable<T> dan Runnable. *Interface* Comparable<T> diimplementasikan oleh kelas Element, Skill, dan Engimon. Sebagai contoh, deklarasi *method* `compareTo` di *interface* Comparable<Engimon> diimplementasikan di kelas Engimon untuk keperluan *sorting* Engimon.

```
// Engimon.java
public class Engimon implements Comparable<Engimon> {

    ...

    @Override
    public int compareTo(Engimon o) {
        // kalau kedua element tidak sama, akan mensort elemennya
        if (!o.monElements[0].equals(this.monElements[0])) {
            return o.monElements[0].compareTo(this.monElements[0]);
        }
        // kalau kedua element sama tapi elemen kedua tidak sama akan mensort element kedua
        else if (!o.monElements[1].equals(this.monElements[1])) {
            return o.monElements[1].compareTo(this.monElements[1]);
        }
        // kedua element sama dan kedua element kedua sama
        return o.monLevel - this.monLevel;
    }
}
```

## 2.5 Generic Type & Wildcards

Berikut merupakah salah satu contoh penggunaan generic dan wildcard pada InventoryGUI yang menerima List turunan dari item yaitu bisa berupa List Engimon atau List Skill

```
// InventoryGUI.java
private static GridPane populateItem(List<? extends Item> items, Player player, Stage prevStage){
    GridPane gridPane = new GridPane();
    for (int i = 0; i < items.size(); i++) {
        InventroyItem inv = new InventroyItem(i, items.get(i), player, prevStage);
        gridPane.add(inv, i % 5, i / 5);
        }
    return gridPane;
}
```

## 2.6 Exception Handling

Blok *try* dan *catch* bersarang digunakan di kelas Map seperti ditunjukkan pada cuplikan kode ini:

```
   public Map(String fileName) {
        namaFile = fileName;
        cells = new ArrayList<>();
        try {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));
            String line = reader.readLine();
            maxX = (line != null)? line.length() : 0;
            Position.MAX_X = maxX;
            maxY = 0;
            while (line != null){
                for (int i = 0; i < maxX; i++) {
                    try {
                        char c = line.charAt(i);
                        CellType cellType;
                        switch (c){
                            case '^':
                                cellType = CellType.Mountain_Cell;
                                break;
                            case '=':
                                cellType = CellType.Tundra_Cell;
                                break;
                            case 'o':
                                cellType = CellType.Sea_Cell;
                                break;
                            default:
                                cellType = CellType.Grassland_Cell;
                        }
                        Cell cell = new Cell(i, maxY, cellType);
                        cells.add(cell);
                    }
                    catch (StringIndexOutOfBoundsException e){
                        Cell cell = new Cell(i, maxY, CellType.Grassland_Cell);
                        cells.add(cell);
                    }
                }
                maxY++;
                line = reader.readLine();
            }
            Position.MAX_Y = maxY;
        }
        catch (IOException e) {
            System.out.println("map tidak ditemukan");
        }
    }
```

Berikut cuplikan kode *exception handling* menggunakan kelas `Throwable` yang disediakan oleh Java.

```java
// NotInitializedException.java
package oopokemon.exception;

public class NotInitializedException extends Throwable implements Exceptions{

    public NotInitializedException() {
        super();
    }

    @Override
    public String getErrorMessage() {
        return "object not initialized";
    }
}
```

```java
...
    public static GameState loadGame(String source, Pane mapContainer) throws NotInitializedException {
        File input = new File("bin/savefiles/" + source +".json");
        try {
            JsonElement fileElement = JsonParser.parseReader(new FileReader(input));
            JsonObject fileObject = fileElement.getAsJsonObject();

            Map map;
            Player player;
            List<Enemy> enemyList = new ArrayList<>();
            EnemyHandler enemyHandler;

            // Mengambil data
            if (fileObject.has("map")) {
                String mapfile = fileObject.get("map").getAsString();
                map = new Map(mapfile);
            }
            else throw new NotInitializedException();
            if (fileObject.has("player")) {
```

```
        JsonObject playerObject = fileObject.get("player").getAsJsonObject();
        Position playerpos = new Gson().fromJson(playerObject.get("position").getAsJsonObject(), Position.class);

        ActiveEngimon activeEngimon = new ActiveEngimon(map);

        Inventory inventory = new Inventory();

        Engimon currentEngimon = null;

        if (fileObject.has("activeEngimon")){
            JsonObject aeObject = fileObject.get("activeEngimon").getAsJsonObject();
            Position aePos = new Gson().fromJson(aeObject.get("position").getAsJsonObject(), Position.class);
            activeEngimon.setPositionOcc(aePos.x, aePos.y);

            if (aeObject.has("engimon")) {
                JsonObject engimonObj = aeObject.get("engimon").getAsJsonObject();
                currentEngimon = Engimon.fromJson(engimonObj);
                inventory.addEngimon(currentEngimon);
            }
        }

        if (fileObject.has("inventory")){
            JsonObject invObj = fileObject.get("inventory").getAsJsonObject();
            if (invObj.has("engimonList")){
                JsonArray engimonArray = invObj.get("engimonList").getAsJsonArray();
                for (JsonElement engimonElement : engimonArray){
                    JsonObject engimonJSONObj = engimonElement.getAsJsonObject();
                    Engimon playerEngimon = Engimon.fromJson(engimonJSONObj);
                    inventory.addEngimon(playerEngimon);
                }
            }
            if (invObj.has("skillList")){
                JsonArray skillString = invObj.get("skillList").getAsJsonArray();
                Skill[] skills = new Gson().fromJson(skillString, Skill[].class);
                for (Skill skillItem : skills){
                    inventory.addSkill(skillItem);
                }
```

```
                    }
                }

                player = new Player(playerpos.x, playerpos.y, activeEngimon, inventory, map);
                player.setActiveEngimon(currentEngimon);
//                ActiveEngimon activeEngimon = new ActiveEngimon()

            }
            else throw new NotInitializedException();
            if (fileObject.has("enemies")) {
                JsonArray jsonArray = fileObject.getAsJsonArray("enemies");
                for (JsonElement enm : jsonArray) {
                    JsonObject enobj = enm.getAsJsonObject();
                    Position enemypos = new Gson().fromJson(enobj.get("position").getAsJsonObject(), Position.class);
                    int baselvl = enobj.get("level").getAsInt();
                    int jenis = Engimon.getTypeInt(enobj.get("type").getAsString());
                    int exp = enobj.get("exp").getAsInt();
                    Enemy enemy = new Enemy(map,jenis,baselvl);
                    enemy.setPos(enemypos.x, enemypos.y);
                    enemy.setExp(exp);
                    enemyList.add(enemy);
                }
                if (enemyList.size() > 0){
                    enemyHandler = new EnemyHandler(enemyList, mapContainer);
                }
                else throw new NotInitializedException();
            }
            else throw new NotInitializedException();
            System.out.println("Berhasil Load Game");
            return new GameState(map, player, enemyHandler, mapContainer);
        }
        catch (FileNotFoundException e) {
            throw new NotInitializedException();
        }
    }
...
```

Adapun kelas player yg mengimplementasi exception handling jika memasukkan input untuk memilih engimon yang tidak ada

```
// Player.java
...
do {
    input1 = InputBox.inputPrompt("Breeding", "Pilih Engimon 1");
    try {
        if (input1 == -1) {
            return;
        }
        engimon1 = listEngimon.get(input1);
    }
    catch (NullPointerException | IndexOutOfBoundsException e){
        engimon1 = null;
    }
}
while (input1 == null || engimon1 == null);
...
```

## 2.7 Java Collection

Berikut merupakan contoh penggunaan Collection di Java pada kode program yang dibuat.

```
// Element.java
import java.util.HashMap;

public class Element implements Comparable<Element> {
    private ElementType elementType;
    private static final HashMap<Tuple<ElementType, ElementType>, Float> tableElementAdv = new HashMap<>();
...
```

```java
// Element.java
...
import java.util.HashMap;

public class Element implements Comparable<Element> {
    private ElementType elementType;
    private static final HashMap<Tuple<ElementType, ElementType>, Float> tableElementAdv = new HashMap<>();
...
```

```java
// Inventory.java
...
import java.util.*;

import oopokemon.skill.*;
import oopokemon.species.*;

public class Inventory {
    public static final int MAX_CAPACITY = 6;

    private final Vector<Skill> skillBag = new Vector<>();
    private final Vector<Engimon> engimonBag = new Vector<>();
...
```

```java
// Bag.java
...
import java.util.*;

public class Bag<T> {
    public ArrayList<T> listItem;
    public int neff;
    public Bag() {
        this.listItem = new ArrayList<>(Inventory.MAX_CAPACITY);
...
```

```
// Map.java
...
import java.util.List;

public class Map {
    private final List<Cell> cells;
...
```

```
// Sorting skill (Engimon.java)
...
Collections.sort(temporaryskill, new Comparator<Skill>() {
    @Override
    public int compare(Skill o1, Skill o2) {
        return o1.getMasteryLevel()-o2.getMasteryLevel();
    }
});
...
```

## 2.8 Java API

Berikut merupakan contoh penggunaan Java Stream API pada kode program yang telah dibuat walaupun sebenarnya masih banyak lagi.

```
// Renderer.java
public class Renderer {
...
     /**
     * @param map is what to render
     * @param mapPlaceHolder is where to render
     */
    public static void renderAll(Map map, Pane mapPlaceHolder) {
        map.getCells()
                .forEach(cell -> {
                    mapPlaceHolder.getChildren().add(cell.getToRender());
                });
        map.getCells()
                .stream()
                .filter(cell -> cell.occupier != null)
                .map(cell -> cell.occupier.getToRender())
                .forEach(node -> mapPlaceHolder.getChildren().add(node));
    }

...

}
```

```
// GameState.java (saveGame)
```

```
...
        List<Enemy> enemyList = map.getCells().stream()
                .filter(cell -> cell.occupier != null && cell.occupier.occupierType == OccupierType.Enemy_Type)
                .map(cell -> (Enemy) cell.occupier).collect(Collectors.toList());
...
```

```
// Inventory.java
...
public class Inventory {

...

    public int getHighestLevel(){
        if (this.engimonBag.isEmpty()) return 1;
        return engimonBag
                .stream()
                .map(Engimon::getLevel)
                .max(Comparator.naturalOrder())
                .get();
    }
}
```

## 3. Bonus Yang dikerjakan

### 3.1 Bonus yang diusulkan oleh spek

#### 3.1.1 Multi-threading (real-time gameplay)

Membuat kelas yang mengimplementasi Runnable dengan atribut thread dirinya sendiri, setiap kali memulai permainan, EnemyHandler akan langsung memulai thread dan menggerakkan semua WildEngimon secara random. Pada awalnya kami menggunakan satu boolean untuk EnemyHandler yaitu wantToSusp, namun terdapat masalah yaitu thread tidak terbunuh pada saat keluar dari aplikasi (gc tidak mengambil thread dikarenakan thread masih menunggu notify) untuk mengatasinya kami menggunakan dua boolean yaitu dengan tambahan boolean running untuk untuk menginterrupt dan memberhentikan thread supaya di koleksi oleh garbage collection

Berikut adalah implementasinya (tidak termasuk logika-logika gameplay)

```java
...
public class EnemyHandler implements Runnable {
    private final List<Enemy> enemyList;
    private final AtomicBoolean running = new AtomicBoolean(false);
    private final AtomicBoolean wantToSusp = new AtomicBoolean(false);
    private final Thread thread;

    private int interval = 500;

    public EnemyHandler(Map map, int size) throws NotInitializedException {
        this.enemyList = new ArrayList<>();
        for (int i = 0; i < size; i++) {
            Random rand = new Random();
            enemyList.add(new Enemy(map, rand.nextInt(8),1));
        }
        thread = new Thread(this);
        thread.start();
    }
    public EnemyHandler(List<Enemy> enemyList) {
        this.enemyList = enemyList;
        thread = new Thread(this);
        thread.start();
    }

    @Override
    public void run() {
        moveAllRandom();
    }


    public void moveAllRandom() {
        running.set(true);
        while (running.get()) {
            try {
                Thread.sleep(interval);
                synchronized (this){
                    while (wantToSusp.get()){
                        wait();
                    }
                    enemyList.forEach(enemy -> enemy.move(new Random().nextInt(4)));
                }

            }
            catch (InterruptedException ignored) {}
        }
    }
    public synchronized void suspend() {
        wantToSusp.set(true);
    }

    public synchronized void resume(){
        wantToSusp.set(false);
        notify();
    }
    public void setInterval(int interval) {
        if (interval >= 1000){
            this.interval = interval;
        }
    }

    public void interrupt(){
        this.running.set(false);
        this.thread.interrupt();
    }
}
```

```
 // File OOPokemonApp.java
...
    @Override
    public void stop() throws Exception {
        super.stop();
        if (this.musicPlayer != null){
            musicPlayer.interrupt();
        }
        if (this.enemyHandler != null){
            enemyHandler.interrupt();
        }
    }
...
```

### 3.1.2 Unit Testing Implementation

Unit Testing merupakan suatu prosedur *testing* pada perangkat lunak yang telah dibuat untuk menguji masing-masing komponen/bagian dari perangkat lunak. Berikut adalah contoh dari beberapa implementasi Unit Testing yang telah dibuat pada program ini.

```
// File CellTest.java
```

```java
class CellTest {
    private Cell c1;
    private Cell c2;
    private Cell c3;
    private Occupier o1;
    private Occupier o2;

    @BeforeEach
    void setUp() {
        c1 = new Cell();
        c2 = new Cell(16, 23, Grassland_Cell);
        c3 = new Cell(6, 15, Tundra_Cell);
    }

    @Test
    void getCellType() {
        assertEquals(Grassland_Cell, c1.getCellType());
        assertEquals(Grassland_Cell, c2.getCellType());
        assertEquals(Tundra_Cell, c3.getCellType());
    }

    @Test
    void setOccupier() {
        c1.setOccupier(o2);
        c3.setOccupier(o1);
        assertEquals(o2, c1.getOccupier());
        assertNull(c2.getOccupier());
        assertEquals(o1, c3.getOccupier());
    }
}
```

```java
// File ElementTest.java
class ElementTest {
    private static Element e1;
    private static Element e2;
    private static Element e3;

    @BeforeEach
    void setUp() {
        e1 = new Element();
        e2 = new Element(Fire);
        e3 = new Element(Ice);
    }

    @Test
    void getElementType() {
        assertAll("Element Type",
            () -> assertEquals(None, e1.getElementType()),
            () -> assertEquals(Fire, e2.getElementType())
        );
    }

    @Test
    void getElementAdv() {
        assertAll("Element Advantage",
                () -> assertEquals(-5, Element.getElementAdv(e1.getElementType(), e2.getElementType())),
                () -> assertEquals(2, Element.getElementAdv(e2.getElementType(), e3.getElementType())),
                () -> assertEquals(-5, Element.getElementAdv(e3.getElementType(), e1.getElementType()))

        );
    }
...
```

```java
// File EngimonTest.java
class EngimonTest {
    private Engimon e1;
    private Articuno e2;
    private Inferail e3;
    private Engimon e4;
    private Seismotoad e5;
    private Skill s1;
    private Sunstrike s2;
    private Nimbus s3;
    private StaticStorm s4;
    private SplinterBlast s5;
    private IceVortex s6;

    @BeforeEach
    void setUp() {
        e1 = new Engimon();
        e2 = new Articuno("Artik");
        e3 = new Inferail("Inferred");
        e4 = new Engimon(e1);
        e5 = new Seismotoad("Toad");
        s1 = new Skill();
        s2 = new Sunstrike();
        s3 = new Nimbus();
        s4 = new StaticStorm();
        s5 = new SplinterBlast();
        s6 = new IceVortex();
    }

    @Test
    void isContainSkill() {
        assertAll("Engimon contains Skill",
                () -> assertTrue(e1.isContainSkill(s1)),
                () -> assertTrue(e4.isContainSkill(s1)),
                () -> assertTrue(e2.isContainSkill(s6)),
                () -> assertTrue(e3.isContainSkill(s2)),
                () -> assertTrue(e3.isContainSkill(s4)),
                () -> assertTrue(e2.isContainSkill(s1)),
                () -> assertFalse(e4.isContainSkill(s3)),
                () -> assertFalse(e1.isContainSkill(s2)),
                () -> assertFalse(e2.isContainSkill(s2)),
                () -> assertFalse(e2.isContainSkill(s5)),
                () -> assertFalse(e3.isContainSkill(s3))
        );
    }

    @Test
    void learnSkill() {
        assertAll("Engimon learns Skill",
                () -> assertTrue(e2.learnSkill(s5)),
                () -> assertTrue(e3.learnSkill(s3)),
                () -> assertFalse(e3.learnSkill(s2)),
                () -> assertFalse(e1.learnSkill(s2)),
                () -> assertFalse(e4.learnSkill(s3)),
                () -> assertFalse(e2.learnSkill(s2)),
                () -> assertFalse(e3.learnSkill(s2)),
                () -> assertFalse(e2.learnSkill(s6))
        );
    }
...
```

## 3.2 Bonus Kreasi Mandiri

### Camera System

Membuat efek seperti kamera (Memfokuskan scene pada player) dengan menggeser Pane atau layout sejauh offset layar

```
private void cameraHandler() {
    mapContainer.setTranslateX(-getCameraWidth() * (myPlayer.position.x / getNumOfCellHoriz()));
    mapContainer.setTranslateY(-getCameraHeight() * (myPlayer.position.y / getNumOfCellVert()));
}
```

## MusicPlayer

Music player adalah seperti namanya adalah music player, dengan memanfaatkan kelas MediaPlayer javaFX

```
public class MusicPlayer {

    private final MediaPlayer mediaPlayer;

    /**
     * @param namaFile is the location where the music file is located relative to the project folder
     * @param musicType is the type of music
     * @param loop true for loop, false for play only once
     */
    public MusicPlayer(String namaFile, MusicType musicType, boolean loop){
        mediaPlayer = new MediaPlayer(new Media(Paths.get(namaFile).toUri().toString()));
        if (loop) {
            mediaPlayer.setOnEndOfMedia(() -> mediaPlayer.seek(Duration.ZERO));
        }
        mediaPlayer.setVolume(musicType.volume);
    }

    public enum MusicType {
        BGM(0.5),
        SFX(0.1);
        private double volume;

        MusicType(double volume){
            this.volume = volume;
        }

        public void setVolume(double volume) {
            this.volume = volume;
        }
    }
    public void pause(){
        mediaPlayer.pause();
    }
    public void play(){
        mediaPlayer.play();
    }
    public void interrupt() {
        mediaPlayer.stop();
    }
}
```

Implementasi MusicPlayer adalah pada saat dengan menjalankan background music pada saat memulai permainan dan memainkan sound effect sesuai tipe cell yang di lalui

```
// File OOPokemon.java (Memainkan BGM)
private void playBGM() {
    musicPlayer = new MusicPlayer("bin/music/Anville Town.mp3", MusicPlayer.MusicType.BGM, true);
    musicPlayer.play();
}
```

```
// File Player.java (Memainkan SFX)
String namaFile = map.getCellAtPosition(this.position).getCellType().getClip();
MusicPlayer musicPlayer = new MusicPlayer(namaFile, MusicPlayer.MusicType.SFX,false);
musicPlayer.play();
```

**Kustomisasi ukuran Cell, Banyaknya Cell horizontal/vertical, Volume Music**

```
// File GameState.java (Mengatur Konfigurasi File)
private static class Config{
    public final float cellWidth = getCellWidth();
    public final float cellHeight = getCellHeight();
    public final int numOfCellHoriz = getNumOfCellHoriz();
    public final int numOfCellVert = getNumOfCellVert();
    public final double musicVol = getMusicVol();
    public final double sfxVol = getSfxVol();
    Config(){}
}

public static void saveConfig(int hcell, int vcell, double _musicVol, double _sfxVol) {
    numOfCellHoriz = hcell;
    numOfCellVert = vcell;
    musicVol = _musicVol;
    sfxVol = _sfxVol;

    Gson gson = new GsonBuilder().setPrettyPrinting().create();
    String json = gson.toJson(new GameState.Config());

    try {
        FileWriter myWriter = new FileWriter("bin/config.json");
        myWriter.write(json);
        myWriter.close();
        System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}

// File config.json
{
  "cellWidth": 200.0,
  "cellHeight": 200.0,
  "numOfCellHoriz": 10,
  "numOfCellVert": 5,
  "musicVol": 0.15,
  "sfxVol": 0.5
}
```

## 4. External Library

**Google GSON 2.8.6**

Alasan menggunakan GSON adalah untuk mempermudah serialisasi dan deserialisasi object (save dan load).

**JavaFX** (Java 10 ke bawah masih di include dalam oracleJDK)

Alasan menggunakan JavaFX adalah karena lebih modern, styling JavaFX juga bisa menggunakan CSS sehingga mempermudah desain

Pembuatan GUI juga bisa dilakukan dengan scene builder dengan FXML walaupun kami tidak memakainya. Selain JavaFX menyediakan kelas media guna pemutaran musik dan sebagainya

Unit Test :

**JUnit 5.4.2**

Alasan penggunaan JUnit adalah sederhana, mudah digunakan, dan memang dibuat khusus untuk *unit testing* (telah tersedia *library* untuk *assertion*). Selain itu, JUnit juga langsung menampilkan hasil dari *unit testing* yang telah dilakukan.

## 5. Pembagian Tugas

| Modul (dalam poin spek) | Designer | Implementer |
|---|---|---|
| I. Engimon | 13519107 | 13519107 |
| I.1. Species | 13519107 | 13519107 |
| II. Skill | 13518014 | 13518014 |
| III. Player | 13519075<br>13519102 | 13519075<br>13519102 |
| III.1. Active Engimon | 13519075<br>13519102 | 13519102 |
| IV. Battle | 13519066<br>13519102 | 13519066<br>13519075<br>13519102 |
| IV.1. Enemy | 13519109 | 13519075<br>13519109 |
| V. Breeding | 13519066 | 13519066<br>13519075 |
| VI. Peta | 13519075<br>13519109 | 13519075<br>13519102 |
| VII. Graphical User Interface | 13519075<br>13519107 | 13518014<br>13519066<br>13519075<br>13519102<br>13519107<br>13519109 |

| | | |
|---|---|---|
| VIII. Save and Load Functionality | 13519075 | 13518014<br>13519075<br>13519107 |
| Bonus Multithreading | 13519075<br>13519109 | 13519075<br>13519109 |
| Bonus Unit Testing Implementation | 13519107 | 13519107 |
| Bonus Kreasi Mandiri | 13519075 | 13519075 |