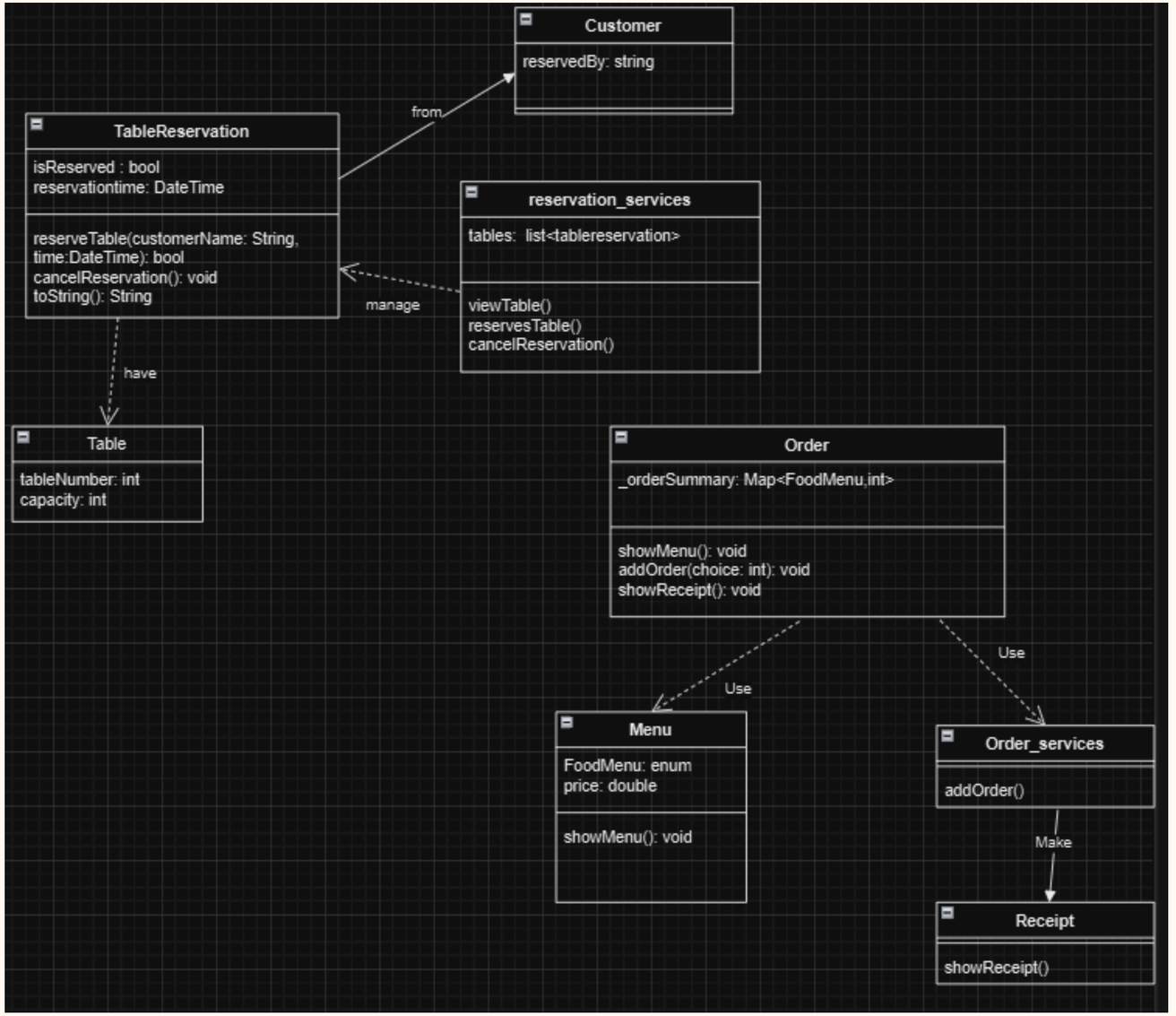
mini Project

Say Sakphearith

Content

- UML
- Demo Code

UML


Customer
reservedBy: string
from
TableReservation
isReserved : bool
reservationtime: DateTime
reserveTable(customerName: String, time:DateTime): bool
cancelReservation(): void
toString(): String
reservation_services
tables: list<tablereservation>
viewTable()
reservesTable()
cancelReservation()
manage
have
Table
tableNumber: int
capacity: int
Order
_orderSummary: Map<FoodMenu,int>
showMenu(): void
addOrder(choice: int): void
showReceipt(): void
Use
Use
Menu
FoodMenu: enum
price: double
showMenu(): void
Order_services
addOrder()
Make
Receipt
showReceipt()

FoodMenu

Resturant > Models > menu.dart > ...

```dart
enum FoodMenu {
  Pizza(8.99),
  Burger(5.49),
  Pasta(7.99),
  Sushi(12.49),
  Salad(4.99);

  final double price;
  const FoodMenu(this.price);
}
```

MenuServices

```dart
import '../Models/menu.dart';

class MenuServices {
  void showMenu() {
    print("Menu:");
    for (var i = 0; i < FoodMenu.values.length; i++) {
      print(
          "${i + 1}. ${FoodMenu.values[i].name} - \$${FoodMenu.values[i].price.toStringAsFixed(2)}");
    }
  }
}
```

Order class

```dart
import 'menu.dart';

class OrderFood {
  final Map<FoodMenu, int> _orderSummary = {};

  Map<FoodMenu, int> get orderSummary => _orderSummary;

  void addToOrder(FoodMenu item) {
    _orderSummary.update(item, (qty) => qty + 1, ifAbsent: () => 1);
  }

  Map<FoodMenu, int> getOrderSummary() => _orderSummary;

  double calculateTotal() {
    return _orderSummary.entries
        .map((entry) => entry.key.price * entry.value)
        .fold(0.0, (prev, curr) => prev + curr);
  }
}
```

orderservices

```dart
import '../Models/menu.dart';
import '../Models/order.dart';

class OrderServices {
  final OrderFood _orderFood;

  OrderServices(this._orderFood);

  void addOrder(int choice) {
    if (choice > 0 && choice <= FoodMenu.values.length) {
      var selectedFood = FoodMenu.values[choice - 1];
      _orderFood.addToOrder(selectedFood);
      print("Added ${selectedFood.name} to your order.");
    } else {
      print("Invalid choice. Please try again.");
    }
  }

  void showReceipt() {
    final orderSummary = _orderFood.getOrderSummary();

    if (orderSummary.isEmpty) {
      print("\nNo items ordered.");
      return;
    }

    print("\nOrder Summary:");
    orderSummary.forEach((item, quantity) {
      double itemTotal = item.price * quantity;
      print("${item.name} x $quantity = \$${itemTotal.toStringAsFixed(2)}");
    });

    print("\nTotal: \$${_orderFood.calculateTotal().toStringAsFixed(2)}");
  }
}
```

Reservation class

```dart
class TableReservation {
  final int tableNumber;
  final int capacity;
  bool isReserved = false;
  DateTime? reservationTime;
  String? reservedBy;

  TableReservation(this.tableNumber, this.capacity);

  bool reserveTable(String customerName, DateTime time) {
    if (isReserved) {
      print("Table $tableNumber is already reserved.");
      return false;
    } else {
      isReserved = true;
      reservedBy = customerName;
      reservationTime = time;
      print("Table $tableNumber reserved by $customerName at $time.");
      return true;
    }
  }

  void cancelReservation() {
    if (!isReserved) {
      print("Table $tableNumber is not currently reserved.");
    } else {
      print(
          "Reservation for table $tableNumber by $reservedBy has been cancelled.");
      isReserved = false;
      reservedBy = null;
      reservationTime = null;
    }
  }

  @override
  String toString() {
    if (isReserved) {
      return "Table $tableNumber is reserved by $reservedBy for $reservationTime.";
    } else {
      return "Table $tableNumber is available.";
    }
  }
}
```

```dart
import 'dart:io';
import '../Models/table_reserve.dart';

class ReservationService {
  final List<TableReservation> _tables;

  ReservationService(this._tables);

  void showMenu() {
    print("\n--- Table Reservation ---");
    print("1. View all tables");
    print("2. Reserve a table");
    print("3. Cancel a reservation");
    print("4. Exit");
    stdout.write("Choose an option: ");
  }

  void viewTables() {
    for (var table in _tables) {
      print(table);
    }
  }

  void reserveTable() {
    stdout.write("Enter table number to reserve: ");
    int? tableNumber = int.tryParse(stdin.readLineSync() ?? '');

    if (tableNumber == null ||
        tableNumber < 1 ||
        tableNumber > _tables.length) {
      print("Invalid table number.");
      return;
    }

    stdout.write("Enter your name: ");
    String? name = stdin.readLineSync();

    if (name == null || name.isEmpty) {
      print("Invalid name.");
      return;
    }

    stdout.write(
        "Enter reservation hour from now (e.g., 1 for 1 hour from now): ");
    int? hours = int.tryParse(stdin.readLineSync() ?? '');
```

```dart
    if (hours == null || hours < 0) {
      print("Invalid hour.");
      return;
    }

    bool reserved = _tables[tableNumber - 1]
        .reserveTable(name, DateTime.now().add(Duration(h

    if (!reserved) {
      print("Failed to reserve table.");
    }
  }

  void cancelReservation() {
    stdout.write("Enter table number to cancel reservatio
    int? tableNumber = int.tryParse(stdin.readLineSync()

    if (tableNumber == null ||
        tableNumber < 1 ||
        tableNumber > _tables.length) {
      print("Invalid table number.");
      return;
    }

    _tables[tableNumber - 1].cancelReservation();
  }
}
```

Reservation Services

```dart
import 'dart:io';
import '../Models/table_reserve.dart';

class ReservationService {
  final List<TableReservation> _tables;

  ReservationService(this._tables);

  void showMenu() {
    print("\n--- Table Reservation ---");
    print("1. View all tables");
    print("2. Reserve a table");
    print("3. Cancel a reservation");
    print("4. Exit");
    stdout.write("Choose an option: ");
  }

  void viewTables() {
    for (var table in _tables) {
      print(table);
    }
  }

  void reserveTable() {
    stdout.write("Enter table number to reserve: ");
    int? tableNumber = int.tryParse(stdin.readLineSync() ?? '');

    if (tableNumber == null ||
        tableNumber < 1 ||
        tableNumber > _tables.length) {
      print("Invalid table number.");
      return;
    }

    stdout.write("Enter your name: ");
    String? name = stdin.readLineSync();

    if (name == null || name.isEmpty) {
      print("Invalid name.");
      return;
    }

    stdout.write(
        "Enter reservation hour from now (e.g., 1 for 1 hour from now): ");
    int? hours = int.tryParse(stdin.readLineSync() ?? '');

    if (hours == null || hours < 0) {
      print("Invalid hour.");
      return;
    }

    bool reserved = _tables[tableNumber - 1]
        .reserveTable(name, DateTime.now().add(Duration(hours: hours)));

    if (!reserved) {
      print("Failed to reserve table.");
    }
  }

  void cancelReservation() {
    stdout.write("Enter table number to cancel reservation: ");
    int? tableNumber = int.tryParse(stdin.readLineSync() ?? '');

    if (tableNumber == null ||
        tableNumber < 1 ||
        tableNumber > _tables.length) {
      print("Invalid table number.");
      return;
    }

    _tables[tableNumber - 1].cancelReservation();
  }
}
```

Demo Time