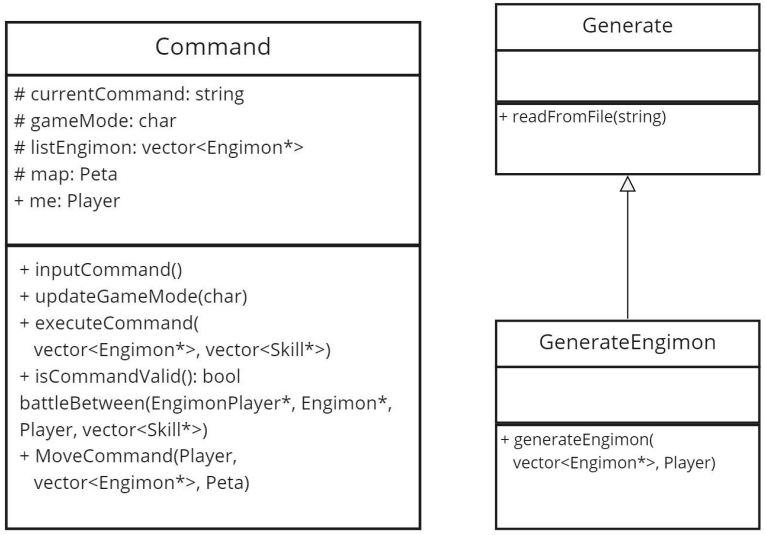
Kelas : K1

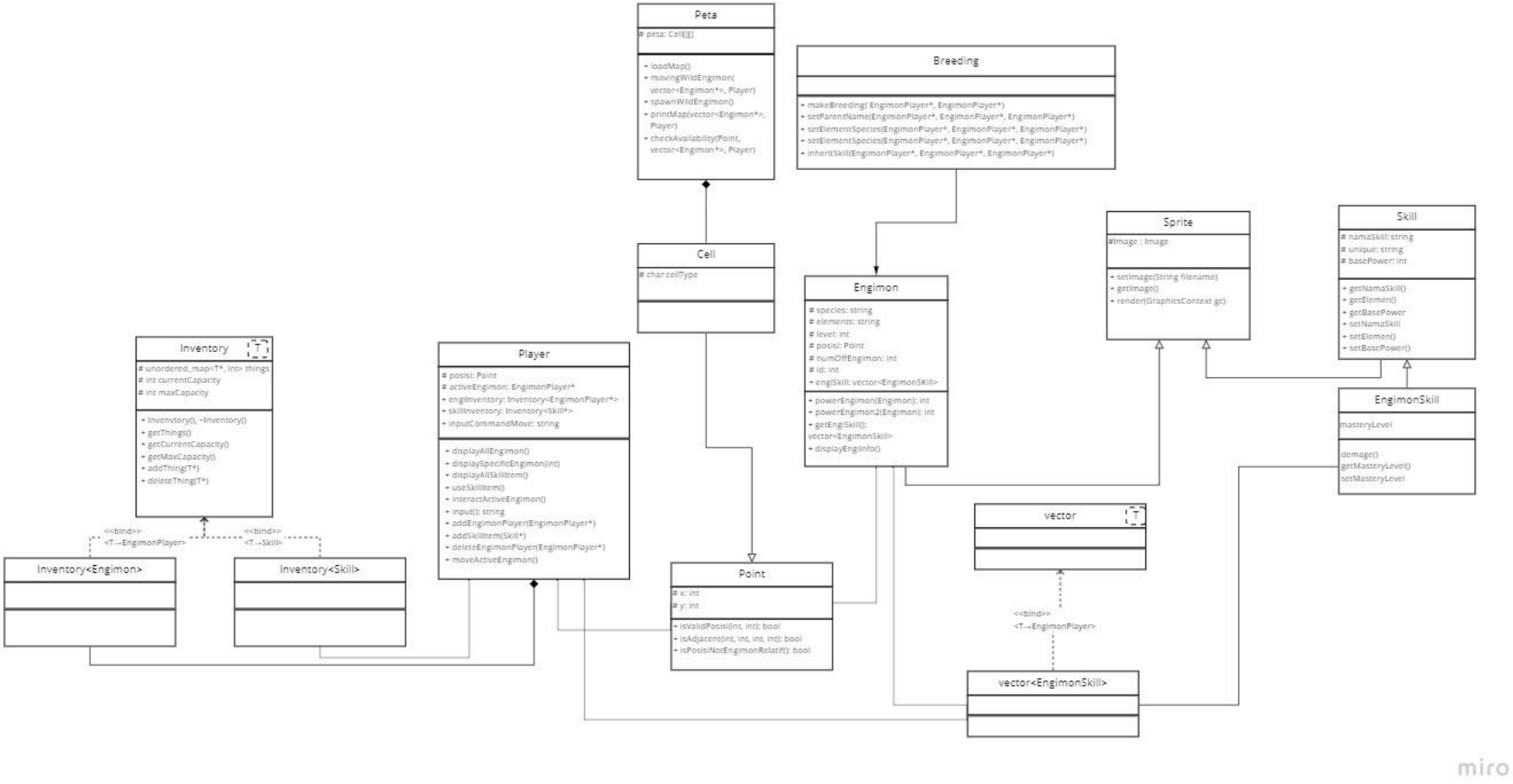
Nama Kelompok : Samlekum Keos

1. 13519007 / Muhammad Tito Prakasa
2. 13519009 / Mochammad Fatchur Rochman
3. 13519010 / Rexy Gamaliel Rumahorbo
4. 13519024 / M Hilal Alhamdy
5. 13519030 / Ferdy Irawan Firdaus
6. 13519038 / Ridho Daffasyah

Asisten Pembimbing : 13517034 / Muhammad Faris Luthfan Wakan

1. Diagram Kelas




Peta
peta: Cell[][]
+ loadMap()
+ movingWildEngimon(vector<Engimon*>, Player)
+ spawnWildEngimon()
+ printMap(vector<Engimon*>, Player)
+ checkAvailability(Point, vector<Engimon*>, Player)
Breeding
+ makeBreeding(EngimonPlayer*, EngimonPlayer*)
+ setParentName(EngimonPlayer*, EngimonPlayer*, EngimonPlayer*)
+ setElementSpecies(EngimonPlayer*, EngimonPlayer*, EngimonPlayer*)
+ setElementSpecies(EngimonPlayer*, EngimonPlayer*, EngimonPlayer*)
+ inheritSkill(EngimonPlayer*, EngimonPlayer*, EngimonPlayer*)
Sprite
#image : Image
+ setImage(String filename)
+ getImage()
+ render(GraphicsContext gc)
Skill
namaSkill: string
unique: string
basePower: int
+ getNamaSkill()
+ getElement()
+ getBasePower
+ setNamaSkill
+ setElemen()
+ setBasePower()
Cell
char cellType
Engimon
species: string
elements: string
level: int
posisi: Point
numOffEngimon: int
id: int
+ engiSkill: vector<EngimonSkill>
+ powerEngimon(Engimon): int
+ powerEngimon2(Engimon): int
+ getEngiSkill(): vector<EngimonSkill>
+ displayEnginfo()
EngimonSkill
masteryLevel
demage()
getMasteryLevel()
setMasteryLevel
Inventory
T
unordered_map<T*, int> things
int currentCapacity
int maxCapacity
+ Invenvtory(), ~Inventory()
+ getThings()
+ getCurrentCapacity()
+ getMaxCapacity()
+ addThing(T*)
+ deleteThing(T*)
Player
posisi: Point
activeEngimon: EngimonPlayer*
+ engiInventory: Inventory<EngimonPlayer*>
+ skillInventory: Inventory<Skill*>
+ inputCommandMove: string
+ displayAllEngimon()
+ displaySpecificEngimon(int)
+ displayAllSkillItem()
+ useSkillItem()
+ interactActiveEngimon()
+ input(): string
+ addEngimonPlayer(EngimonPlayer*)
+ addSkillItem(Skill*)
+ deleteEngimonPlayer(EngimonPlayer*)
+ moveActiveEngimon()
vector
T
<<bind>>
<T→EngimonPlayer>
<<bind>>
<T→Skill>
Inventory<Engimon>
Inventory<Skill>
Point
x: int
y: int
+ isValidPosisi(int, int): bool
+ isAdjacent(int, int, int, int): bool
+ isPosisiNotEngimonRelatif(): bool
<<bind>>
<T→EngimonPlayer>
vector<EngimonSkill>
miro

2. Penerapan Konsep OOP

2.1. Polymorphism

```
//drop item
try {
    Skill dropSkill = currentPlayer.checkOwned(eng2.getEngiSkill().get(0));
    currentPlayer.skillInventory.addThing(dropSkill);
    outputText1.setText(eng1.get_name() + " Telah Menang! \nMendapatkan " + diffPower*10 + "Exp \nMenda
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

Polymorphism di sini berguna karena skill yang di-drop oleh engimon bertipe EngimonSkill sedangkan inventory menerima objek bertipe Skill dalam method addThing. Sehingga tidak perlu repot-repot membuat objek Skill baru melainkan memanfaatkan objek turunan Skill yaitu EngimonSkill

2.2. Inheritance/Composition/Aggregation

```java
package sample.BackEnd;

public class EngimonSkill extends Skill{
    protected int masteryLevel;

    protected static int MAX_MASTERY_LEVEL = 3;

    public EngimonSkill() { super(); }
    public EngimonSkill(String skill, int power, String element){
        super(skill, element, power);
        this.masteryLevel = 1;
    }
    public EngimonSkill(Skill baseSkill){
        super(baseSkill);
        this.masteryLevel = 1;
    }
    public int damage() { return basePower*masteryLevel; }
    public int getMasteryLevel() { return this.masteryLevel; }
    public void setMasteryLevel(int masteryLevel){
        if (masteryLevel<=MAX_MASTERY_LEVEL){
            this.masteryLevel = masteryLevel;
        }else{
            this.masteryLevel = MAX_MASTERY_LEVEL;
        }
    }
}
```

File tersebut merupakan contoh penerapan inheritance, class engimon skill merupakan subclass dari class skill. Dalam class skill telah memiliki behavior umum dari skill engimon yaitu nama skill, unique, dan base power. Keuntungan memakai ini adalah tidak perlu mengulang-ngulang penulisan kode lagi dan class engimon skill ini memiliki behavior yang lebih spesifik dari skill , namun memiliki semua behavior dari superclass nya yaitu Skill.

2.3. Abstract Class

```java
package sample;

import javafx.scene.image.Image;
import javafx.scene.canvas.GraphicsContext;
import javafx.geometry.Rectangle2D;

import java.io.File;

public abstract class Sprite
{
    protected Image image;

    public Sprite() { }

    public void setImage(String filename) { image = new Image(filename); }

    public Image getImage() { return  image; }

    public abstract void render(GraphicsContext gc);

}
```

Sprite akan diturunkan menjadi Engimon dan Skill, mengapa hal ini dibutuhkan karena dalam GUI diperlukan visualisasi masing-masing entitas Engimon dan Skill sehingga dengan meng-inherit abstract class Sprite yang tadinya Engimon dan Skill tidak memiliki gambar sekarang mereka memiliki gambar yang dapat ditampilkan.

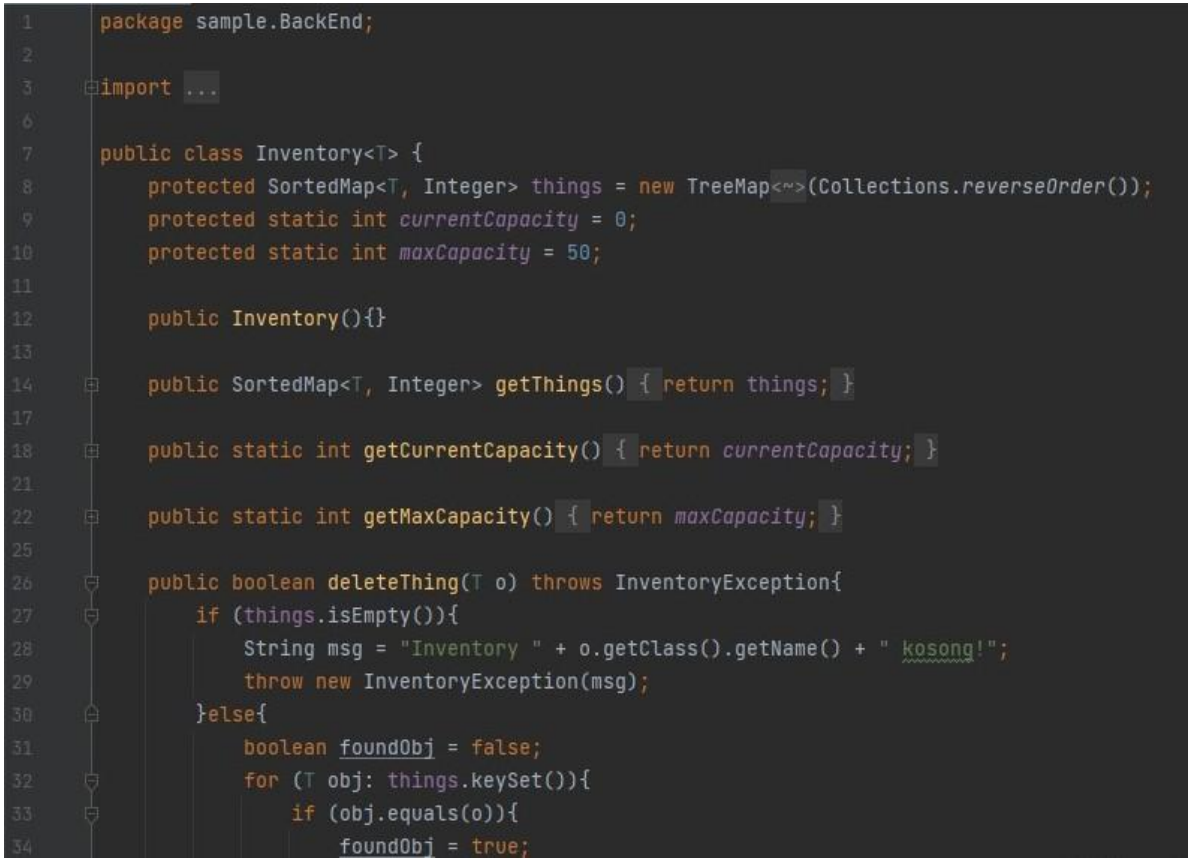
2.4. Interface

```
public class Skill implements Comparable<Skill>{
    protected String namaSkill;
    protected String unique;
    protected int basePower;

    public Skill(){...}
    public Skill(String namaSkill, String unique, int basePower){...}
    public Skill(Skill S){...}
    public String getNamaSkill() { return this.namaSkill; }
    public String getUnique() { return this.unique; }
    public int getBasePower() { return this.basePower; }
    public void setNamaSkill(String namaSkill) { this.namaSkill = namaSkill; }
    public void setUnique(String unique) { this.unique = unique; }
    public void setBasePower(int basePower) { this.basePower = basePower; }
    @Override
    public int compareTo(Skill o) {
        if (this.getBasePower()>o.getBasePower()){
            return 1;
        }else if (this.getBasePower()==o.getBasePower()){
            return 0;
        }else{
            return -1;
        }
    }
}
```

Pada Class Skill, Point, dan Engimon diterapkan interface Comparable agar dapat melakukan komparasi antar objek masing-masing kelas (hal ini krusial dalam pensortiran engimon dan skill item yang ditampilkan).

2.5. Generic Type & Wildcards



```
package sample.BackEnd;

import ...

public class Inventory<T> {
    protected SortedMap<T, Integer> things = new TreeMap<~>(Collections.reverseOrder());
    protected static int currentCapacity = 0;
    protected static int maxCapacity = 50;

    public Inventory(){}

    public SortedMap<T, Integer> getThings() { return things; }

    public static int getCurrentCapacity() { return currentCapacity; }

    public static int getMaxCapacity() { return maxCapacity; }

    public boolean deleteThing(T o) throws InventoryException{
        if (things.isEmpty()){
            String msg = "Inventory " + o.getClass().getName() + " kosong!";
            throw new InventoryException(msg);
        }else{
            boolean foundObj = false;
            for (T obj: things.keySet()){
                if (obj.equals(o)){
                    foundObj = true;
```

Dari spek, dibutuhkan inventory yang bisa menyimpan engimon dan skill. Berangkat dari sana maka dibanding membuat kelas inventory untuk masing-masing kelas (engimon dan skill) lebih baik kita memanfaatkan fitur generic types.

2.6. Exception Handling

```
try{
    Engimon engi2 = cmd.findWildEngi(ListOfGeneratedEngimon,P);
    battle(P.getActiveEngimon(),engi2,primaryStage);
    primaryStage.setScene(battle);
}catch(Exception e) {
    System.out.println(e.getMessage());
}
```

Maksud dari potongan kode di atas adalah jika ditemukan engimon di sekitar player maka lakukan battle, namun jika tidak ditemukan maka akan melemparkan exception dan tidak dilakukan battle (juga pemindahan frame ke scene “battle”). Sehingga akan terasa smooth apabila melakukan percobaan battle (akan pindah scene battle jika ditemukan dan akan tetap di scene map jika tidak ditemukan).

2.7. Java Collection

```java
package sample.BackEnd;

import java.util.Collections;
import java.util.SortedMap;
import java.util.TreeMap;

public class Inventory<T> {
    protected SortedMap<T, Integer> things = new TreeMap<T,Integer>(Collections.reverseOrder());
    protected static int currentCapacity = 0;
    protected static int maxCapacity = 50;
```

```java
package sample.BackEnd;

import javafx.scene.canvas.GraphicsContext;
import sample.Sprite;
import java.util.List;
import java.util.Scanner;
import java.util.Vector;
import java.util.Random;

public class Engimon extends Sprite implements Comparable<Engimon> {
    //protected atribut
    /*1*/ protected String name;
    /*2*/ protected int life;
    /*3.1*/ protected String parentsName;
    /*3.2*/ protected String species;
    /*4*/ protected List<EngimonSkill> engiSkill = new Vector<>();
    /*5*/ protected String elements;
    /*6*/ protected int level;
    /*7*/ protected int exp;
    /*8*/ protected int cumExp;
    protected boolean isOwnedByPlayer;
```

Dalam penerapan java collection, pada inventory, memudahkan karena pengalokasian memorynya dinamis dan tidak fixed size.

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Multi-threading (real-time gameplay)

Pada program yang kami buat, *multi-threading* diterapkan dalam implementasi pergerakan Engimon liar. Sebelumnya, Engimon liar bergerak berdasarkan sistem *turn-based*, yakni setiap Player bergerak; dengan menggunakan *multi-threading*, Engimon dapat bergerak secara independen secara periodik. Untuk setiap Engimon liar, dibuat object *runnable EngimonThread* yang berfungsi menggerakkan Engimon dengan sebuah *thread*. Selain itu, *threading* digunakan juga pada *SpawnEngimon*, untuk *spawning* Engimon setiap waktu tertentu.

3.1.2. Unit Testing Implementation

Unit testing menggunakan JUnit (memerlukan instalasi JUnit terlebih dahulu jika sebelumnya belum pernah mengunduh *library* JUnit) yaitu *assertEquals*, *assertNotEquals, assertNull, dan aseertNotNull*. Diimplementasikan tujuh kelas unit testing yaitu kelas *Breeding, Cell, Engimon, Engimon Skill, Inventory, Point,* dan *Skill*. *Unit testing* sendiri diimplementasikan dalam setiap *method* di dalam tujuh kelas tersebut dan setiap *method* diimplementasikan dua *unit testing* yaitu skenario ketika method tersebut sukses (*success*) dan skenario ketika method tersebut gagal (*fail*).

4. External Library

Untuk menampilkan gameplay dari Tugas Besar OOP 2 ini, kelompok kami mengimplementasi GUI menggunakan JavaFx 16, dan JUnit 5.4 untuk unit testing.

5. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
1. Engimon	Ridho Daffasyah	Ridho Daffasyah
2. Skill	M Hilal Alhamdy	M Hilal Alhamdy
2.a. Engimon Skill	M Hilal Alhamdy	M Hilal Alhamdy
3.a. Command	Rexy Gamaliel	Rexy Gamaliel
3.b. Inventory	Muhammad Tito	Muhammad Tito
3.c. Active Engimon	Ferdy Irawan & Rexy G	Ferdy Irawan & Rexy G
4. Battle	Fatchur & Hilal	Fatchur & Hilal
5. Breeding	Ferdy Irawan Firdaus	Ferdy Irawan Firdaus
6. Peta	M Fatchur	M Fatchur
7. Generate	M Tito Prakasa	M Tito Prakasa
7.a. Generate Engimon	M Tito Prakasa	M Tito Prakasa

8. Point	Ferdy Irawan Firdaus	Ferdy Irawan Firdaus
9. Player	Ferdy Irawan & Rexy G	Ferdy Irawan & Rexy G
10. Main Program	Semua	Semua
11. Multithread	Rexy G	Rexy G
12. Unit Testing	Ferdy Irawan	Ferdy Irawan
13. GUI	Ridho Daffasyah dan Tito Prakasa	Ridho Daffasyah dan Tito Prakasa