

User Manual

A. Game Description

Spakbor Hills adalah permainan simulasi bertani berbasis teks yang dibuat untuk membantu Dr. Asep Spakbor menjadi petani sukses demi mencegah kekacauan dunia. Pemain dapat bertani, memancing, memasak, menjual hasil panen, serta menjalin hubungan dengan NPC. Setiap aksi membutuhkan strategi, energi, dan waktu dalam game. Tujuan akhir permainan adalah mengumpulkan 17.209 gold dan menikah dengan salah satu NPC.

B. How to Install

- 1. Clone repository GitHub proyek Spakbor Hills.
- Buka folder proyek menggunakan IDE seperti VS Code.
- Compile dan jalankan file utama, Main.java, melalui terminal atau fitur "Run" di IDE.
- 4. Program akan berjalan dan menampilkan menu utama permainan.



C. Game Play

Untuk memainkan permainan *Spakbor Hills*, gunakan tombol **W**, **A**, **S**, dan **D** untuk menggerakkan karakter. Menu **Help** dapat digunakan untuk melihat panduan lengkap mengenai aksi-aksi dalam permainan. Saat memilih **New Game**, pemain akan diminta untuk memasukkan nama karakter, nama kebun, dan jenis kelamin. Sepanjang permainan, pemain dapat melakukan berbagai **aksi** sesuai dengan tombol yang tercantum di menu Help. Jika ingin mengakhiri permainan, gunakan opsi **Exit** dari menu utama.







Gameplay

Main Menu

Setelah program dijalankan, pemain akan langsung masuk ke tampilan menu utama. Pada menu ini terdapat beberapa opsi:

- New Game untuk memulai permainan baru
- · Help untuk melihat panduan aksi
- · View Player Info untuk melihat status pemain
- Statistics untuk melihat data progres
- Exit untuk keluar dari permainan



NPC List 🌘



Pemain dapat melihat atribut dan preferensi dari setiap NPC yang tersedia, seperti:

- Loved Items
- Liked Items
- Hated Items
- Status Hubungan











Daftar ini membantu pemain memilih hadiah yang tepat agar dapat meningkatkan heartPoints dan mencapai status Fiance atau Spouse.

Weather & Season Info



Cuaca dan musim sangat memengaruhi hasil pertanian dan pemancingan:

- · Weather: Sunny atau Rainy
- · Season: Spring, Summer, Fall, Winter

Setiap tanaman dan ikan hanya tersedia pada kondisi tertentu.

Pemain dapat menggunakan fitur watching untuk melihat cuaca hari itu melalui TV (jika tersedia) atau langsung dari rumah.





Gameplay

Inventory & Farm Map 📻



Setelah memilih New Game, pemain akan memasuki permainan di area Farm berukuran 32x32 tile. Inventory awal berisi item seperti:

- Hoe
- Pickaxe
- Watering Can
- Fishing Rod
- 15 Parsnip Seeds

Pemain dapat menggunakan item tersebut untuk mulai bertani. Setiap item memiliki fungsi berbeda dan hanya bisa digunakan jika Player berada di tile yang sesuai.

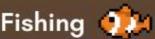
Farming Actions 🦂



Farming Actions:

- till tanah dengan Hoe
- plant benih di tanah yang dibajak
- · water tanaman agar tumbuh
- harvest hasil tanaman siap panen

Aksi-aksi ini akan mengurangi energi dan memakan waktu dalam game. Hindari kehabisan energi karena pemain akan tertidur otomatis dan kehilangan waktu hari itu.



Fishing dapat dilakukan di Pond, Ocean, Forest River, atau Mountain Lake. Pemain akan menebak angka dalam batas percobaan tertentu tergantung jenis ikan: Common (1–10), Regular (1–100), Legendary (1–500)





Gameplay

Cooking 🔑

Cooking bisa dilakukan di dalam House. Pemain butuh:

- Resep masakan
- Bahan makanan
- · Bahan bakar (firewood/coal)

Hasil makanan akan menambah energi pemain.

Milestone & End Condition 🖀



Permainan tidak memiliki akhir pasti, namun terdapat milestone yang dianggap pencapaian:

- · Pemain mencapai 17.209 Gold
- Pemain berhasil menikah dengan salah satu NPC

Jika salah satu tercapai, akan muncul halaman End Game Statistics yang menampilkan:

- Total penghasilan dan pengeluaran
- Statistik musim
- Status hubungan NPC
- Jumlah panen & ikan yang ditangkap

Class Diagram

Awal Perancangan

Player Inventory name; String - demic Mapsitum, mts pender: String energy int. + Inventory() . void Farmhame: Spring + Addition (item : ham, quantity partner; NPC rint) reciti + namove barn (bees : Ham. inventory inventory male tity. Incation Execution tint) besid equipped: Equipment + pattlemCount (tem then); att + getAlberrall: Mapetiers, inty + Player Ingma (String) : void. gotName[]: String. getGender(): String + getEnergy[]; int. disductEnergy tenergy : intl : int. Game/Model: + addfinargy() : int. + getfamhame() (String - white collection them -Barris Barris + gatPartner0 :MPC -worldWap : WorldMap + getGold(); int -String chime Aplayer Stayer + gotinientory() : inventory. -buyPrice : est -isRunning: boolean - gettecation(): Location -additionrint -gameMiew: GameViete + getEquipped() : Equipment -type: temfyee · move to kile: Tile: void + pameModel() I void 4 Remonance : String, buyPrice + numO ; word int, sellimos : int) : word -handleinpublingut, String) I void + pathtame(): String + endCame(): void + gottheyPricet) : int. + saveCarro() coold + getSellPrice() int. + hold Carred Cond d evimenfactor Action - emergy Cost; int WorldMap - Hera Cost : Local Time - required terms stern[] - locations: List-Locations - energiappayer: Player, farm: farm, args: String): void + WorldMap() : word + isExecutable(player :Player) : - gettocation (name: String) : Lembion exabitractive Deployable Object - name: String: -width sint - beight: int. Deployable Objectmana Stringth : World + gethtame(): String

+ getWidth () : int + getHeight() : int

Class Diagram

Akhir Perancangan

NPC name: String int : heartPoints loveditems; Item[] likeditems : Item[] hateditems : Item[] relationshipsStatus RelationshipStatus + NPC(name: String): void + getName() : String + getHeartPoints() : int + getLikeditems() : item[] + getLikeditems() : item[] + getHateditems() : item[] + getRelationshipStatus() : RelationshipStatus Player name : String gender: String energy: int farmName : String partner: NPC gold:int inventory : Inventory location: Location equipped: Equipment + Player (name : String) ; void + getName() ; String getGender(): String + getEnergy() : int + deductEnergy (energy : int) : int + addEnergy() : int + getFarmName() : String + getPartner(): NPC + getGold(): int + getCont(); inc + getInventory(); Inventory + getEquipped(); Equipment + moveTo (tile; Tile); void <<interface>> Action energyCost: int timeCost : LocalTime requireditems:item[] + execute(player: Player, Farm: Farm, args : String) : void + isExecutable(player : Player)

Inventory items: Map<Item, int> GameModel + Inventory(): void + additem (item : Item, quantity farm : Farm worldMap : WorldMap player : Player + removeltem (item : Item. isRunning : boolean gameView : GameView + getItemCount (Item : Item) : int + getAlltems(): Map<Item, int> + gameModel(): void + run(): void + run() : Void - handleInput(Input: String) : void + endGame() : void + saveGame() : void + loadGame() : void <<abstract>>> Item <<enum>> Weather - buvPrice : int SUNNY Farm -type: ItemType name : String player : Player + Item(name: String, buyPrice + currentWeather Weather = SUNNY : int, sellPrice : int) ; void + getName() : String farmMap : FarmMap time : LocalTime + getBuyPrice(): int + getSellPrice(): int day : int season : Season + setSeason(season: Season): weather : Weather house : House + getSeason() : Season shippingBin : ShippingBin + Farm(name : String) : void <<enum>> Season + update() : void + getPlayer() : Player + getFarmMap() : FarmMap SUMMER + getTime() : LocalTime + getDay() : int WINTER + getSeason(): Season + getWeather() : Weather + currentSeason : Season = + nextDay : String + updateSeason() : String + updateWeather(): String + setSeason(season : Season) : + getSeason(): Season Tile -x:int crop: Crops WorldMap + Tile (x: int, y: int) : void + getX(): int + getY(): int - locations : List<Location> + getType(): TileType + setType (type: TileType) + WorldMap(): void : void + getCrop() : Crops getLocation (name: String) Location - name : String + Location(name: String) <<abstract>> DeployableObject Full Version: name: String

+ DeployableObject(name : Stringt)

getName(): StringgetWidth (): intgetHeight(): int



https://drive.google.com/file/

d/1ERCwaxJIrTLuz8zqHMPfjT

ndN4m32DN5/view

Design Pattern

Command Pattern

Letak: src/action/

Penjelasan: Semua aksi dalam game (seperti Chatting, Fishing, Eating, dll.) mengimplementasikan antarmuka Action dengan metode execute().

Fungsi: Memisahkan logika aksi dari pemicu aksi (misalnya dari input keyboard), sehingga fleksibel dan modular.

Renderer/Manager Pattern

Letak: src/render/ObjectRenderer.java

Penjelasan: Kelas ini bertanggung jawab untuk mengelola dan menggambar seluruh objek game.

Fungsi: Mengatur lifecycle dari objekobjek yang tampil di layar, meski tidak mengikuti pola Composite atau Iterator secara eksplisit.

Abstract Base Class Pattern

Letak: src/Objects/GameObject.java

Penjelasan: GameObject adalah kelas abstrak untuk semua objek dalam permainan.

Fungsi: Menyediakan kerangka dasar untuk pewarisan objek seperti rumah, kolam, dan lain-lain.





Design Pattern

Event Handling

Letak: src/inputs/KeyboardListener.java

Penjelasan: Menangani input keyboard dan men-trigger aksi sesuai tombol yang ditekan.

Fungsi: Menghubungkan input pemain dengan aksi melalui instansiasi objek Action, mendukung Command Pattern.



Perubahan

Class Diagram

Pada tahap perancangan awal, class diagram dibuat berdasarkan rencana fitur dan struktur game *Spakbor Hills* secara konseptual. Namun, selama proses implementasi, terdapat beberapa penyesuaian terhadap *class diagram* awal karena kebutuhan teknis, efisiensi struktur kode, dan hasil diskusi kami saat development berlangsung.

1. Penambahan Interface Action dan Implementasi Pola Command

Awalnya: Aksi-aksi pemain dirancang sebagai metode langsung dalam kelas Player.

Perubahan: Dibuat interface Action dan setiap aksi seperti Chatting, Fishing, dan Sleeping diimplementasikan sebagai kelas tersendiri.

Alasan: Untuk menerapkan Command Pattern agar struktur kode lebih modular, memudahkan penambahan aksi baru tanpa mengubah kode lama, dan membuat pemrosesan input lebih fleksibel.

2. Penyesuaian Atribut dan Metode di Player dan Inventory

Awalnya: Beberapa atribut seperti energy, favoriteItem, dan partner belum dirancang secara detail.

Perubahan: Ditambahkan selama implementasi berdasarkan pengembangan fitur interaksi dan relasi karakter.

Alasan: Untuk mendukung gameplay yang lebih kompleks, seperti fitur hubungan dengan NPC dan pengelolaan stamina.

3. Penghapusan atau Penyederhanaan Class

Awalnya: Beberapa class sempat dirancang namun tidak jadi digunakan (misalnya, class FarmObjectManager).

Perubahan: Fungsinya digabungkan ke kelas lain seperti FarmMap atau ObjectRenderer.

Alasan: Untuk menyederhanakan arsitektur dan menghindari redundansi.





18223113 - Aldoy Fauzan Avanza

Coding:

- Membuat GitHub Repository
- Membuat Game Window, Player, Keyboard, Camera, dan Mouse
- Membuat New Game (Inisiasi Game State)
- Membuat Save and Load (Menu Game), GUI sesuai pembagian tugas, Keyboard dan Mouse Input
- · Membuat inisiasi kode awal
- Membuat Action (plant, watering, tilling, harvest)

Dokumen:

Membantu Class Diagram



18223121 - Fudhail Fayyadh

Coding:

- Membuat Menu Game & Ending Condition (Statistics)
- Membuat WorldMap dan Methodnya
- Membuat GUI sesuai pembagian tugas
- Membuat Map(Ocean, Lake, River, Village, House, Store)

Dokumen:

Membantu Class Diagram





Log Activity

18223123 - Harfhan Ikhtiar Ahmad Ridzky

Coding:

- Membuat House & Shipping Bin (Bonus: Furnitures (Bed buat sleep))
- Membuat NPC (Bonus: Easter Egg, Free Market, Let's Go Gambling)
- · Membuat inventory dan methodnya
- Membuat action NPC

Dokumen:

Membantu Class Diagram



18223131 - Aulia Azka Azzahra

Coding:

- Membuat Farm Action
- · Membuat Items & Fishing
- · Membuat GUI sesuai pembagian tugas

Dokumen:

- · Membuat Booklet
- Membuat dan mencari asset
- Membuat class diagram



18223138 - Sonya Putri Fadilah

Coding:

- · Membuat Object (FarmMap, Collision)
- Membuat Inventory: open inventory, show time, dan show location
- Membuat GUI sesuai pembagian tugas

Dokumen:

- Membuat booklet
- · Membuat class diagram
- Membuatt dan mencari asset





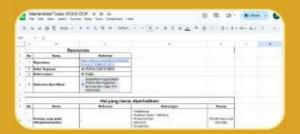
Development Process - M1

Meet-Up 1



Pada **pertemuan pertama**, kami membahas keseluruhan rencana tugas besar, termasuk pembuatan *mastersheet*, perancangan *class diagram*, serta pembagian tugas untuk milestone pertama (M1).

Mastersheet



Kami membuat mastersheet yang berisi **seluruh kebutuhan** terpusat tugas besar.

Inisialisasi GitHub



Pembuatan repo dilakukan pada **21 April 2025**

Asistensi 1



Asistensi pertama melalui **Google Meet** membahas seputar Q&A dan *class diagram*.

Membuat Class Diagram



Kami membuat *class diagram* di draw.io



Development Process - M2

Meet-Up 2



Pada **pertemuan kedua**, kami mulai meng-coding dan terus beproses sampai menjelang hari H *deadline*. Walaupun banyak ngestucknya.

Asistensi 2



Asistensi H-2 pengumpulan Tubes dan masih banyak yang harus di kejar

Meet-Up 3



H-1 deadline . Meet-Up 3 kita ga sempet foto bareng :(

Meet-Up 4



H-Jam *dealine*. Masih sibuk pull and push:)

Finally



Akhirnya jadi walaupun

