Kelas : 01
Nama Kelompok : keluarga cemara
1. 13519005 / Yusuf Alwansyah
2. 13519028 / Hafid Abi Daniswara
3. 13519035 / Fakhri Nail Wibowo
4. 13519037 / Arsa Daris Gintara
5. 13519044 / Kinantan Arya Bagaspati
6. 13519052 / Syamil Cholid Abdurrasyid
Asisten Pembimbing : Muhammad Fariz Luthfan Wakan

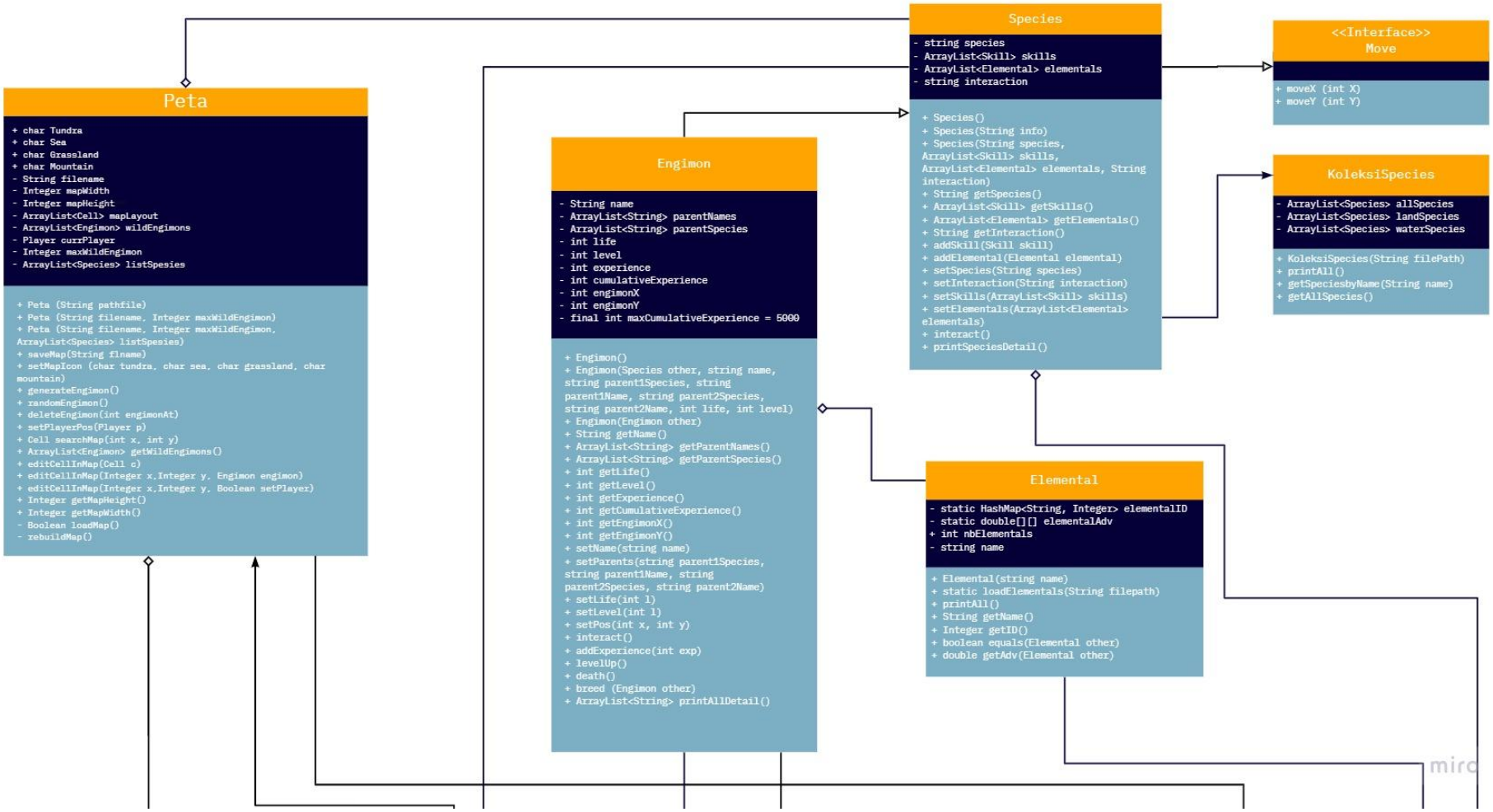
1. Diagram Kelas

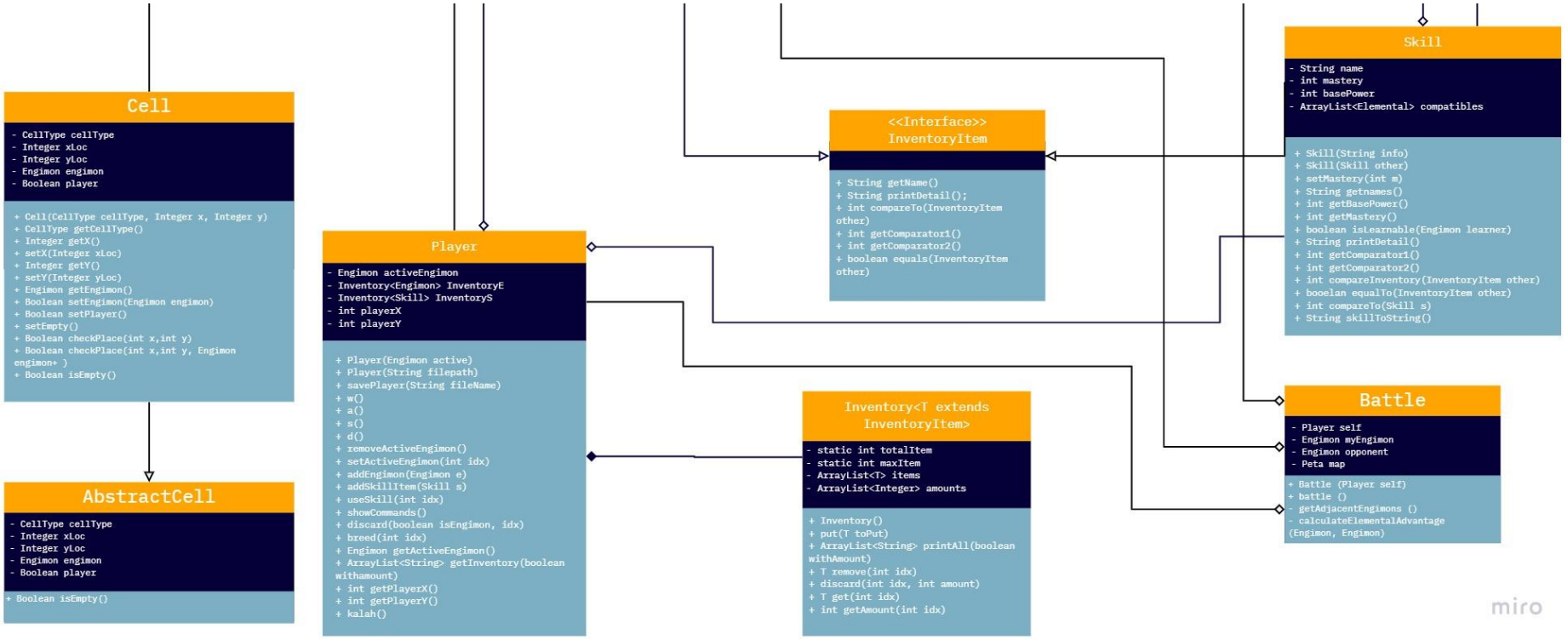
Program diinisialisasi pada kelas untuk GUI di StartPage dan GamePage, dan semua aksi dan proses diatur melalui kedua kelas itu. Dalam diagram kelas tidak ditampilkan karena bukan bagian dari desain. Program yang kami buat flow utamanya didasarkan ke kelas Player karena kami rasa semua command paling cocok berada pada kelas Player. Tetapi, secara flow kelas Player dan Maps berjalan bersama. Oleh karena itu, pada diagram di bawah kelas player memiliki asosiasi dengan kelas maps. Untuk membantu menyimpan lokasi pada peta, menggunakan kelas baru bernama Cell yang berisi koordinat x dan y. Dalam peta disimpan semua objek yang sedang berada di map.

Spesies merupakan parent class dari Engimon, desain ini dipakai karena setiap Spesies memiliki kemiripan yang diturunkan pada masing-masing Engimon. Spesies diambil dari kelas KoleksiSpesies yang isinya diambil dari file eksternal, hal ini dilakukan agar tidak perlu menentukan spesies-spesies yang ada secara manual dan hardcode. Pada Engimon ada hubungan komposisi dengan Elemental, hal ini karena tidak setiap Engimon yang memiliki Spesies sama akan memiliki jumlah atau elemen yang sama.

Penerapan Inventory menggunakan Generic dan Interface karena ada dua objek yang akan disimpan, Engimon dan Skill. Generic dipakai agar kedua objek tersebut dapat disimpan dalam objek yang sama, Interface dipakai agar kedua objek memiliki method yang sama untuk bisa berinteraksi dengan Inventory.

Penerapan Battle dipisahkan menjadi kelas sendiri yang akan dipanggil oleh kelas Player ketika akan melakukan battle, hal ini dipilih agar proses Battle terenkapsulasi dan lebih mudah untuk diuji. Battle mempunyai atribut Player, Peta dan Engimon milik Player dan Engimon yang akan menjadi musuh.


Peta
+ char Tundra
+ char Sea
+ char Grassland
+ char Mountain
- String filename
- Integer mapWidth
- Integer mapHeight
- ArrayList<Cell> mapLayout
- ArrayList<Engimon> wildEngimons
- Player currPlayer
- Integer maxWildEngimon
- ArrayList<Species> listSpesies
+ Peta (String pathfile)
+ Peta (String filename, Integer maxWildEngimon)
+ Peta (String filename, Integer maxWildEngimon, ArrayList<Species> listSpesies)
+ saveMap(String flname)
+ setMapIcon (char tundra, char sea, char grassland, char mountain)
+ generateEngimon()
+ randomEngimon()
+ deleteEngimon(int engimonAt)
+ setPlayerPos(Player p)
+ Cell searchMap(int x, int y)
+ ArrayList<Engimon> getWildEngimons()
+ editCellInMap(Cell c)
+ editCellInMap(Integer x,Integer y, Engimon engimon)
+ editCellInMap(Integer x,Integer y, Boolean setPlayer)
+ Integer getMapHeight()
+ Integer getMapWidth()
- Boolean loadMap()
- rebuildMap()
Species
- string species
- ArrayList<Skill> skills
- ArrayList<Elemental> elementals
- string interaction
+ Species()
+ Species(String info)
+ Species(String species, ArrayList<Skill> skills, ArrayList<Elemental> elementals, String interaction)
+ String getSpecies()
+ ArrayList<Skill> getSkills()
+ ArrayList<Elemental> getElementals()
+ String getInteraction()
+ addSkill(Skill skill)
+ addElemental(Elemental elemental)
+ setSpecies(String species)
+ setInteraction(String interaction)
+ setSkills(ArrayList<Skill> skills)
+ setElementals(ArrayList<Elemental> elementals)
+ interact()
+ printSpeciesDetail()
<<Interface>>
Move
+ moveX (int X)
+ moveY (int Y)
KoleksiSpecies
- ArrayList<Species> allSpecies
- ArrayList<Species> landSpecies
- ArrayList<Species> waterSpecies
+ KoleksiSpecies(String filePath)
+ printAll()
+ getSpeciesbyName(String name)
+ getAllSpecies()
Engimon
- String name
- ArrayList<String> parentNames
- ArrayList<String> parentSpecies
- int life
- int level
- int experience
- int cumulativeExperience
- int engimonX
- int engimonY
- final int maxCumulativeExperience = 5000
+ Engimon()
+ Engimon(Species other, string name, string parent1Species, string parent1Name, string parent2Species, string parent2Name, int life, int level)
+ Engimon(Engimon other)
+ String getName()
+ ArrayList<String> getParentNames()
+ ArrayList<String> getParentSpecies()
+ int getLife()
+ int getLevel()
+ int getExperience()
+ int getCumulativeExperience()
+ int getEngimonX()
+ int getEngimonY()
+ setName(string name)
+ setParents(string parent1Species, string parent1Name, string parent2Species, string parent2Name)
+ setLife(int l)
+ setLevel(int l)
+ setPos(int x, int y)
+ interact()
+ addExperience(int exp)
+ levelUp()
+ death()
+ breed (Engimon other)
+ ArrayList<String> printAllDetail()
Elemental
- static HashMap<String, Integer> elementalID
- static double[][] elementalAdv
+ int nbElementals
- string name
+ Elemental(string name)
+ static loadElementals(String filepath)
+ printAll()
+ String getName()
+ Integer getID()
+ boolean equals(Elemental other)
+ double getAdv(Elemental other)
miro



2. Penerapan Konsep OOP

2.1. Polymorphism

Species.java

```
106     //Method lain
107     public void interact(){
108         System.out.println(getInteraction());
109     }
```

Engimon.java

```
117     //Method lain
118     public void interact(){
119       System.out.println(this.name + ": " + getInteraction());
120     }
```

Pada kelas Species dan Engimon menggunakan konsep polymorphism dimana pada kelas Engimon method interact memiliki aksi yang lain berupa tambahan atribut name pada interaksinya. Konsep ini dipakai agar kelas turunan memiliki method yang lebih relevan, dimana pada kasus ini interact menjadi lebih relevan jika diberi tambahan nama dari species.

2.2. Inheritance/Composition/Aggregation

```
public class Battle {
    private Player self;
    private Engimon myEngimon;
    private Engimon opponent;
    private Peta map;
```

Pada kelas Battle menggunakan konsep Aggregation dimana kelas Player, Engimon dan Map merupakan bagian dari Battle, tetapi lifecycle mereka tidak selalu bergantung pada kelas Battle. Konsep ini dipakai agar proses battle bisa terenkapsulasi dan lebih mudah untuk diuji daripada menjadikan battle sebagai suatu method sendiri.

```
public class Player {
    private Inventory<Engimon> InventoryE;
    private Inventory<Skill> InventoryS;
    private int playerX, playerY;
    private Engimon activeEngimon;
```

Pada kelas Player terdapat penggunaan konsep Composition, yaitu kelas Inventory adalah hal penyusun dari kelas Player. Setiap kelas Player pasti memiliki Inventory dan kedua kelas tersebut memiliki lifecycle yang sama. Penggunaan konsep ini menyesuaikan spesifikasi tugas dan konsep dari game itu sendiri dimana Player itu memiliki Inventory.

2.3. Abstract Class

```
abstract class AbstractCell {
    private CellType cellType;
    private Integer xLoc;
    private Integer yLoc;
    private Engimon engimon;
    private Boolean player;

    public abstract Boolean isEmpty();
}
```

Menggunakan kelas abstrak pada penerapan kelas Cell, kelas Cell dibuat untuk memudahkan dalam membuat kelas Peta. Kelas Cell akan menginherit dari kelas abstrak AbstractCell yang berisi properti dan satu method abstrak.

2.4. Interface

a. Move Interface

Move interface berfungsi sebagai interface yang harus diterapkan untuk memindahkan baik player, engimon dari sumbu x dan sumbu Y. berisi dua method, yaitu: moveX dan moveY. Pada kasus ini kami menggunakan interface karena method ini banyak dipakai dan berkaitan antara satu dengan yang lain, namun tidak memiliki hubungan Is-a dengan class - class pemakainya, sehingga kami menggunakan interface.

```
public interface Move{
  public void moveX(int x);

  public void moveY(int y);
}
```

b. InventoryItem Interface

Pada Inventory terdapat dua objek yang bisa disimpan, berupa Engimon dan Skill. Oleh karena itu, untuk mempermudah membuat method agar kedua objek tersebut disimpan maka dibuat Interface yang berisi method yang diperlukan.

```
interface InventoryItem {

    public String getName();
    public String printDetail();
    public int compareInventory(InventoryItem other);
    public int getComparator1();
    public int getComparator2();
    public boolean equals(InventoryItem other);
}
```

2.5. Generic Type & Wildcards

Penggunaan Generic disini digunakan pada class Inventory

```
public class Inventory <T extends InventoryItem> {
    private static int totalItem;
    private static int maxItem = 30;
    private List<T> items;
    private List<Integer> amounts;
```

Hal tersebut karena Inventory pada konteks ini memiliki dua macam isi yaitu Inventory untuk Engimon dan Inventory untuk Skill Item. InventoryItem adalah sebuah interface yang akan diimplementasi oleh kelas Engimon dan kelas Skill yang isinya adalah method-method yang

digunakan di kelas Inventory. Dengan begitu, T extends InventoryItem juga hanya akan membuat kelas Engimon atau Skill (atau InventoryItem) yang dapat menginstansiasi kelas generic dari Inventory.

2.6. Exception Handling

Penerapan exception handling dengan memanfaatkan throw, try, dan catch untuk menangkap kejadian diluar ekspetasi ketika mendesain program. Penggunaan exception handling akan membuat program tetap berjalan walau terjadi runtime error dan memberikan print out kelayar untuk memberi tahu kesalahan apa yang sebenarnya terjadi.

```
public Engimon breed(Engimon other) throws IllegalArgumentException {
  if (this.level < 4 || other.level < 4) {
    throw new IllegalArgumentException("Level Engimon tidak mencukupi");
  }
```

```
catch (Exception e) {
System.out.println(e.getMessage());
```

Pada kode bagian ini merupakan penanganan kasus ketika breeding dengan menggunakan engimon yang levelnya kurang dari 4. Bagian ini memerlukan *exception* karena method breeding memerlukan *return* engimon namun dalam kasus ini tidak bisa memberikan return berupa engimon.

```
public class PetaException extends Exception{
    public PetaException(){
        System.out.println("error tidak dapat mengenali karakter pada peta!");
    }
}
```

2.7. Java Collection

Java Collection yang digunakan ada 3, yaitu ArrayList, List dan HashMap. ArrayList digunakan untuk menyimpan objek/variabel yang sama dalam satu tempat yang besar kemungkinan akan dimodifikasi isinya, sedangkan List digunakan untuk menyimpan kumpulan objek/variabel yang tidak akan atau jarang berubah. Hal ini karena dengan ArrayList proses modifikasi isi lebih mudah daripada dengan List. HashMap digunakan untuk menyimpan elemen-elemen yang ada sehingga mudah untuk diambil ketika dibutuhkan.

3. Bonus Yang dikerjakan

4. External Library

Implementasi GUI menggunakan Java Swing dan Java AWT, library dipilih karena untuk pembuatan GUI bisa menggunakan drag and drop sehingga tidak perlu menulis kode GUI dari awal. Library juga memiliki referensi dan tutorial yang lebih banyak dari library lain.

5. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
I.1. Engimon	13519037	13519037
I.2. Skill	13519044	13519044
I.3. Player	13519052	13519052
I.3.b Inventory	13519044, 13519052	13519044, 13519052
I.4. Battle	13519035	13519035
I.5. Breeding	13519005	13519005
I.6. Map	13519028, 13519052	13519028
II.1. GUI	13519044, 13519028	13519044, 13519028
II.2. Save Load	13519028, 13519044, 13519052	13519028, 13519044, 13519052