

SPAKBOR HILL'S

HidupUNPAD Group 10 - K04

Persada Ramiza Abyudaya - 18223033
Devon Viraditya Tanumihardja - 18223039
Surya Suharna - 18223075
Velicia Christina Gabriel - 18223085





Table of Contents

1 Table of Contents

2 Overview Game

3 User Manual

4 Technical Concept

5 Design Pattern Applied

6 Final Structure

7 Explanation of Changes

8 Gameplay

9 The Story Behind

10 Division of Tasks

11 Documentations

Overview Game



Spakbor Hills adalah game simulasi bertani berbasis Java dengan antarmuka grafis (GUI) menggunakan Java Swing. Pemain dapat bertani, memancing, memasak, menjual hasil panen, dan menjalin relasi dengan NPC. Didukung sistem musim, cuaca, energi, serta fitur interaktif berbasis GUI, game ini menawarkan pengalaman bermain yang strategis dan mendalam tanpa batas waktu, dengan milestone seperti kekayaan dan pernikahan.

Selain bertani dan pengelolaan energi untuk berbagai aktivitas, Spakbor Hills menawarkan dunia yang hidup, memungkinkan pemain menjalin hubungan mendalam dengan para penduduk desa (NPC). Kedekatan dengan mereka dapat ditingkatkan melalui pemberian hadiah yang disukai atau dibenci, yang seiring waktu akan membuka berbagai event cerita unik. Bagi pemain yang mencari teman hidup, beberapa karakter tertentu bahkan dapat dinikahi, menambahkan dimensi personal pada perjalanan mencapai kekayaan dan kebahagiaan di Spakbor Hills. Kehidupan sosial ini berjalan selaras dengan siklus musim dan cuaca yang dinamis, yang juga mempengaruhi aktivitas lain seperti memancing beragam jenis ikan sesuai kondisinya dan memasak hidangan lezat dari resep-resep yang berhasil dikumpulkan.



Male
Character



Female
Character



User Manual

Step 1

Mendownload JDK melalui website <https://www.oracle.com/id/java/technologies/downloads/> dan JavaFX plugin melalui website <https://openjfx.io/>. JavaFX diperlukan dengan tujuan sebagai platform yang memungkinkan pengguna untuk memakai user interface yang menarik pada aplikasi yang dikembangkan

Step 2

Melakukan git clone pada github repository
<https://github.com/suryasuharna23/Tugas-Besar-OOP.git>

Step 3

Membuka folder hasil clone melalui terminal atau command prompt dan pastikan directory sudah sesuai dengan nama folder

Step 4

Menjalankan permainan dengan cara mengetikkan `./gradlew run` pada command prompt atau terminal.

Technical Concept

Inheritance

Kelas Player (pemain) dan NPC (karakter non-pemain) mewarisi dari kelas Entity (entitas game). Artinya, Player dan NPC otomatis mendapatkan kemampuan dasar semua entitas game (seperti punya posisi di peta worldX, worldY dan bisa digambar), lalu Player menambahkan hal spesifik seperti inventaris, dan NPC punya dialog.

```
public class Player extends Entity implements Observer{  
    KeyHandler keyH;  
    public final int screenX;  
    public final int screenY;  
    private String fName;  
    public int currentEnergy;
```

```
public class NPC extends Entity { & PlayerRandomizer  
    public int maxHeartPoints = 150; 5 usages  
    public int currentHeartPoints = 0; 19 usages  
    public boolean isMarriageCandidate = false; 9 usages  
    public boolean hasReceivedGiftToday = false;
```

Abstract Class/Interface

Interface Edible mendefinisikan "kontrak" bahwa item yang bisa dimakan harus memiliki metode eat(Player player). Kelas seperti OBJ_Food (makanan) dan OBJ_Drop (hasil panen) mengimplementasikan Edible, sehingga keduanya punya metode eat() spesifik mereka.

```
public interface Edible { 16 usages  
    public void eat(Player player);  
}
```

Polymorphism

Di GamePanel, ada ArrayList<Entity> entities yang bisa berisi berbagai macam entitas game (seperti Player, NPC, OBJ_Chest). Ketika game menggambar semua entitas ini dengan memanggil entity.draw(g2) untuk setiap entity dalam daftar tersebut, metode draw() yang spesifik untuk Player akan dijalankan untuk objek Player, metode draw() untuk NPC akan dijalankan untuk objek NPC, dan seterusnya.

```
public Player player = new Player();  
public ArrayList<Entity> entities = new ArrayList<>();  
public ArrayList<NPC> npcs = new ArrayList<>(); 8 usages  
public Entity currentInteractingNPC = null; 30 usages
```

```
for (Entity entity : entities) {  
    entity.draw(g2);  
}
```

Generics

Penggunaan ArrayList<Entity> entities di GamePanel. Tanda <Entity> berarti ArrayList tersebut secara spesifik dirancang untuk menyimpan objek—objek yang bertipe Entity (atau turunan dari Entity). Ini membantu mencegah kesalahan karena anda tidak bisa secara tidak sengaja menambahkan objek yang bukan Entity ke dalamnya.

```
public class GamePanel extends JPanel implements Runnable {  
    public Player player = new Player();  
    public ArrayList<Entity> entities = new ArrayList<>();  
    public ArrayList<NPC> npcs = new ArrayList<>();  
    public Entity currentInteractingNPC = null;
```

Exceptions

Dalam metode Entity.setup(String imagePath), ada blok try-catch. Ini mencoba memuat gambar untuk entitas. Jika gambar tidak ditemukan (menyebabkan IOException), program tidak akan crash. Sebaliknya, blok catch akan menangani error tersebut, misalnya dengan mencetak pesan ke konsol bahwa gambar gagal dimuat.

```
} catch (IOException e) {  
    System.out.println("Exception in Entity.setup for " +  
        entity);  
} catch (IllegalFormatException e) {  
    System.out.println("IllegalArgument Exception in Entity.  
        " + e.getMessage());  
}
```

Concurrency

Kelas GameClock (jam game) berjalan sebagai Thread terpisah dari GamePanel (panel utama game). Ini berarti jam dalam game (waktu, musim, cuaca) bisa terus berjalan dan diperbarui secara independen, sementara pada saat yang sama GamePanel menangani input pemain, pergerakan karakter, dan penggambaran ke layar.

```
synchronized (pauseLock) {  
    if (paused) {  
        try {  
            pauseLock.wait();  
        } catch (InterruptedException e) {  
            if (running) break;  
            Thread.currentThread().interrupt();  
        }  
    }  
}
```

Design Pattern

```
public class FoodRegistry { <uses & PeriodicHarvests>
    private static final FoodRegistry foodRegistry = new FoodRegistry();

    private static Map<String, Function<GamePanel, OBJ_Food>> registry;

    private FoodRegistry() { registry = new HashMap<>(); }

    public static void registerFood(String foodName, Function<GamePanel, OBJ_Food> food) {
        registry.put(foodName, food);
    }
}
```

Registry Pattern

Pola Registry menyediakan cara terpusat untuk mendapatkan akses ke objek atau, dalam kasus ini, cara untuk membuat objek (fungsi membuat objek). Ini seperti memiliki katalog pusat di mana setiap entri adalah "resep" atau "blueprint" untuk membuat objek tertentu, yang bisa diambil berdasarkan nama.

```
package spakborhills.action;

import spakborhills.GamePanel;
import spakborhills.items.Item;

public interface Command { @usage & FPISScene
    public void execute(GamePanel gp);
}

public class PlantingCommand implements Command { @usage & FPISScene
    private final Player player; @usage
    public PlantingCommand(Player player) { this.player = player; }

    @Override @usage & FPISScene
    public void execute(GamePanel gp) {
        Entity equipped = player.getEquippedItem();
    }
}
```

Command Pattern

Pola Command mengubah sebuah permintaan atau aksi menjadi sebuah objek yang berdiri sendiri. Ini memungkinkan parameterisasi klien dengan aksi yang berbeda, mengantrekan aksi, atau mendukung operasi yang bisa dibatalkan.

```
package spakborhills.interfaces;

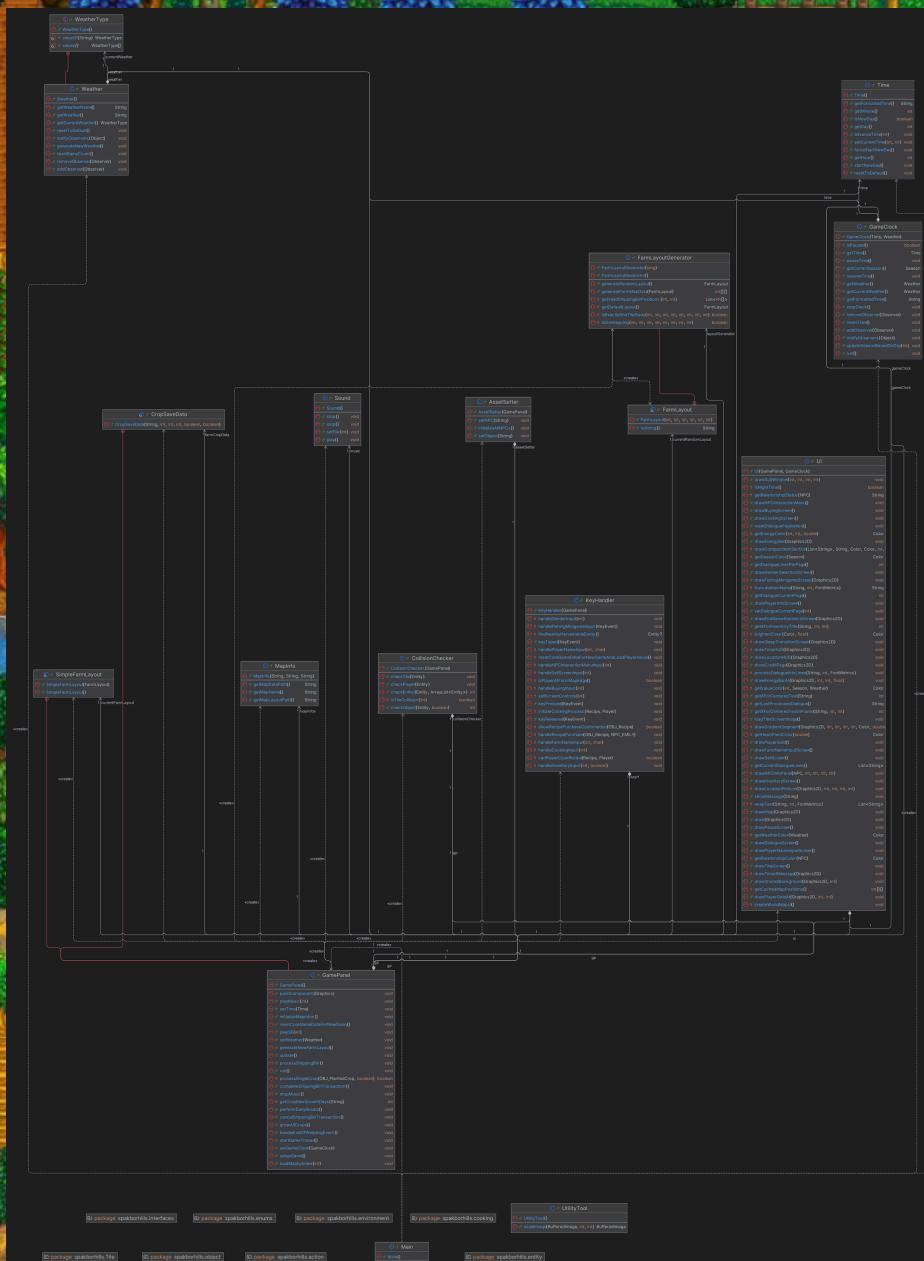
public interface Observer { @usage
    void update(Object event); 2M
}

public interface Observable { @usage
    void addObserver(Observer observer); 2M
    void removeObserver(Observer observer);
    void notifyObservers(Object event); 2M
}
```

Observer Pattern

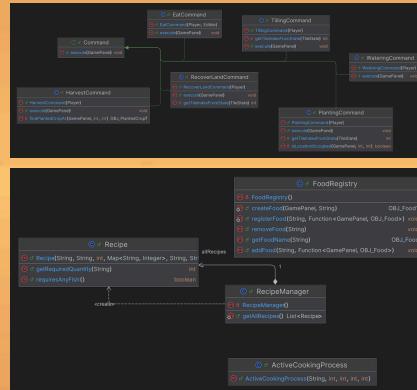
Pola Observer mendefinisikan mekanisme berlangganan di mana beberapa objek (Observers) dapat memantau dan bereaksi terhadap perubahan keadaan pada objek lain (Subject/Observable).

Final Structure

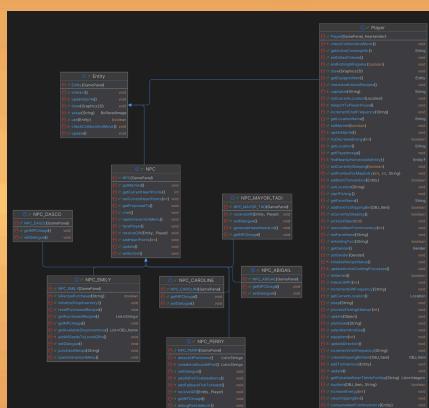


Explanation of Changes

Terdapat pemisahan beberapa action yang dapat dilakukan player menggunakan design pattern. Selain action-action di atas, action dilakukan di kelas-kelas yang berkaitan, seperti GamePanel.java, NFC.java, dan Player..java,



Kami melakukan perubahan pada susunan kelas untuk NPC. Pada milestone 1, kami berencana untuk membuat NPC menjadi satu kelas dan setiap NPC hanya berbeda atribut. Namun, pada milestone 2 kami memutuskan untuk melakukan breakdown structure pada NPC sehingga setiap NPC memiliki kelas dengan namanya masing-masing.



Gameplay



Menu pertama kali sebelum masuk game



Salah satu menu Credits berisikan apresiasi kepada pihak yang telah membantu



Ketika masuk ke New Game, player diminta memasukkan namanya



Ketika masuk ke New Game, player diminta memasukkan nama farm



Menu World Map berisikan semua Map di dalam game



Map Player House yang mana merupakan rumah bagi Player



Map salah satu rumah NPC yaitu Abigail



Map salah satu rumah NPC yaitu Caroline



Map salah satu rumah NPC yaitu Dasco



Map salah satu rumah NPC yaitu Major Tadi



Map salah satu rumah NPC yaitu Perry



Map Forest River di mana Player dapat meremching di tempat ini



Map Mountain Lake di mana Player dapat meremching di tempat ini



Map Ocean di mana Player dapat meremching di tempat ini



Map Farm Player di mana Player dapat bercocok tanam



Map Store Emily di mana Player dapat meremching di tempat ini

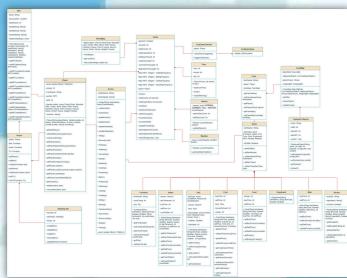


The Story Behind

Pada suatu malam yang cerah tanggal 16 April 2025, tiba-tiba petir menyambar. Sebuah notifikasi masuk ke dalam handphone kami. Seperti yang mudah ditebak, TUGAS BESAR OBJECT ORIENTED PROGRAMMING. Kami menghela napas sebentar lalu perjalanan panjang dan melelahkan itu dimulai. :)

Milestone 1

Setelah itu masing-masing dari kami melakukan brainstorming untuk menentukan struktur class diagram yang akan kami implementasikan setelah itu. Kami benar-benar bingung, namun akhirnya kami membuat juga class diagram tersebut. Sejujurnya pada saat itu kami ragu pada struktur class diagram yang kami buat tapi pada akhirnya seperti beberapa hal yang terjadi di hidup kita ‘jalanin dulu aja’.



Pada 24 April 2025 kami melakukan Asistensi 1 bersama Kak Abil dan yappp. Kami rasa memang banyak dari struktur class diagram yang kami buat yang perlu diimprove lagi. Tapi kami cukup lega ketika Kak Abil mengatakan bahwa class diagram masih bisa direvisi dan tidak menjadi acuan baku ketika implementasi di Milestone 2.

Dari input yang kami terima sebenarnya class diagram yang kami buat secara umum sudah cukup baik. Namun ada beberapa masukkan dari Kak Abil terkait Game State. Suatu kelas general yang menyimpan state dari game yang sedang di jalankan. Pada asistensi tersebut juga kami bersyukur mendapat banyak pencerahan terlebih terkait framework yang akan kami gunakan pada implementasi di Milestone 2 nanti.



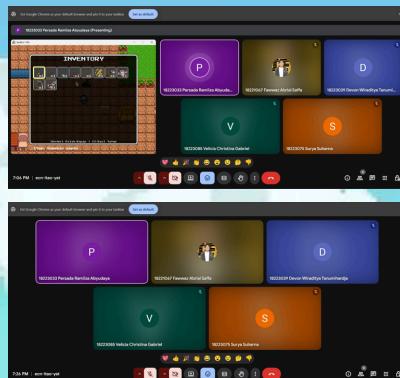


The Story Behind



Setelah merancang struktur class diagram, tidak lama kami membuat repository Github sebagai workspace kami mengerjakan tugas besar ini. Lalu kami bertemu untuk melakukan diskusi terkait pembagian tugas. Pada saat itu Surya mengusulkan untuk membuat GUI semua map terlebih dahulu agar kami dapat mengimplementasikan logic programnya di map yang sudah dibuat.

Satu per satu code kami tulis, kopi demi kopi sudah kami teguk agar mata kami tetap aktif untuk menyelesaikan tugas besar ini. Akhirnya program sudah hampir selesai dari hasil kerja keras kami. Entah berapa kali keyboard masing-masing laptop kami ditekan, pada akhirnya 27 Mei 2025 kami melakukan asistensi kembali bersama Kak Abil. Dari asistensi tersebut kami mendapatkan beberapa konfirmasi dari spesifikasi yang belum sepenuhnya kami pahami. Asistensi pun selesai dan kami melanjutkan menulis kode dengan hati yang riang. :)



Di malam yang sama setelah kami melakukan asistensi, Hidung Persada tiba-tiba mengeluarkan darah. Kami semua panik, sepertinya ia kurang sehat atau kelelahan. Pada akhirnya memang ada harga yang harus dibayar untuk menjadi #itbboys ini.

Dengan napas yang lepas dan hati yang lega akhirnya kami berhasil menyelesaikan Tugas Besar Object Oriented Programming ini. Waktu yang kami lalui bukanlah waktu yang mudah tapi tugas besar yang bagi kami sulit itu bukan berarti mustahil diselesaikan. Terima kasih untuk Surya, Persada, Wete, dan Veli yang telah bersama-sama menyelesaikan tugas ini. Mudah-mudahan kita semua sukses!

Division of Tasks

Selengkapnya



Persada

1. Menginisiasi pembuatan GUI.
2. Mengerjakan berbagai action pada game.
3. Melakukan pembuatan kelas yang berkaitan dengan Player.
4. Membuat kelas-kelas yang berkaitan dengan NPC.
5. Melakukan debugging dan testing.



Wete

1. Membuat action Tilling, Watering, Planting, dan Harvesting.
2. Membuat Time, Season, Weather.
3. Membuat pemilihan gender Player dan player info.
4. Melakukan debugging dan testing.



Surya

1. Mengatur flow kerja kelompok.
2. Melakukan pengelolaan assets dan membuat maps untuk GUI.
3. Membuat action Fishing.
4. Mengerjakan pembuatan booklet.
5. Melakukan debugging dan testing.



Velicia

1. Melakukan pengelolaan assets.
2. Membuat World map dan map-map lainnya.
3. Membuat kelas-kelas yang berkaitan dengan Items.
4. Membuat end game.
5. Melakukan debugging dan testing.

Documentations



Ini hari pertama kami mengerjakan tugas besar di luar kampus. Thanks to Jabarano Dago sudah menjadi tempat kami mengerjakan tugas besar ini.



Hurry Up! Deadline semakin dekat. Lagi-lagi Jabarano tempatnya. Seperti biasa kami bekerja mengerjakan tugas ini sejak sore hingga tengah malam dan berulang terus sejak setidaknya 5 hari terakhir sebelum pengumpulan deadline.



Intermezo! Tiba-tiba kami mendapatkan informasi dari social media X seperti pada gambar. Terima kasih untuk siapapun yang mengirimkan. Kami merasa diingatkan.



Thank You!



IF2010 Pemograman
Berorientasi Objek



Wise men say that
this is all 'nurturing'