

IF2010 Pemrograman Berorientasi Objek

TUGAS BESAR MILESTONE 2

NimonsCooked

Dibuat oleh:

| | |
|------------------------------|----------|
| Irgi Satya Priangga | 18224041 |
| Adrian Akhdan Assyauqi | 18224055 |
| Muhammad Azikra Wira Pratama | 18224089 |
| Sirojul Firdaus | 18224095 |

PENDAHULUAN

Nimons Cooked adalah game simulasi memasak interaktif yang ditulis dalam bahasa Java menggunakan sistem build Gradle. Game ini dikembangkan sebagai Tugas Besar mata kuliah Object-Oriented Programming (IF2212).

Dalam permainan ini, pemain berperan sebagai chef sushi yang bertugas mengelola dapur, memproses bahan mentah, memasak sesuai resep, serta menyajikan hidangan kepada pelanggan dalam waktu tertentu. Game menantang pemain untuk berpikir strategis demi meraih skor tertinggi.

Game ini memiliki tiga tingkat kesulitan:

- Easy: Unlimited lives
- Normal: 3 lives
- Hardcore: 1 life

Lives dalam konteks game ini menunjukkan berapa kali pemain dapat melakukan kesalahan dalam menyajikan pesanan sebelum permainan berakhir.

User Manual dan Gameplay

1. Cara Menjalankan Game

- Pastikan Java dan Gradle sudah terinstall.
- Clone repository game:
git clone <https://github.com/Awp-confirm/Tubes-OOP-K01-L>
- Buka proyek di IDE (IntelliJ, Eclipse, VS Code).
- Jalankan perintah di terminat:
./gradlew run

2. Kontrol Dasar

| Tombol | Fungsi |
|---------------|---------------------------------|
| W / A / S / D | Bergerak |
| SHIFT | Dash/Sprint |
| SPACE | Aksi / Interaksi dengan station |
| F | Switch antara Chef 1 & Chef 2 |
| Q | Melempar item |
| ESC | Pause / menu |

3. Tujuan Permainan

Pemain harus menyelesaikan pesanan yang muncul selama durasi permainan. Setiap pesanan memiliki batas waktu dan target skor.

Pesanan yang berhasil disajikan sesuai resep akan menambah skor. Kesalahan menyajikan akan mengurangi lives.

4. Gameplay

Pemain berperan sebagai chef dalam dapur sushi. Setiap permainan memiliki durasi waktu tertentu (mis. 5–10 menit). Pesanan akan muncul terus selama game.

5. Alur Permainan

a. Main Menu

Pemain memilih tingkat kesulitan (Easy, Normal, Hardcore).

b. Pesanan Muncul

Di sisi kiri layar muncul order yang harus dipenuhi.

c. Ambil Bahan

Chef mengambil bahan di Ingredient Storage.

d. Proses Bahan

- Potong di Cutting Station
- Masak di Cooking Station

e. Rakitan Hidangan

Rakit semua bahan pada plate sesuai resep di Assembly Station.

f. Melayani Hidangan

Sajikan di Serving Station agar skor bertambah.

g. Cuci Piring

Plate kotor otomatis berpindah di Plate Storage dan dibersihkan di Washing Station jika dibutuhkan.

h. Waktu Habis / Lives Habis

Game berakhir dan skor akhir ditampilkan.

6. Struktur Objek Game

a. Chef Player

Entitas yang dikendalikan pemain, terdiri dari dua chef yang bisa di-switch. Chef memiliki satu slot inventori dan dapat melakukan beberapa aksi seperti pick up, use station, drop item.

b. Item

Item merupakan semua objek yang bisa diambil dan diproses:

- Ingredient
- Dish
- Kitchen Utensils

Item merupakan superclass abstrak.

Ingredient

Item yang diproses menjadi bahan masakan.

State: RAW, CHOPPED, COOKING, COOKED, BURNED.

Dish

Hidangan akhir hasil kombinasi ingredient.

Kitchen Utensils

Peralatan untuk memasak atau menyajikan:

- Plate
- Boiling Pot
- Frying Pan

Masing-masing memiliki daftar contents (set ingredient).

c. Stations (Lokasi Kerja)

| Station | Fungsi |
|---------------------|------------------------|
| Cutting Station (C) | Memotong bahan |
| Cooking Station (R) | Memasak dengan utensil |



| | |
|------------------------|----------------------|
| Assembly Station (A) | Rakitan hidangan |
| Serving Counter (S) | Sajikan dish |
| Washing Station (W) | Cuci plate |
| Ingredient Storage (I) | Ambil bahan mentah |
| Plate Storage (P) | Ambil / simpan plate |
| Trash Station (T) | Buang bahan / item |

d. Order

Setiap order memiliki:

- Posisi (urutan aktif)
- Recipe (resep)
- Time Limit (bar)

e. Recipe

Dekomposisi bahan untuk dish tertentu

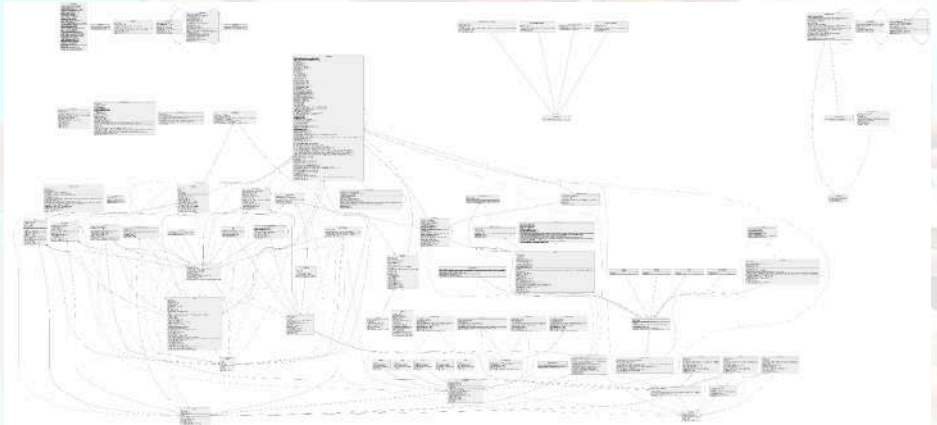
- Kappa Maki: Nori (Raw) + Nasi (Cooked) + Timun (Chopped)
- Sakana Maki: Nori (Raw) + Nasi (Cooked) + Ikan (Chopped)
- Ebi Maki: Nori (Raw) + Nasi (Cooked) + Udang (Cooked)
- Fish Cucumber Roll: Nori (Raw) + Nasi (Cooked) + Ikan (Chopped) + Timun (Chopped)

Contributor

| Nama | NIM | Tugas |
|------------------------|----------|--|
| Irgi Satya Priangga | 18224041 | <ol style="list-style-type: none">1. Membuat model tile:<ul style="list-style-type: none">- walkable- wall- station tile- spawn2. Mengubah map menjadi:<ul style="list-style-type: none">- koordinat 2D- daftar tile- rule interaksi per tile3. Membuat semua aset desain untuk GUI4. Karena full desain dari awal, melakukan finishing dan pengecekan semua kode |
| Adrian Akhdan Assyauqi | 18224055 | <ol style="list-style-type: none">1. Mendokumentasikan game flow:<ul style="list-style-type: none">- munculnya order- validasi order- scoring & penalty- syarat Stage Over2. Mendokumentasikan menu sistem:<ul style="list-style-type: none">- Main Menu- Stage Select- Result Screen3. Membuat sequence diagram untuk:<ul style="list-style-type: none">- menyelesaikan order- navigasi menu- flow start hingga stage selesai4. Membuat UML class diagram untuk:<ul style="list-style-type: none">- Entitas inti: Chef, Item, Ingredient, Dish, KitchenUtensils- Hierarki Station- Struktur Map dan Tile- Kelas Order dan Recipe |
| Muhammad Azikra Wira | 18224089 | <ol style="list-style-type: none">1. Mendokumentasikan struktur Chef Player:<ul style="list-style-type: none">- atribut (id, posisi, arah hadap, inventory) |

| | | |
|-----------------|----------|--|
| Pratama | | <ul style="list-style-type: none">- perilaku (gerak, pick/drop, ganti Chef, aturan inventory) <ol style="list-style-type: none">2. Menyusun struktur hierarki Item:<ul style="list-style-type: none">- abstract Item- Ingredient (RAW, CHOPPED, COOKED)- Dish- Kitchen Utensils3. Mendefinisikan interface:<ul style="list-style-type: none">- Preparable- CookingDevice4. Membuat UML class diagram untuk:<ul style="list-style-type: none">- Entitas inti: Chef, Item, Ingredient, Dish, KitchenUtensils- Hierarki Station- Struktur Map dan Tile- Kelas Order dan Recipe |
| Sirojul Firdaus | 18224095 | <ol style="list-style-type: none">1. Mendokumentasikan tiap station:<ul style="list-style-type: none">- Cutting- Cooking- Assembly- Washing- Serving- Ingredient Storage- Plate Storage- Trash2. Menuliskan aturan interaksi:<ul style="list-style-type: none">- trigger- input- output- kondisi error3. Menyesuaikan interaksi dengan Item dan Chef. |

Class Diagram

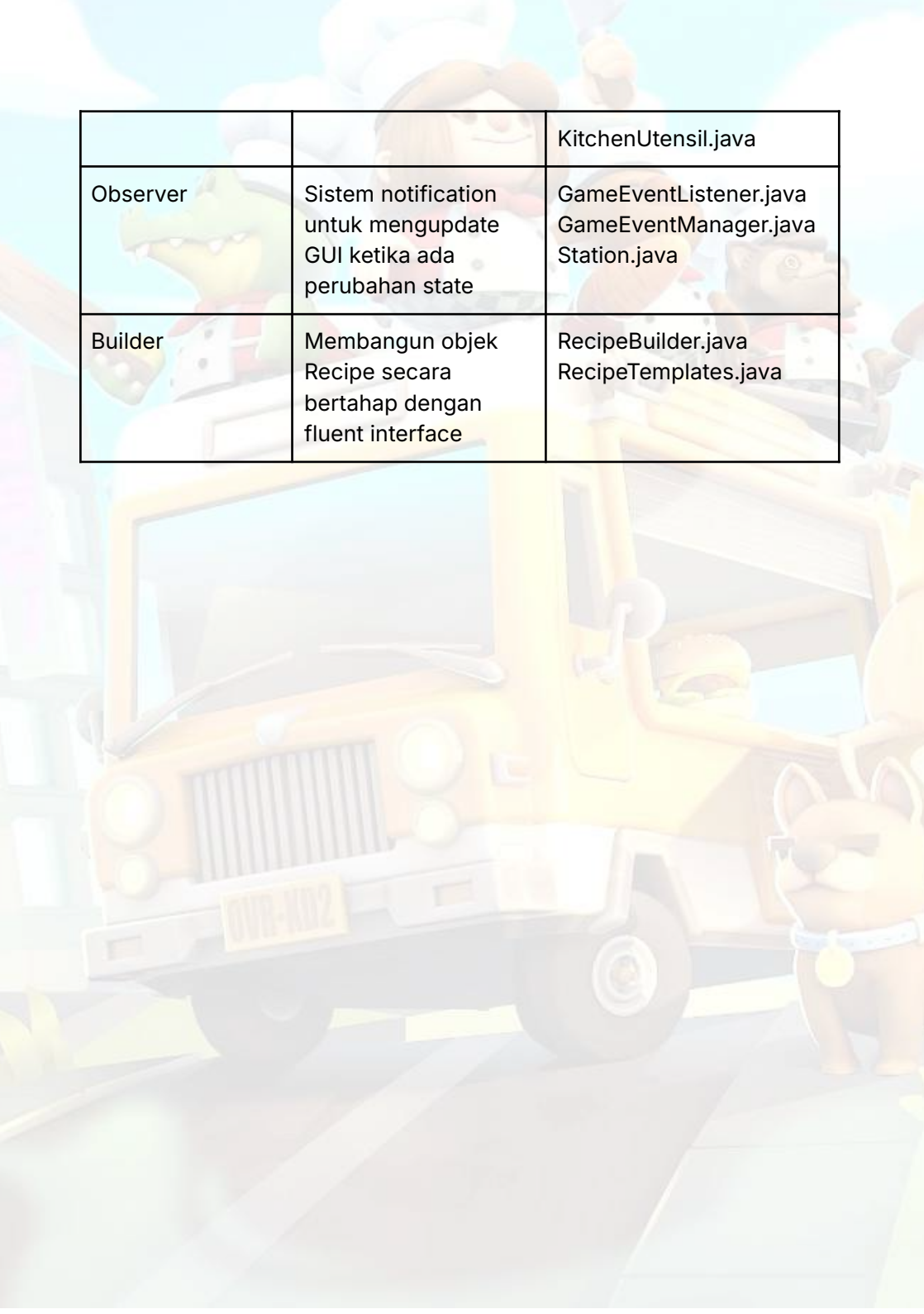


Jika ingin lebih jelas:

<https://drive.google.com/file/d/1ODjwcsrC5wTmNu3Xis3wxqZjXEk2jJoU/view?usp=sharing>

Design Pattern

| Design Pattern | Tujuan | Implementasi |
|-----------------|--|---|
| Singleton | Memastikan hanya ada satu instance dari class tertentu dan menyediakan akses global ke instance tersebut | GameSettings.java GameLogManager.java PlateStorageStation.java GameScreen.java OrderManager.java GameTaskExecutor.java SoundManager.java StageProgress.java |
| Factory Method | Membuat objek stasiun berdasarkan karakter dari map file tanpa perlu specify class exact | MapLoader.java |
| Strategy | Mendefinisikan family of algorithms dan membuat mereka interchangeable | CookingDevice.java FryingPan.java BoilingPot.java Oven.java |
| Template Method | Mendefinisikan kerangka algoritma di superclass, membiarkan subclass override langkah-langkah tertentu | Station.java CuttingStation.java CookingStation.java ServingStation.java AssemblyStation.java WashingStation.java TrashStation.java PlateStorageStation.java IngredientStorageStation. java Item.java |


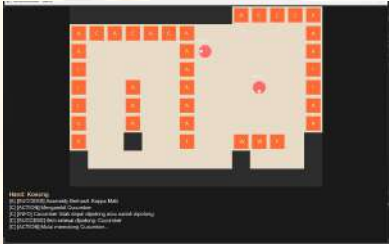
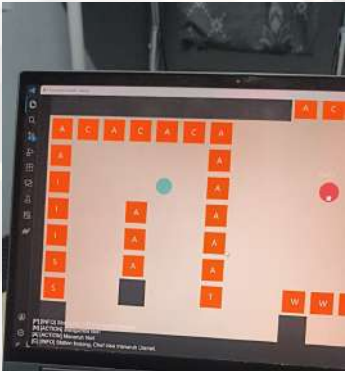


| | | |
|----------|---|---|
| | | KitchenUtensil.java |
| Observer | Sistem notification untuk mengupdate GUI ketika ada perubahan state | GameEventListener.java GameEventManager.java Station.java |
| Builder | Membangun objek Recipe secara bertahap dengan fluent interface | RecipeBuilder.java RecipeTemplates.java |

Proses Pengembangan

1. Proses Pembuatan Station

Proses pengembangan station pada game ini dilakukan secara bertahap. Pada tahap awal, station dikembangkan dan diuji tanpa menggunakan map untuk memastikan logika dasar dan alur interaksi sudah berjalan dengan benar. Setelah itu, sistem diintegrasikan ke dalam map dengan dua chef untuk menguji interaksi secara bersamaan serta koordinasi antar pemain.

| Tanpa Map | 2 Chef |
|---|--|
|  |  <div data-bbox="622 879 776 935"><p>Head Kowang 1) [ACTION] Move Chef and Kowang Map 2) [ACTION] Move Chef and Kowang Map 3) [ACTION] Move Chef and Kowang Map 4) [ACTION] Move Chef and Kowang Map 5) [ACTION] Move Chef and Kowang Map 6) [ACTION] Move Chef and Kowang Map</p></div> |
| 1 Chef | |
|  <div data-bbox="434 1374 588 1430"><p>ptmrcia [ACTION] Move Chef and Kowang Map ptmrcia [ACTION] Move Chef and Kowang Map ptmrcia [ACTION] Move Chef and Kowang Map ptmrcia [ACTION] Move Chef and Kowang Map ptmrcia [ACTION] Move Chef and Kowang Map ptmrcia [ACTION] Move Chef and Kowang Map</p></div> | |

2. Pembuatan Sistem Order

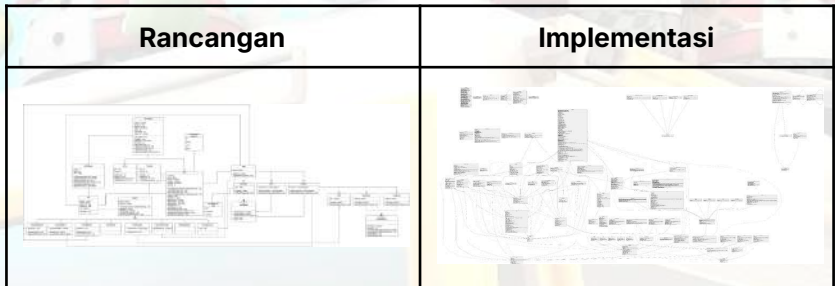
Pengembangan sistem order pada game ini diawali dengan pembuatan desain draft untuk menentukan alur pemesanan, tampilan antarmuka, serta hubungan antara order dan gameplay. Pada tahap ini, kemunculan order dan kebutuhan bahan dapat ditampilkan dengan jelas. Setelah desain dasar terbentuk, sistem kemudian dikembangkan lebih lanjut hingga mendekati versi final dengan integrasi penuh ke dalam map dan mekanik permainan.

| Desain Sistem Order Draft | Sistem Order yang Hampir Selesai |
|---|---|
|  |  |

3. Rancangan vs Implementasi

Pada tahap perbandingan antara rancangan awal dan implementasi, dilakukan banyak perubahan pada class diagram karena kebutuhan sistem yang berkembang seiring proses pengembangan game. Rancangan awal dibuat sebagai gambaran konseptual, namun ketika mulai diimplementasikan, ditemukan bahwa beberapa kelas perlu dipecah, digabung, atau ditambahkan agar lebih sesuai dengan mekanik permainan yang sebenarnya. Perubahan ini dilakukan untuk mengakomodasi fitur baru seperti sistem order, interaksi antar station, serta dukungan lebih dari satu chef, yang belum

sepenuhnya terdefinisi pada tahap perancangan awal. Selain itu, penyesuaian juga diperlukan untuk mengurangi ketergantungan antar kelas, dan mempermudah pengelolaan state permainan, sehingga struktur kode menjadi lebih fleksibel, mudah dikembangkan, dan lebih mencerminkan kebutuhan implementasi nyata.



EXTRAS



AJDIAJSDJALSDJK - Zikra



..... - Adrian