A...

(partial class, right side cut off)

priceInUAH: d

massInGrams

optionID: int

ingredients: Li

condition: Cor

timeForCookir

specifications:

Extends ────────────────── Extends ──────────

AbstractSalad
saladDressing: String

AbstractSoup
consistence: String

Extends ─── Extends

HerringSalad
- persentageOfSalt: int

GreekSalad
- freshnessOfVegs: String

MushroomSoup
- mushroomType: String

PumpkinSoup
- countryOfOrigin: String

<<enum>>
DegreeOfMeatDoneness

+ RARE

+ MEDIUM_RARE

+ MEDIUM

+ MEDIUM_DONE

+ WELL_DONE

Use

Goulash
- degreeOfMeatDoneness: DegreeOfMeatDonene
- pungencyLevel: String

Launcher

Launcher

+ main(args: String[]): void

Use

AbstractDish

...ouble
...: int
...st<Ingredient>
...ndition
...ngInMinutes: int
... List<String>

Use

<<enum>> Condition

+ HOT
+ COLD

<<enum>> SortType

+ ASCENDING
+ DESCENDING

Use

RestorationManager

- menu: list<AbstractDish>
- tables: list<Table>
+ findTable(group: GroupOfClients): boolean
+ findAppropriateDishes(group: GroupOfClients): boolean
+ findVegetarianDishes(): void
+ findBabyDishes(): void
+ findNonAllergicDishes(): void
+ findDiabeticDishes(): void
+ sortByPrice(sortType: SortType, listOfDishes: List<Abst...
+ sortByMass(sortType: SortType, listOfDishes: List<Abst...
+ takeOrder(group: GroupOfClients): boolean
+ giveOrderToClient(group: GroupOfClients): boolean
+ giveReceiptToClient(group: GroupOfClients): boolean
+ cleanTheTable(): boolean

AbstractMainCourse

sideDish: String
massOfSideDishInGrams: int
meat: String
massOfMeatInGrams: int
sauce: String
massOfSauceInGrams: int

Extends

AbstractDessert

persentageOfSugar: int

Extends

Pancake

- syrupType: String

Extends

Muffin

- filling: String

Extends

Lasagna

- recipeUsed: String

Extends

MashedPotato

- potatoType: type

Extends

...ess

DishMaker

+ field: type

+ fulfillOrder(order: Order): boolean

+ cookPancakes(dishCounter: int): list<AbstractDish>

+ cookMuffins(dishCounter: int): list<AbstractDish>

+ cookGreekSalads(dishCounter: int): list<AbstractDish>

+ cookHerringSalads(dishCounter: int): list<AbstractDish>

+ cookGoulashes(dishCounter: int): list<AbstractDish>

+ cookLasagnas(dishCounter: int): list<AbstractDish>

+ cookPumpkinSoups(dishCounter: int): list<AbstractDish>

+ cookMushroomSoups(dishCounter: int): list<AbstractDish>

+ cookMashedPotato(dishCounter: int): list<AbstractDish>

---- Use ----

Order

- dishes: list<AbstractDish>

- numberOfEachDish: list<int>

- groupName: GroupOfClients

Use

Table

- numberOfSeats: int

- isFree: boolean

- isClean: boolean

---- Use ----

Use

----- (left partial class) -----

...ractDish>): void

...ractDish>): void

GroupOfClients

- numberOfPeople: int

- preferences: List<String>

- order:Order

- table: Table

+ getMenu(): void

+ makeOrder(list<AbstractDish>): Order

+ getDetailsAboutDish(dish:AbstractDish): void

+ enjoyMeal(): boolean

+ payTheReceipt(): boolean

+ leave(): boolean