21110324_LuongDangKhoi_KTPM Lab02

*** Yêu cầu 1:**

Các test case:

- Kiểm tra người dùng dưới tuổi cho phép.
- Kiểm tra số đêm đặt là 0 (không hợp lệ).
- Kiểm tra loại phòng không hợp lệ.
- Kiểm tra số phòng đặt vượt quá giới hạn.
- Kiểm tra thời gian đặt không hợp lệ (ít hơn 1 ngày trước ngày nhận phòng).
- Kiểm tra số phòng còn lại không đủ.
- Kiểm tra số người trong phòng vượt quá sức chứa.
- Kiểm tra số dư không đủ để thanh toán.
- Kiểm tra đặt phòng thành công.
- Kiểm tra hủy phòng trong thời hạn cho phép.
- Kiểm tra hủy phòng sau thời hạn không được phép.
- Kiểm tra đặt phòng Deluxe với nhiều đêm và phòng.
- Kiểm tra đặt phòng với tên phòng không phân biệt chữ hoa/chữ thường.

- Đầu tiên kiểm tra các thuộc tính của đối tượng không có thuộc tính về ngày đặt phòng, nên thêm thuộc tính bookingDate và khai báo thư viện java.time.LocalDate. Do bookingDate vừa mới được thêm vào và chưa có logic nên các cases có liên quan đến thời gian đều không đạt.

```
Before:
You must be at least 18 years old.
You must book at least 1 night.
Invalid room type.
Invalid room type.
Invalid room type.
Not enough Suite rooms available.
Too many people for the room.
Insufficient balance.
Invalid room type.
Room canceled successfully!
Room canceled successfully!
```

```java
public static void testBefore() {  1 usage
    // Test case 1: Giới hạn độ tuổi
    HotelBookingBefore booking1 = new HotelBookingBefore( userAge: 17,  roomType: "Standard",  nights: 2,  numberOfRoom
    System.out.println(booking1.bookRoom()); // Expected: "You must be at least 18 years old."

    // Test case 2: Số đêm tối thiểu
    HotelBookingBefore booking2 = new HotelBookingBefore( userAge: 25,  roomType: "Standard",  nights: 0,  numberOfRoom
    System.out.println(booking2.bookRoom()); // Expected: "You must book at least 1 night."

    // Test case 3: Loại phòng hợp lệ
    HotelBookingBefore booking3 = new HotelBookingBefore( userAge: 25,  roomType: "Luxury",  nights: 2,  numberOfRooms:
    System.out.println(booking3.bookRoom()); // Expected: "Invalid room type."

    // Test case 4: Giới hạn đặt phòng tối đa
    HotelBookingBefore booking4 = new HotelBookingBefore( userAge: 25,  roomType: "Standard",  nights: 2,  numberOfRoom
    System.out.println(booking4.bookRoom()); // Expected: "You can only book up to 3 rooms."

    // Test case 5: Giới hạn thời gian đặt phòng
    HotelBookingBefore booking5 = new HotelBookingBefore( userAge: 25,  roomType: "Standard",  nights: 2,  numberOfRoom
    System.out.println(booking5.bookRoom()); // Expected: "You must book at least 1 day in advance."

    // Test case 6: Số phòng còn lại không đủ
    HotelBookingBefore booking6 = new HotelBookingBefore( userAge: 25,  roomType: "Suite",  nights: 2,  numberOfRooms: 3
    System.out.println(booking6.bookRoom()); // Expected: "Not enough Suite rooms available."

    // Test case 7: Giới hạn số lượng người
    HotelBookingBefore booking7 = new HotelBookingBefore( userAge: 25,  roomType: "Deluxe",  nights: 2,  numberOfRooms:
    System.out.println(booking7.bookRoom()); // Expected: "Too many people for the room."

    // Test case 8: Xác minh thanh toán
    HotelBookingBefore booking8 = new HotelBookingBefore( userAge: 25,  roomType: "Deluxe",  nights: 2,  numberOfRooms:
    System.out.println(booking8.bookRoom()); // Expected: "Insufficient balance."

    // Test case 9: Đặt phòng thành công
    HotelBookingBefore booking9 = new HotelBookingBefore( userAge: 25,  roomType: "Standard",  nights: 2,  numberOfRoom
    System.out.println(booking9.bookRoom()); // Expected: "Booking successful!"

    // Test case 10: Hủy phòng trước 24 giờ
    System.out.println(booking9.cancelRoom()); // Expected: "Room canceled successfully!"

    // Test case 11: Hủy phòng sau thời hạn
    System.out.println(booking9.cancelRoom()); // Expected: "Room cancellation is not allowed after the deadline
}
```

- Sửa các lỗi logic:

+ Giới hạn thời gian dặt phòng trước ít nhất 1 ngày

```java
if (bookingDate.isBefore(LocalDate.now().plusDays( daysToAdd: 1))) {
    return "You must book at least 1 day in advance.";
}
```

+ Giới hạn số người theo sức chứa mỗi loại phòng

```java
int roomCapacity = switch (roomType) {
    case "Standard" -> 2;
    case "Deluxe" -> 4;
    case "Suite" -> 6;
    default -> 0;
};
```

+ Giới hạn phòng huỷ:

```
public String cancelRoom(boolean isWithin24Hours) {  2 usages
    if (!isWithin24Hours) {
        return "Room cancellation is not allowed after the deadline.";
    }
    return "Room canceled successfully!";
}
```

+ Thêm logic kiểm tra thanh toán và thời gian trả phòng:

```
public String cancelRoom(boolean isWithin24Hours, boolean isPaid) {  2 usages
    if (!isPaid) {
        return "Room cancellation is not allowed because the payment has not been made.";
    }
    if (!isWithin24Hours) {
        return "Room cancellation is not allowed after the deadline.";
    }
    return "Room canceled successfully!";
}
```

- Kết quả các test case sau khi chỉnh sửa:

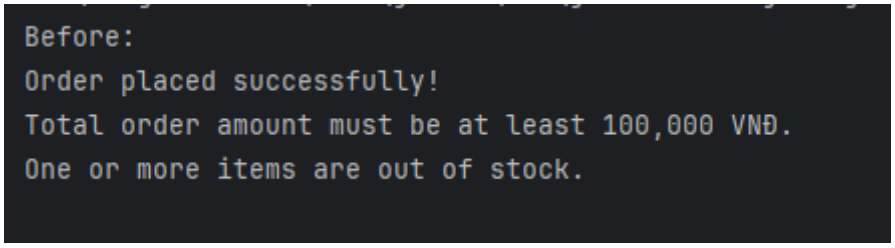
```
After:
You must be at least 18 years old.
You must book at least 1 night.
Invalid room type.
You can only book up to 3 rooms.
You must book at least 1 day in advance.
Not enough Suite rooms available.
Too many people for the room.
Insufficient balance.
Booking successful!
Room canceled successfully!
Room cancellation is not allowed after the deadline.
```

*** Yêu cầu 2:**

```java
public static void testBefore() {  1 usage
    // Test Case 1.1: Đặt đúng 5 món
    FoodOrderBefore order1 = new FoodOrderBefore(
            Arrays.asList("Pizza", "Burger", "Fries", "Salad", "Soup"),
            Arrays.asList(1, 1, 1, 1, 1),
            totalAmount: 600000,
            isPaid: true,
            paymentMethod: "credit_card",
            hasMainDish: true,
            isPromoCodeApplied: true,
            isItemAvailable: true
    );
    System.out.println(order1.placeOrder()); // Expect: "Order placed successfully!"

    // Test Case 2.2: Tổng giá trị đơn hàng < 100.000 VNĐ
    FoodOrderBefore order2 = new FoodOrderBefore(
            Arrays.asList("Fries"),
            Arrays.asList(1),
            totalAmount: 95000,
            isPaid: true,
            paymentMethod: "cash",
            hasMainDish: true,
            isPromoCodeApplied: true,
            isItemAvailable: true
    );
    System.out.println(order2.placeOrder()); // Expect: "Total order amount must be at least 100,000 VNĐ."

    // Test Case 5.2: Một món đã hết hàng
    FoodOrderBefore order3 = new FoodOrderBefore(
            Arrays.asList("Pizza", "Fries"),
            Arrays.asList(1, 1),
            totalAmount: 150000,
            isPaid: true,
            paymentMethod: "cash",
            hasMainDish: true,
            isPromoCodeApplied: false,
            isItemAvailable: false
    );
    System.out.println(order3.placeOrder()); // Expect: "One or more items are out of stock."
}
```

```java
public static void testAfter() {  1 usage
    // Test Case 3.2: Tổng giá trị đơn hàng > 500.000 VNĐ và áp mã giảm giá
    FoodOrderAfter order1 = new FoodOrderAfter(
            Arrays.asList("Pizza", "Burger", "Fries", "Salad", "Soup"),
            Arrays.asList(1, 1, 1, 1, 1),
            totalAmount: 600000,
            isPaid: true,
            paymentMethod: "credit_card",
            hasMainDish: true,
            isPromoCodeApplied: true,
            isItemAvailable: true
    );
    System.out.println(order1.placeOrder()); // Expect: "Order placed successfully!"

    // Test Case 6.2: Một món có số lượng > 10
    FoodOrderAfter order2 = new FoodOrderAfter(
            Arrays.asList("Pizza", "Fries"),
            Arrays.asList(11, 1),
            totalAmount: 300000,
            isPaid: true,
            paymentMethod: "credit_card",
            hasMainDish: true,
            isPromoCodeApplied: true,
            isItemAvailable: true
    );
    System.out.println(order2.placeOrder()); // Expect: "You can only order a maximum of 10 portions of each

    // Test Case 7.2: Chưa thanh toán
    FoodOrderAfter order3 = new FoodOrderAfter(
            Arrays.asList("Pizza", "Fries"),
            Arrays.asList(1, 1),
            totalAmount: 150000,
            isPaid: false,
            paymentMethod: "cash",
            hasMainDish: true,
            isPromoCodeApplied: true,
            isItemAvailable: true
    );
    System.out.println(order3.processPayment()); // Expect: "Payment is required before processing the order.
```

```java
// Test Case 10.2: Đặt không đủ 30 phút trước giao hàng
FoodOrderWithDelivery order4 = new FoodOrderWithDelivery(
        Arrays.asList("Pizza", "Burger"),
        Arrays.asList(1, 1),
        totalAmount: 200000,
        isPaid: true,
        paymentMethod: "cash",
        hasMainDish: true,
        isPromoCodeApplied: true,
        isItemAvailable: true,
        System.currentTimeMillis(), // Thời gian hiện tại
        System.currentTimeMillis() + 15 * 60 * 1000 // Thời gian giao hàng: 15 phút sau
);
System.out.println(order4.placeOrder()); // Expect: "Orders must be placed at least 30 minutes before
```



BlackBox Testing:

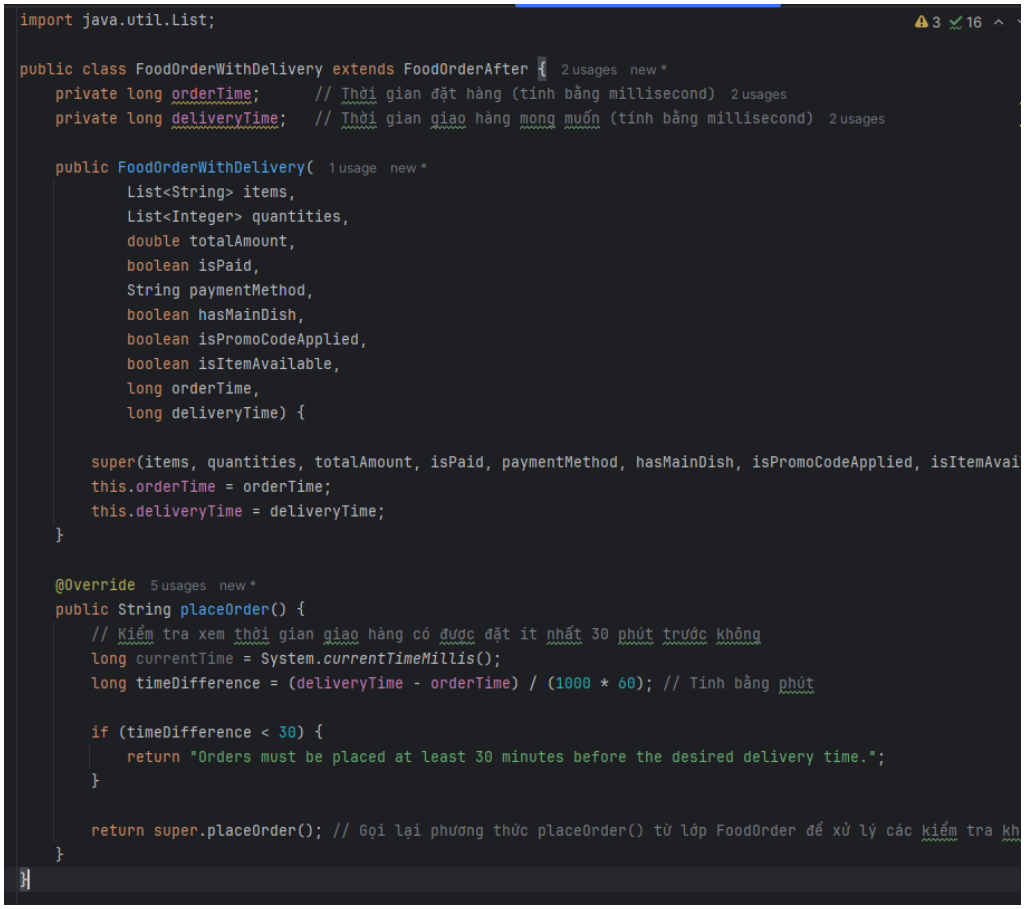
- Đặt quá 5 món → Hiển thị lỗi.
- Giá trị đơn hàng < 100.000 → Hiển thị lỗi.
- Đặt món hết hàng (isItemAvailable = false) → Hiển thị lỗi.
- Số lượng một món > 10 → Hiển thị lỗi.
- Không có món chính (hasMainDish = false) → Hiển thị lỗi.
- Đơn hàng trên 500.000 nhưng không áp mã khuyến mãi → Hiển thị lỗi.
- Thanh toán qua thẻ tín dụng nhưng < 200.000 → Hiển thị lỗi.
- Chưa thanh toán trước khi giao → Hiển thị lỗi.

WhiteBox Testing:

- Các nhánh logic được kiểm tra:

- items.size() > maxItemsPerOrder
- totalAmount < 100000
- !isItemAvailable
- Lặp qua tất cả quantities để kiểm tra quantity > 10
- !hasMainDish
- totalAmount > 500000 && !isPromoCodeApplied
- paymentMethod.equals("credit_card") && totalAmount < 200000
- !isPaid

- Thêm lớp FoodOrderWithDelivery kế thừa FoodOrderAfter (dey96 là lớp sau khi chỉnh sửa của FoodOrder)



```java
import java.util.List;

public class FoodOrderWithDelivery extends FoodOrderAfter {
    private long orderTime;      // Thời gian đặt hàng (tính bằng millisecond)
    private long deliveryTime;   // Thời gian giao hàng mong muốn (tính bằng millisecond)

    public FoodOrderWithDelivery(
            List<String> items,
            List<Integer> quantities,
            double totalAmount,
            boolean isPaid,
            String paymentMethod,
            boolean hasMainDish,
            boolean isPromoCodeApplied,
            boolean isItemAvailable,
            long orderTime,
            long deliveryTime) {

        super(items, quantities, totalAmount, isPaid, paymentMethod, hasMainDish, isPromoCodeApplied, isItemAvai
        this.orderTime = orderTime;
        this.deliveryTime = deliveryTime;
    }

    @Override
    public String placeOrder() {
        // Kiểm tra xem thời gian giao hàng có được đặt ít nhất 30 phút trước không
        long currentTime = System.currentTimeMillis();
        long timeDifference = (deliveryTime - orderTime) / (1000 * 60); // Tính bằng phút

        if (timeDifference < 30) {
            return "Orders must be placed at least 30 minutes before the desired delivery time.";
        }

        return super.placeOrder(); // Gọi lại phương thức placeOrder() từ lớp FoodOrder để xử lý các kiểm tra kh
    }
}
```

- Chỉnh sửa hàm placeOrder:

```java
public String placeOrder() {  4 usages  2 overrides  new *
    // Giới hạn số món trong đơn hàng
    if (items.size() > maxItemsPerOrder) {
        return "You can only order up to 5 items.";
    }

    // Kiểm tra tổng giá trị đơn hàng
    if (totalAmount < 100000) {
        return "Total order amount must be at least 100,000 VNĐ.";
    }

    // Kiểm tra tất cả các món ăn còn hàng
    if (!isItemAvailable) {
        return "One or more items are out of stock.";
    }

    // Kiểm tra số lượng từng món không vượt quá 10 phần
    for (int quantity : quantities) {
        if (quantity > 10) {
            return "You can only order a maximum of 10 portions of each item.";
        }
    }

    // Kiểm tra đơn hàng có ít nhất một món chính
    if (!hasMainDish) {
        return "You must have at least one main dish in your order.";
    }

    // Khuyến mãi chỉ áp dụng cho đơn hàng trên 500.000 VNĐ
    if (totalAmount > 500000 && !isPromoCodeApplied) {
        return "Promo code must be applied for orders over 500,000 VNĐ.";
    }

    // Giới hạn thanh toán qua thẻ tín dụng
    if (paymentMethod.equals("credit_card") && totalAmount < 200000) {
        return "Credit card payment is only available for orders above 200,000 VNĐ.";
    }

    // Đơn hàng hợp lệ
    return "Order placed successfully!";
}
```

- Các test case sau khi chỉnh sửa:

```
After:
Order placed successfully!
You can only order a maximum of 10 portions of each item.
Payment is required before processing the order.
Orders must be placed at least 30 minutes before the desired delivery time.
```