
RESTAURANT
MANAGEMENT
SYSTEM

GROUP MEMBER

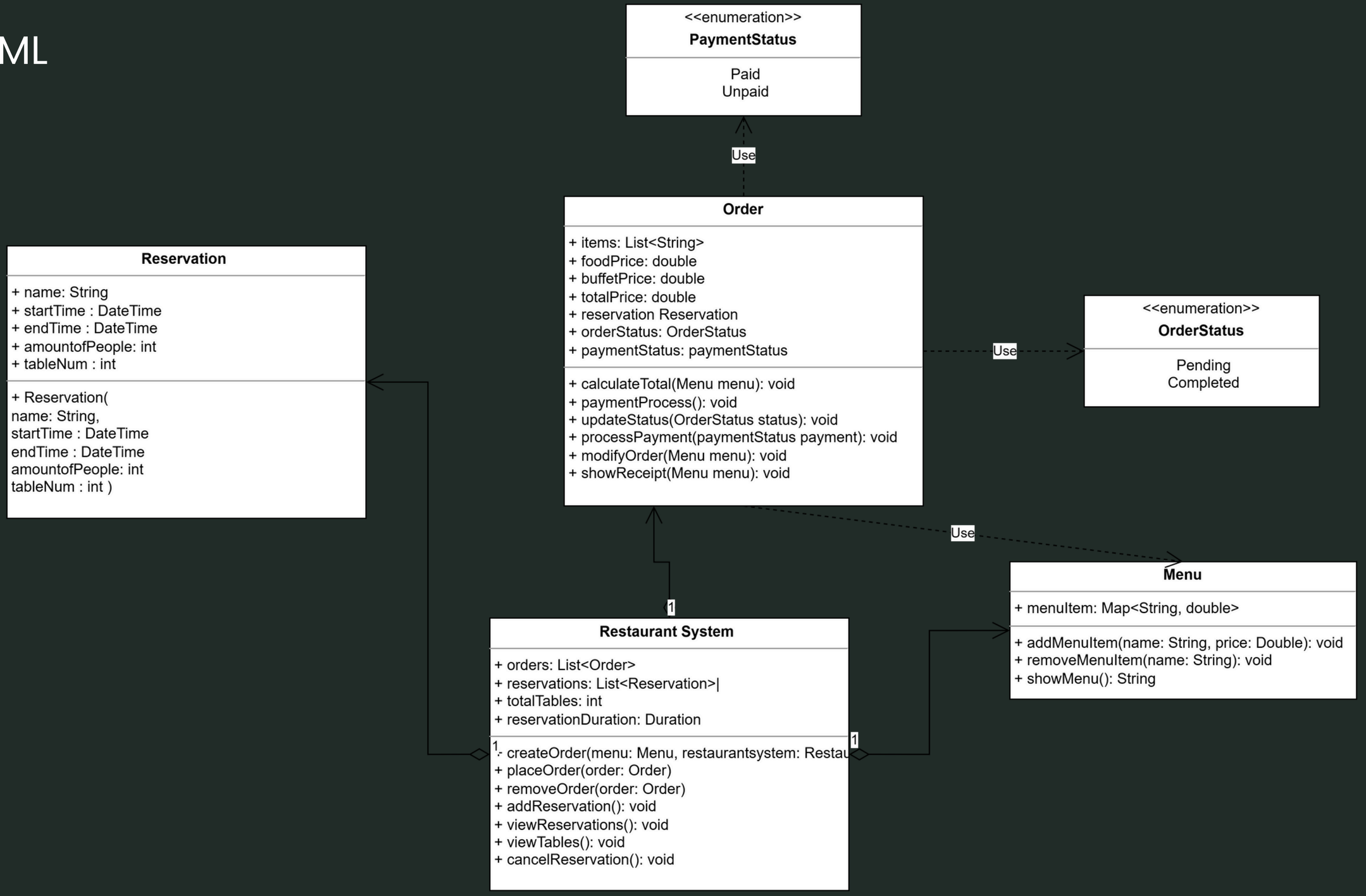
OEM PIMOL

Manage on Customer information, Reservation creation and viewing and Checking table availability

NON SORANY

Manage on Order creation, Menu management, Adding/removing items from orders and Finalizing orders and displaying receipts

UML


<<enumeration>>
PaymentStatus
Paid
Unpaid
Use
Order
+ items: List<String>
+ foodPrice: double
+ buffetPrice: double
+ totalPrice: double
+ reservation Reservation
+ orderStatus: OrderStatus
+ paymentStatus: paymentStatus
+ calculateTotal(Menu menu): void
+ paymentProcess(): void
+ updateStatus(OrderStatus status): void
+ processPayment(paymentStatus payment): void
+ modifyOrder(Menu menu): void
+ showReceipt(Menu menu): void
Use
<<enumeration>>
OrderStatus
Pending
Completed
Reservation
+ name: String
+ startTime : DateTime
+ endTime : DateTime
+ amountofPeople: int
+ tableNum : int
+ Reservation(
name: String,
startTime : DateTime
endTime : DateTime
amountofPeople: int
tableNum : int)
Use
1
Restaurant System
+ orders: List<Order>
+ reservations: List<Reservation>|
+ totalTables: int
+ reservationDuration: Duration
1
- createOrder(menu: Menu, restaurantsystem: Restau
+ placeOrder(order: Order)
+ removeOrder(order: Order)
+ addReservation(): void
+ viewReservations(): void
+ viewTables(): void
+ cancelReservation(): void
1
Menu
+ menuItem: Map<String, double>
+ addMenuItem(name: String, price: Double): void
+ removeMenuItem(name: String): void
+ showMenu(): String

CLASS MENU

```dart
// Class for Menu Item
class Menu {
  Map<String, double> menuItems = {};

  void addMenuItem(String name, double price) {
    menuItems[name] = price;
  }

  void removeMenuItem(String name) {
    menuItems.remove(name);
  }

  double? getPrice(String name) {
    return menuItems[name];
  }

  void showMenu() {
    print('Menu');
    menuItems.forEach((name, price) {
      print('Item: $name, Price: \$${price.toStringAsFixed(2)}');
    });
  }
}
```

CLASS ORDER

```
class Order {
  List<String> items = [];
  double foodPrice = 0.0;
  double buffetPrice = 0.0;
  double totalPrice = 0.0;
  Reservation reservation;
  OrderStatus orderStatus = OrderStatus.pending;
  PaymentStatus paymentStatus = PaymentStatus.unpaid;

  // Constructor to accept a Reservation instance
  Order(this.reservation);
```

```
void calculateTotal(Menu menu) {
  foodPrice = items.fold(0, (sum, item) => sum + (menu.getPrice(item) ?? 0.0));
  buffetPrice = reservation.amountofPeople * 10;
  totalPrice = foodPrice + buffetPrice;
}
```

```
 void showReceipt(Menu menu) {
   print('Receipt');
   print('Customer Name: ${reservation.name}');
   print('Table Number: ${reservation.tableNum}');
   print('Reservation Time: ${reservation.startTime} to ${reservation.endTime}');

   items.forEach((item) {
     double? itemPrice = menu.getPrice(item);
     print('Item: $item, Price: \$${itemPrice != null ? itemPrice.toStringAsFixed(2) : 'N/A'}');
   });
   print('Amount for ${reservation.amountofPeople} people: \$${buffetPrice.toStringAsFixed(2)}');
   print('Total: \$${totalPrice.toStringAsFixed(2)}');
 }
}
```

```
void paymentProcess() {
  stdout.write("Would you like to process the payment? (yes/no): ");
  String input = stdin.readLineSync() ?? '';

  if (input.toLowerCase() == 'yes') {
    processPayment(PaymentStatus.paid);
    updateStatus(OrderStatus.completed);
    print("Payment processed.");
  } else {
    print("Payment not processed.");
  }

  print("Order Status: $orderStatus");
  print("Payment Status: $paymentStatus");
}

void updateStatus(OrderStatus status) {
  orderStatus = status;
}

void processPayment(PaymentStatus payment) {
  paymentStatus = payment;
}
```

```dart
void modifyOrder(Menu menu) {
 while (true) {
   print("\nType 'add.item name' or 'remove. item name' \nPress Enter to exit.");
   stdout.write("Please enter what your order: ");
   String? input = stdin.readLineSync();

   // Exit if no input is given
   if (input == null || input.trim().isEmpty) {
     print("Exiting item modification.");
     break;
   }

   if (input.startsWith("add.")) {
     String itemToAdd = input.substring(4);
     if (menu.getPrice(itemToAdd) != null) {
       items.add(itemToAdd);
       print("$itemToAdd has been added to your order.");
     } else {
       print("Item '$itemToAdd' is not available on the menu.");
     }
   } else if (input.startsWith("remove.")) {
     String itemToRemove = input.substring(7);
     if (items.contains(itemToRemove)) {
       items.remove(itemToRemove);
       print("$itemToRemove has been removed from your order.");
     } else {
       print("Item '$itemToRemove' is not in your order.");
     }
   } else {
     print("Invalid command. Please use 'add.<item>' or 'remove.<item>'.");
   }
 }
}
```

CLASS PAYMENT STATUS

CLASS ORDER STATUS

```
// Enumeration for Payment Status
enum PaymentStatus { paid, unpaid }

// Enumeration for Order Status
enum OrderStatus { pending, completed }
```

CLASS RESERVATION

```
class Reservation {
  String name;
  DateTime startTime;
  DateTime endTime;
  int amountofPeople;
  int tableNum;

  Reservation(this.name, this.startTime, this.endTime, this.amountofPeople, this.tableNum);
}
```

CLASS RESTAURANT SYS

```
class RestaurantSys {
  List<Order> orders = [];
  List<Reservation> reservations = [];
  int totalTables = 10; // Total number of tables in the restaurant
  Duration reservationDuration = Duration(hours: 1, minutes: 30);
```

CREATE ORDER

```
void createOrder(Menu menu, RestaurantSys restaurantsystem) {
  // Display available reservations
  restaurantsystem.viewReservations();
  stdout.write("Select a reservation number to place an order: ");
  int? reservationChoice = int.tryParse(stdin.readLineSync() ?? '');

  if (reservationChoice != null && reservationChoice > 0 && reservationChoice <= restaurantsystem.reservations.length) {
    Reservation selectedReservation = restaurantsystem.reservations[reservationChoice - 1];
    Order order = Order(selectedReservation);
```

CREATE ORDER

```
    // Loop to add/remove items or finish order
    while (true) {
      print("\nOptions:");
      print("1. Order");
      print("2. Done");

      stdout.write("Choose an option: ");
      String? option = stdin.readLineSync();

      switch (option) {
        case "1": // Add an item
          menu.showMenu();
          order.modifyOrder(menu);
          break;

        case "2": // Finish the order
          print("Finishing your order...");
          order.calculateTotal(menu);
          restaurantsystem.placeOrder(order);
          order.showReceipt(menu);
          order.paymentProcess();
          return; // Exit the ordering loop

        default:
          print("Invalid option. Please choose a valid option.");
          break;
      }
    }
  } else {
    print("Invalid reservation selection.");
  }
}
```

ORDER

```
void placeOrder(Order order) {
  orders.add(order);
}

void removeOrder(Order order) {
  orders.remove(order);
}
```

ADD RESERVATION

```
void addReservation() {
  stdout.write("Enter name for reservation: ");
  String name = stdin.readLineSync() ?? '';

  stdout.write("Enter reservation time (HH:mm format, e.g., 8:30PM): ");
  String? timeInput = stdin.readLineSync();
  DateTime? startTime = _parseTime(timeInput);

  if (startTime == null) {
    print("\nInvalid time format. Please use HH:mm format.\n");
    return;
  }

  DateTime endTime = startTime.add(reservationDuration);

  stdout.write("Enter the amount of people: ");
  int? amountofPeople = int.tryParse(stdin.readLineSync() ?? '');

  stdout.write("Enter table number (1 - $totalTables): ");
  int? tableNum = int.tryParse(stdin.readLineSync() ?? '');

  if (amountofPeople != null && tableNum != null && tableNum > 0 && tableNum <= totalTables) {
    if (_isTableAvailable(tableNum, startTime, endTime)) {
      reservations.add(Reservation(name, startTime, endTime, amountofPeople, tableNum));
      print("\nReservation for $name at ${_formatTime(startTime)} for $amountofPeople people at table $tableNum has been added.\n");
    } else {
      print("\nTable $tableNum is not available at this time. Please choose a different time or table.\n");
    }
  } else {
    print("\nInvalid input. Please enter valid details.\n");
  }
}
```

ADD RESERVATION

```
DateTime? _parseTime(String? timeInput) {
  try {
    if (timeInput == null) return null;
    List<String> parts = timeInput.split(":");
    int hour = int.parse(parts[0]);
    int minute = int.parse(parts[1]);
    DateTime now = DateTime.now();
    return DateTime(now.year, now.month, now.day, hour, minute);
  } catch (e) {
    return null;
  }
}

String _formatTime(DateTime time) {
  return "${time.hour.toString().padLeft(2, '0')}:${time.minute.toString().padLeft(2, '0')}";
}

bool _isTableAvailable(int tableNum, DateTime startTime, DateTime endTime) {
  for (var reservation in reservations) {
    if (reservation.tableNum == tableNum) {
      bool overlaps = startTime.isBefore(reservation.endTime) && endTime.isAfter(reservation.startTime);
      if (overlaps) return false;
    }
  }
  return true;
}
```

VIEW RESERVATION

```
void viewReservations() {
  if (reservations.isEmpty) {
    print("\nNo reservations found.\n");
  } else {
    print("\nCurrent Reservations:");
    for (int i = 0; i < reservations.length; i++) {
      Reservation reservation = reservations[i];
      print(
        "${i + 1}. Name: ${reservation.name}, Time: ${_formatTime(reservation.startTime)} - ${_formatTime(reservation.endTime)},
        "Number of People: ${reservation.amountofPeople}, Table: ${reservation.tableNum}"
      );
    }
    print("");
  }
}
```

VIEW TABLE

```
void viewTables() {
  print("\nTable Status:");
  for (int i = 1; i <= totalTables; i++) {
    bool tableAvailable = true;
    for (var reservation in reservations) {
      if (reservation.tableNum == i) {
        tableAvailable = false;
        print("Table $i: Reserved from ${_formatTime(reservation.startTime)} to ${_formatTime(reservation.endTime)}");
        break;
      }
    }
    if (tableAvailable) {
      print("Table $i: Available");
    }
  }
  print("");
}
```

CANCEL RESERVATION

```
void cancelReservation() {
  viewReservations();
  stdout.write("Enter the reservation number to cancel: ");
  int? reservationNum = int.tryParse(stdin.readLineSync() ?? '');

  if (reservationNum != null && reservationNum > 0 && reservationNum <= reservations.length) {
    Reservation cancelledReservation = reservations.removeAt(reservationNum - 1);
    print("\nReservation for ${cancelledReservation.name} at table ${cancelledReservation.tableNum} has been canceled.\n");
  } else {
    print("\nInvalid reservation number.\n");
  }
}
}
```

MAIN

```
void main() {
  // Initialize the menu
  Menu menu = Menu();
  menu.addMenuItem("Pork", 0);
  menu.addMenuItem("Crispy Chicken", 0);
  menu.addMenuItem("USA Beef", 0);
  menu.addMenuItem("Juicy Beef Ball", 0);
  menu.addMenuItem("Dumpling", 0);
  menu.addMenuItem("Duck blood", 0);
  menu.addMenuItem("Mild Spicy soup", 0);
  menu.addMenuItem("Tomato soup", 0);
  menu.addMenuItem("Shichuan", 1.99);
  menu.addMenuItem("Soda", 1.99);
  menu.addMenuItem("Tea", 1.99);

  // Table Reservation
  RestaurantSys restaurantsystem = RestaurantSys();
```

MAIN

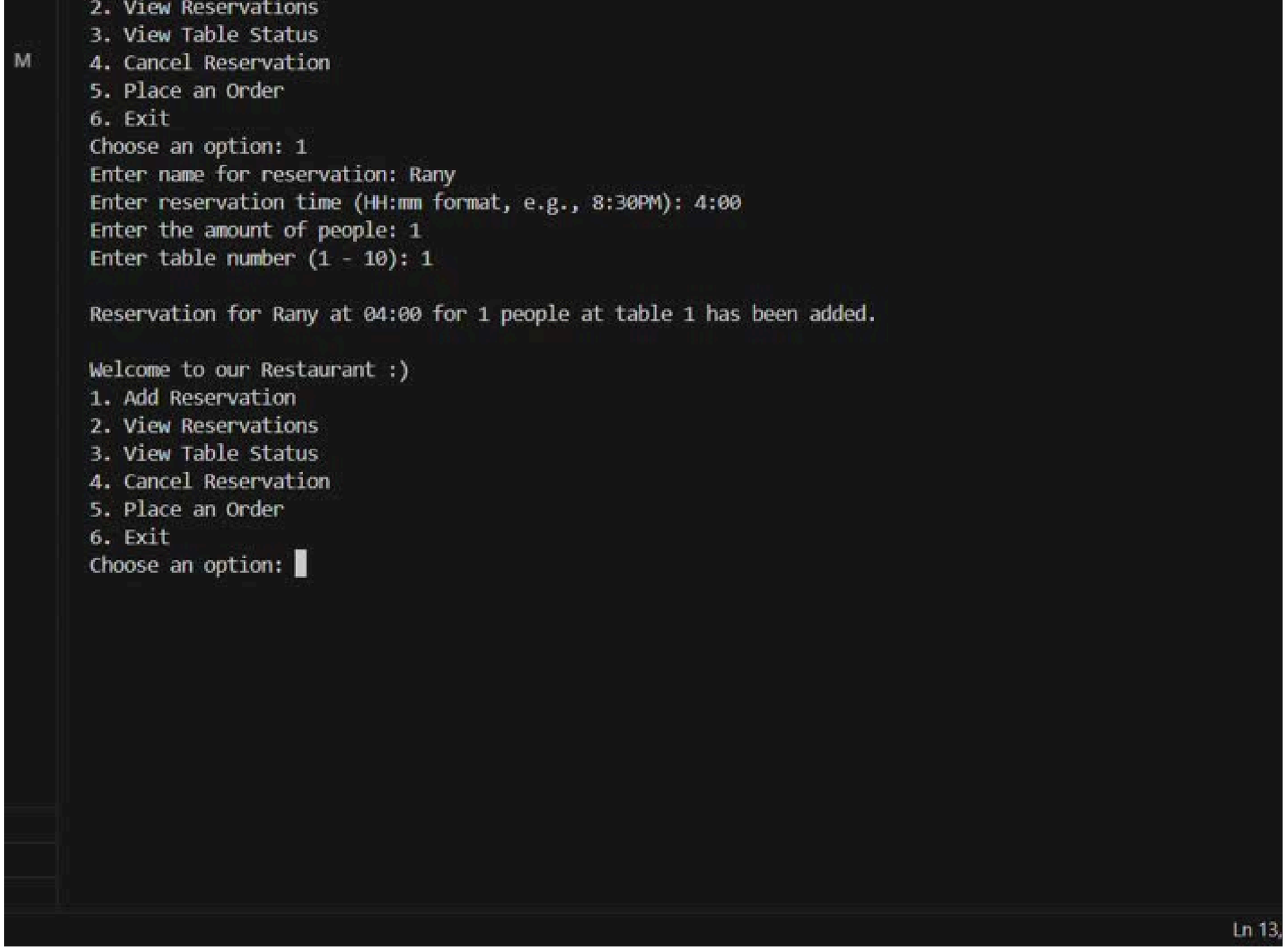
```
while (true) {
  print("Welcome to our Restaurant :)");
  print("1. Add Reservation");
  print("2. View Reservations");
  print("3. View Table Status");
  print("4. Cancel Reservation");
  print("5. Place an Order");
  print("6. Exit");

  stdout.write("Choose an option: ");
  String? choice = stdin.readLineSync();
```

MAIN

```
    switch (choice) {
      case "1":
        restaurantsystem.addReservation();
        break;
      case "2":
        restaurantsystem.viewReservations();
        break;
      case "3":
        restaurantsystem.viewTables();
        break;
      case "4":
        restaurantsystem.cancelReservation();
        break;
      case "5":
        restaurantsystem.createOrder(menu, restaurantsystem);
        break;
      case "6":
        print("Thank you Jub Jub, Goodbye!");
        return;
      default:
        print("\nInvalid option. Choose a valid option.\n");
        break;
    }
  }
}
```

DEMO


2. View Reservations
3. View Table Status
M
4. Cancel Reservation
5. Place an Order
6. Exit
Choose an option: 1
Enter name for reservation: Rany
Enter reservation time (HH:mm format, e.g., 8:30PM): 4:00
Enter the amount of people: 1
Enter table number (1 - 10): 1
Reservation for Rany at 04:00 for 1 people at table 1 has been added.
Welcome to our Restaurant :)
1. Add Reservation
2. View Reservations
3. View Table Status
4. Cancel Reservation
5. Place an Order
6. Exit
Choose an option:
Ln 13,


THANK YOU