

# SPAKBOR HILLS

One Man Carry

## K05 - Kelompok 1

- |          |                                |
|----------|--------------------------------|
| 18223076 | Michael Dimas Sarono           |
| 18223040 | Florecito Natawiryo            |
| 18223026 | Jacob Reinhard Marudut Siagian |
| 18223068 | Muhammad Arya Putra Rrihastomo |
| 18223036 | Favian Rafi Loftiyanto         |



# User Manual

Spakbor Hills adalah sebuah game simulasi pertanian berbasis Java dengan antarmuka Command Line Interface (CLI), di mana pemain berperan sebagai Dr. Asep Spakbor yang mencoba memulai hidup baru sebagai petani. Dalam game ini, pemain dapat melakukan berbagai aktivitas seperti menanam dan memanen tanaman, memancing, memasak, berinteraksi dengan NPC, serta menjual hasil produksi untuk mengumpulkan gold. Permainan mengikuti sistem waktu nyata, musim, dan cuaca yang memengaruhi aktivitas harian. Tujuan utama permainan adalah mencapai dua milestone: mengumpulkan 17.209 gold dan menikahi salah satu NPC, dengan permainan yang bersifat open-ended dan terus berlanjut meskipun milestone telah tercapai.

# User Manual

## How to Install

1. Buka [Buko.github.com/MichaelDimas28/Tubes\\_IF2010\\_K5-1/](https://github.com/MichaelDimas28/Tubes_IF2010_K5-1) kemudian download ZIP dan extract.
2. Buka CMD di direktori tempat file diekstrak. Pastikan cd di src
3. Kompilasi seluruh file Java dengan perintah berikut: javac \*.java
4. Jalankan Game dengan perintah: java Main

## Game Control

<b>W</b>	/	<b>↑</b>	geser ke atas	<b>P</b>	pause
<b>A</b>	/	<b>←</b>	geser ke kiri	<b>I</b>	open inventory
<b>S</b>	/	<b>↓</b>	geser ke bawah		
<b>D</b>	/	<b>→</b>	geser ke kanan		

# User Manual

## Inventory Game Control

**ENTER** : Hold/Unhold item

**W S D** / **↑ ↓ ← →** : navigasi inventory

Depan Shipping Bin + **SPACE** : Open Shipping Bin

## Game Control

itemHeld Watering Can + **ENTER** + tilled land: Siram tanaman

itemHeld Hoe + **ENTER** + tillable Land : Pacul tanah

itemHeld Pickaxe + **ENTER** + Tilled Land : Recover Land

itemHeld Fishing Pole + **ENTER** + Area Tile Air : Mancing

itemHeld anyItem + NPC + **SPACE** : Gifting

Depan NPC + NO itemHeld + **SPACE** : Chatting

itemHeld + **F** : Kasih barang

**ESC** : Melakukan aksi

**SPACE** : Keluar

# Game Play



## Awal Permainan

Tampilan awal permainan ketika memasuki game. Player spawn di Farm Map dengan energy 100 dan Gold 0g. Waktu menunjukkan Day 1



## Shipping Bin

Untuk membuka shipping bin tekan 'space' kemudian pilih item yang ingin dijual, tekan 'enter' untuk menuruhnya. Player dapat menjual item di Shipping Bin kecuali equipment.



## Tilling & Recover Land

Gunakan hoe untuk menggemburkan tanah, gunakan pickaxe untuk recover tanah. Masing-masing membuat energi berkurang 5.



## Fishing

Pegang Fishing Rod kemudian tekan enter untuk memancing. Tebak angka 1-10, ketika benar, pancingan akan tertarik dan lihat apakah mendapat ikan atau tidak. Energi berkurang 5.

# Game Play



## Planting & Watering

Taruh seeds yang diinginkan di tanah yang sudah digembur, kemudian gunakan watering can untuk menyiramnya.



## World Map & Ocean

Jalan ke ujung map untuk ke direkt ke world map. Pada kanan bawah terdapat ocean dan bisa memancing dengan menebak angka lagi untuk mendapat ikan.



## Mountain Lake & Forest River

Berikut tampilan mountain lake yang ada di kanan atas world map dan forest river yang ada di kiri world map.



## Chatting & Gifting

Mengunjungi NPC House dan berinteraksi dengannya. Mengobrol dan memberikan hadiah. Pegang item yang ingin diberikan kemudian tekan 'F'.

# Game Play



## Sleeping & Watching

Dekati TV kemudian tekan 'Enter' untuk menonton. Dekati ranjang dan tekan 'Enter' untuk tidur, nanti akan muncul "Hari telah berganti".



## Harvesting

Ketika sudah waktunya untuk diharvest, gunakan tangan kosong dan tekan 'Enter' untuk memanen. Tanah kembali ke semula.



## Eating & Cooking

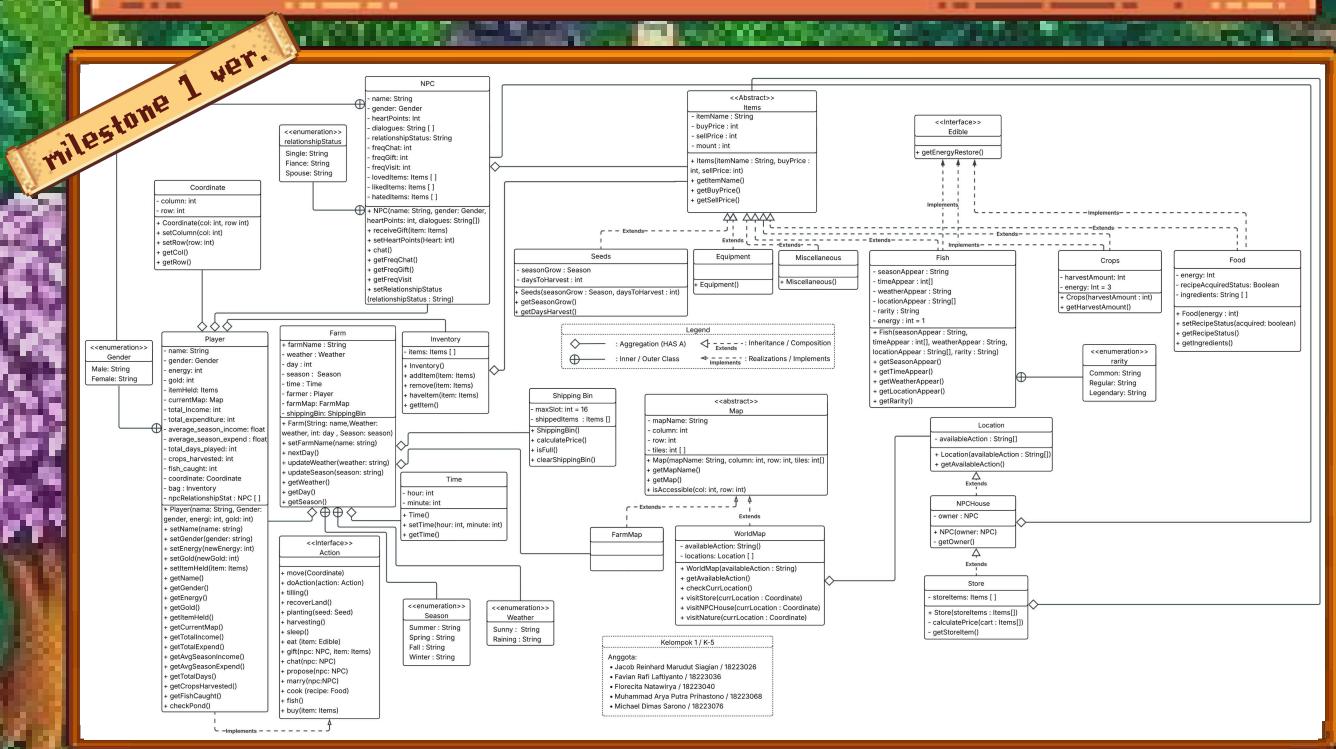
Pegang item yang ingin dimakan, kemudian tekan 'space'. Siapkan bahan, kemudian pergi ke kompor dan mulai memasak, akan muncul menu yang harus dipilih.



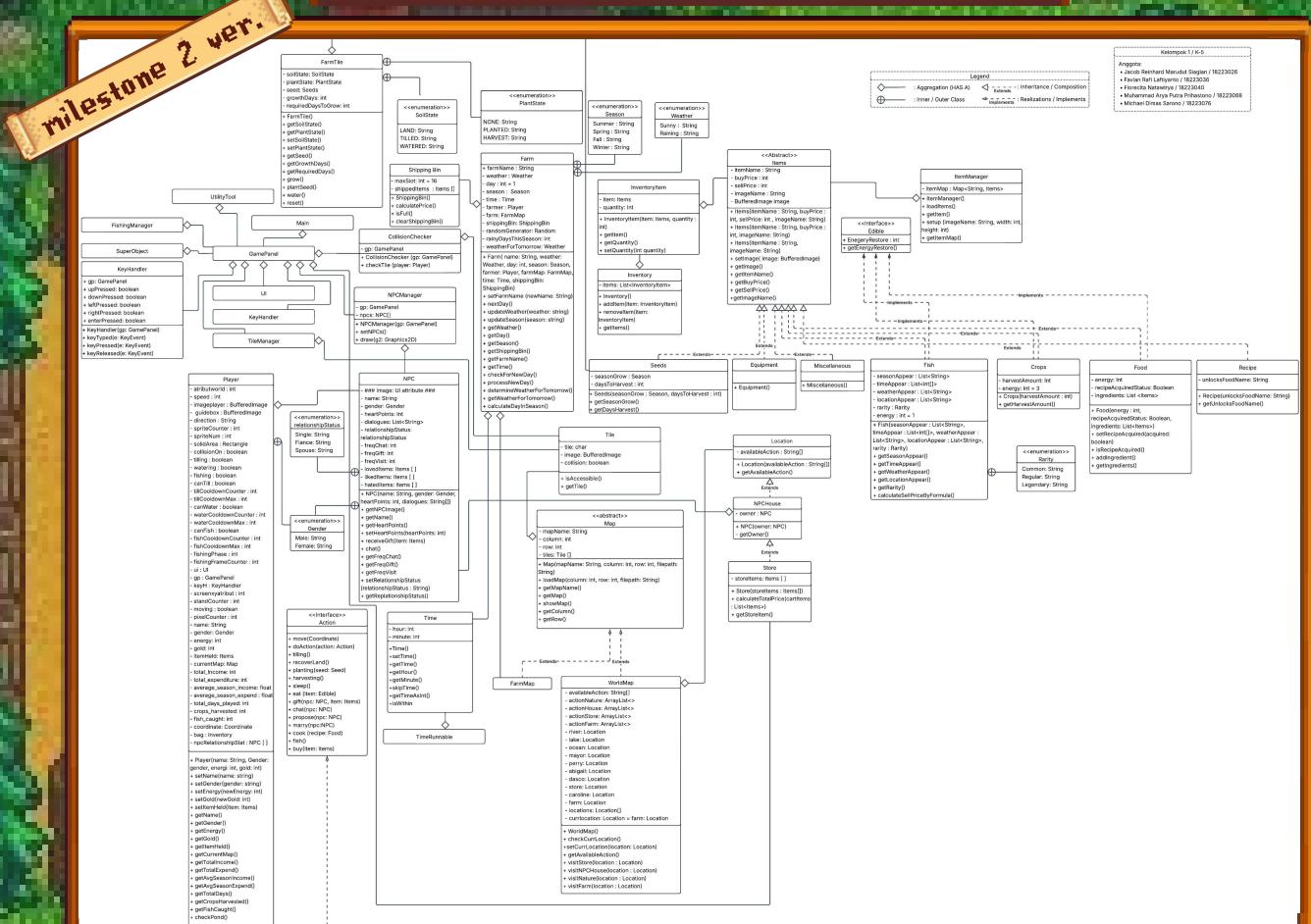
## Proposing & Marry

Dekati NPC yang ingin dilamar, pegang 'Proposal Ring' kemudian tekan 'F' lihat apakah lamaranmu diterima atau ditolak. Datang lagi setelah 1 hari untuk menikah.

## Class Diagram



[tinyurl.com/ClassDiagramOOP](http://tinyurl.com/ClassDiagramOOP)



# Design Pattern

## Composite

### Structural Pattern

Composite adalah sebuah design pattern struktural yang memungkinkan pengguna menyusun objek menjadi struktur hierarki tree dengan pola rekursif pada komposisi objeknya.

```
public abstract class Items {
    private String itemName;
    protected int buyPrice;
    protected int sellPrice;
    private String imageName;
    private BufferedImage image;

    public abstract void eat();

    public String getItemName() {
        return itemName;
    }

    public int getBuyPrice() {
        return buyPrice;
    }

    public int getSellPrice() {
        return sellPrice;
    }

    public String getImageName() {
        return imageName;
    }

    public BufferedImage getImage() {
        return image;
    }
}

public class Food extends Items implements Edible {
    private int energy;
    private boolean recipeAcquired;
    private List<Items> ingredients;

    // Constructor untuk food biasa (tanpa resep, bisa dibeli)
    public Food(String itemName, int buyPrice, int sellPrice, int energy, String imageName) {
        super(itemName, buyPrice, sellPrice, imageName);
        this.energy = energy;
        this.recipeAcquired = false;
        this.ingredients = new ArrayList<>();
    }

    public void eat() {
        if (recipeAcquired) {
            System.out.println("Food has been eaten!");
        } else {
            System.out.println("Food has not been eaten yet.");
        }
    }

    public void setEnergy(int energy) {
        this.energy = energy;
    }

    public int getEnergy() {
        return energy;
    }

    public void setRecipeAcquired(boolean recipeAcquired) {
        this.recipeAcquired = recipeAcquired;
    }

    public boolean isRecipeAcquired() {
        return recipeAcquired;
    }

    public void addIngredient(Items item) {
        ingredients.add(item);
    }

    public List<Items> getIngredients() {
        return ingredients;
    }
}

public class Seeds extends Items {
    private Season seasonGrow;
    private int daysToHarvest;

    public void grow() {
        System.out.println("Seeds are growing!");
    }

    public void harvest() {
        System.out.println("Seeds have been harvested!");
    }

    public void eat() {
        System.out.println("Seeds cannot be eaten!");
    }
}

public class Crops extends Items implements Edible {
    private int harvestAmount;
    private int energy = 3;

    public void eat() {
        System.out.println("Crops have been eaten!");
    }

    public void setHarvestAmount(int harvestAmount) {
        this.harvestAmount = harvestAmount;
    }

    public int getHarvestAmount() {
        return harvestAmount;
    }

    public void setEnergy(int energy) {
        this.energy = energy;
    }

    public int getEnergy() {
        return energy;
    }
}

public class Fish extends Items implements Edible {
    private List<String> seasonAppear; // bi
    private List<int[]> timeAppear; // se
    private List<String> locationAppear; // bi
    private List<String> weatherAppear; // bi
    private Rarity rarity; // enum
    private final int energy = 1;

    public void eat() {
        System.out.println("Fish has been eaten!");
    }

    public void setLocationAppear(List<String> locationAppear) {
        this.locationAppear = locationAppear;
    }

    public void setTimeAppear(List<int[]> timeAppear) {
        this.timeAppear = timeAppear;
    }

    public void setSeasonAppear(List<String> seasonAppear) {
        this.seasonAppear = seasonAppear;
    }

    public void setWeatherAppear(List<String> weatherAppear) {
        this.weatherAppear = weatherAppear;
    }

    public void setRarity(Rarity rarity) {
        this.rarity = rarity;
    }

    public Rarity getRarity() {
        return rarity;
    }
}
```

Items.java (sebagai Component) memiliki tipe abstract class yang akan diextend ke kelas dibawahnya, Food.java (sebagai Composite) karena sebagai turunan Items namun juga memiliki atribut ingredients yang bertipe list of Items, dan subkelas Items.java lainnya seperti: Crops.java, Seeds.java, Fish.java (sebagai Leaf) karena akan dipanggil sebagai atribut dari Food.java dan menjadi kelas yang tidak memiliki subkelas lainnya.

# Design Pattern

## Facade

### Structural Pattern

Facade adalah sebuah design pattern struktural yang menyediakan penyederhanaan interface terhadap sekumpulan kelas atau library yang kompleks.

```
public class GamePanel extends JPanel implements Runnable {
    final int originalTileSize = 16; // 16px untuk setiap pixel
    final int scale = 3; // Agar ukuran pixel menjadi 48px

    public final int tileSize = originalTileSize * scale;
    final int maxScreenCol = 16;
    final int maxScreenRow = 12;
    final int screenWidth = tileSize * maxScreenCol; // 768 pixel
    final int screenHeight = tileSize * maxScreenRow; // 576 pixel
    // final int FPS = 60;

    // World Settings
    public int maxWorldCol;
    public int maxWorldRow;
    public final int maxMap = 13; // Total jumlah map yang ada
    public int currentMap = 0;
    public final int worldWidth = tileSize * maxWorldCol;
    public final int worldHeight = tileSize * maxWorldRow;

    Timer gameClockTimer;
    Farm farm = new Farm(Weather.Sunny, day:1, Season.Spring, new Time(hour:6, minute:0), new ShippingBin());
    TileManager tileM = new TileManager(this);
    KeyHandler keyH = new KeyHandler(this);
    UI ui = new UI(this, this.farm);    UI cannot be resolved to a type
    Thread gameThread;
    public CollisionChecker collisionChecker = new CollisionChecker(this);
    public ItemManager itemManager = new ItemManager();    ItemManager cannot be resolved to a type
    public NPCManager npcManager = new NPCManager(this);    NPCManager cannot be resolved to a type
    public Player player = new Player(name:"Test", Gender.Male, energy:100, gold:1000, this, keyH);
    public Store store = new Store(this);
```

File GamePanel.java menginisiasi dan mengoordinasikan berbagai komponen kelas dan library dari sistem game ini seperti TileManager.java, KeyHandler.java, UI.java, collisionChecker.java, dan beberapa file lainnya.

# Design Pattern

## Observer

### Behavioral Pattern

Observer adalah sebuah design pattern yang mana instance dari kelas subjek akan menyimpan daftar observernya dan memberi tahu secara otomatis tentang perubahan state apa pun dengan memanggil salah satu metode dari observernya.

```
public synchronized void skipTime(int minutesToAdd, Farm farmInstance) {
    this.minute += minutesToAdd;
    while (this.minute >= 60) {
        this.minute -= 60;
        this.hour++;
        if (this.hour >= 24) {
            this.hour -= 24;
            if (farmInstance != null) {
                farmInstance.processNewDay(); // Langsung panggil metode di P
            }
        }
    }
}

public void processNewDay() {
    this.day++; // nextDay() Anda sudah melakukan ini

    // 1. Update cuaca hari ini dari ramalan kemarin
    this.weather = this.weatherForTomorrow; Weather ca
    if (this.weather == Weather.Rainy) { Weather ca
        this.rainyDaysThisSeason++;
        // TODO: Efek hujan pada FarmMap (membuat tile
        // if (this.farmMap != null) this.farmMap.setA
        System.out.println(x:"== HARI INI HUJAN! ==");
    } else {
        // if (this.farmMap != null) this.farmMap.setA
    }
}
```

Time.java sebagai subjek dan Farm.java sebagai observer. Ketika terdapat perubahan state pada Time, dalam kode kami adalah waktu melewati tengah malam, Time akan memanggil skipTime(int minutesToAdd, Farm farmInstance) yang di dalamnya terdapat processNewDay(). Hal ini menyatakan ketergantungan bahwa Farm hanya akan melakukan processNewDay() ketika mendapatkan informasi dari Time.

# Proses Pengembangan

## Pembuatan Grup Line

Pembuatan grup line dan perkenalan awal :D



16 April 2025



20 April 2025

## Meet Pertama!

Kami melakukan Google Meet untuk membaca spesifikasi bersama dan melakukan pembagian tugas

## Asistensi Pertama

Asistensi pertama kami bersama Kak Syafiq sombil melakukan pemahaman terkait spesifikasi



24 April 2025

# Proses Pengembangan

## Mengerjakan Milestone 1

Mengecek dan merevisi bagian antar anggota.



25 April 2025



27 Mei 2025

## Mengerjakan Milestone 2

Menyamakan persepsi dan menggabungkan bagian yang sudah dikerjakan masing-masing

## Asistensi Kedua

Asistensi kedua kami kemudian dilanjut dengan mengerjakan bersama.



30 Mei 2025

**enjoy the game  
and happy  
farming!**

