과제 7 (중간고사 대비)

제출 마감일: 2021-10-19 23:59
업로드 가능한 파일 수: 5
제출 방식: 개인

목적

interface 를 포함하여 지금까지 배운 자바의 기능들을 종합적으로 사용해 보는 연습을 합니다.

설명

한달 전 우리는 카페를 관리하는 시스템을 의뢰 받았습니다.

의뢰인은 별다방을 레퍼런스 모델로 이와 유사한 기능을 수행하는 시스템을 개발해 달라고 합니다.

개발 첫날, 카페에서 판매하는 메뉴 중 일부들 (아메리카노, 망고바나나, 캐모마일) 을 위한 클래스를 개발했습니다.

카페 시스템이 완성되어 갈수록 메뉴를 추가할 수 있도록 Beverage 를 추상 클래스로 정의합니다.

그런 후, 개별 메뉴들은 모두 이를 상속 받도록 디자인 했습니다. (Beverage, Blended, Coffee, Teavana 클래스)

```
public abstract class Beverage {
    enum SIZE {TALL, GRANDE, VENTI}
    String name;
    int basePrice;
    SIZE size;
```

```
    public Beverage(String name, int price, SIZE size) {
        this.name = name;
        this.basePrice = price;
        this.size = size;
    }

    public int getPrice() {
        return basePrice;
    }

    @Override
    public String toString() {
        return "name=" + name + ", Price=" + getPrice() + ", size=" + size.name() ;
    }
}


public class Blended extends Beverage {
    public Blended(String name) {
        super(name, 6300, SIZE.GRANDE);
    }
}


public class Coffee extends Beverage {
    public Coffee(String name) {
        super(name, 4100, SIZE.TALL);
    }
}


public class Teavana extends Beverage {
    public Teavana(String name) {
        super(name, 4100, SIZE.TALL);
    }
}
```

개발 둘째 날, 음료를 주문하는데 필요한 클래스를 생각해 봤습니다.

하나의 주문에는 아메리카노 1 잔, 망고 바나나 1 잔 등 여러 개의 메뉴가 포함될 수 있습니다.

이에 Order 클래스에 여러 개의 메뉴가 포함될 수 있도록,

Order 클래스 내부에 ArrayList 로 주문 음료들을 저장하기로 결정했습니다. (Order, OrderItem 클래스)

```
public class Order {
    private List<OrderItem> items = new ArrayList<>();

    private static int orderCounter = 1;
    private int orderNo ;

    ... // your code here
}


public class OrderItem {
    private Beverage beverage;
    private int quantity;

    public OrderItem(Beverage beverage, int quantity) {
        this.beverage = beverage;
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return beverage.toString() + ", quantity=" + quantity ;
    }
}
```

개발 셋째 날, 고객이 주문한 주문 내용을 저장할 저장소를 개발합니다.

데이터베이스 도입 여부가 아직 결정되지 않았습니다.

우선 ArrayList 에 주문들을 차곡차곡 저장해 둡니다.

저장한 주문들을 순회하기 위해 Iterator 인터페이스를 구현 (implements) 합니다. (OrderRepository 클래스)

```
public class OrderRepository implements Iterator<Order> {
    List<Order> orders = new ArrayList<>();
    // your code here
}
```

개발 넷째 날, 고객이 주문한 음료가 준비되면 픽업할 수 있는 기능을 구현합니다.

먼저 픽업의 여러 가지 방법들 (테이크 아웃, 드라이브 쓰루, 배달 등) 을 처리하기 위한 방법을 고민합니다.

픽업이 될려면 고객이 주문한 음료가 모두 준비되어야 합니다.

음료가 준비되면 알려 주는 인터페이스를 하나 정의합니다.

주문을 받을 때 픽업 방법을 결정하므로 Order 클래스의 내부에 선언합니다. (Pickup 인터페이스)

```
public class Order {
        private Order.PickUp delivery;

        public static interface PickUp {
            public abstract void handle(Order o);
        }
        ...
 }
```

픽업하는 방법에 따라서 시스템이 수행하는 기능과 필요한 정보가 조금씩 다릅니다.

예를 들어, 테이크 아웃 주문은 픽업 장소의 작은 모니터에 "주문번호 XX 고객님! 음료 준비 됐습니다" 가 출력돼야 합니다.

드라이브 쓰루 주문은 전용 픽업 장소에 도착한 자동차 운전자가 주문한 음료를 전달해야 합니다.

배달 주문은 배달 업체에 주문 고객의 주소 정보를 전달해야 합니다.

픽업 인터페이스를 구현 (implements) 하면서 필요한 정보 (주소, 차량번호 등) 을 멤버 변수로 가지는 클래스를 정의합니다.

(Delivery, DriveThru, TakeOut 클래스)

```
class Delivery implements Order.PickUp {
```

```
    private String address;

    // your code here
}

class DriveThru implements Order.PickUp {

    private String vehicleNumber;

    // your code here
}

class TakeOut implements Order.PickUp {

    // your code here
}
```

개발 다섯째 날, 지금까지 구현한 클래스들을 이용해서 시스템을 테스트 해 봅니다.

주문을 생성하는 과정에서 픽업 종류에 따라서 이를 처리하는 클래스들 (Delivery, DriveThru, TakeOut) 을 생성하는 코드를 한 곳에서 처리하고 싶습니다.

빌더냐 팩토리냐 고민하다 팩토리 클래스를 만들기로 결정합니다. (OrderTest, PickUpFactory 클래스)

```
public class PickUpFactory {

    class Delivery implements Order.PickUp {

        private String address;

        // your code here

    }
```

```
    class DriveThru implements Order.PickUp {
        private String vehicleNumber;
        // your code here
    }
    class TakeOut implements Order.PickUp {
        // your code here
    }
    public Order.PickUp makeTakeOut (){
        return new TakeOut();
    }
    public Order.PickUp makeDriveThru (String vehicleNumber){
        return new DriveThru(vehicleNumber);
    }
    public Order.PickUp makeDelivery (String address){
        return new Delivery(address);
    }
}
```

문제

고객의 주문이 준비되면, 고객이 원하는 픽업 방법 (테이크 아웃, 드라이브 쓰루, 배달) 에 따라 정보를 처리하는 프로그램을 작성하시오.

〈참조〉

- OrderTest 클래스 코드가 제공됩니다.
- OrderTest 클래스가 동작하도록 Order, OrderRepository, PickUpFactory 클래스를 구현하시오.
- com.cafe.menu 패키지에는 음료 관련 클래스들이 존재합니다. (Beverage, Blended, Coffee, Teavana)
- com.cafe.order 패키지에는 주문 관련 클래스들이 존재합니다. (Order, OrderItem, OrderRepository, PickUpFactory, OrderTest)

```
public class OrderTest {
    static PickUpFactory pickUpFactory = new PickUpFactory();
    public static void main(String[] args) {
        OrderRepository orderRepository = new OrderRepository();

        orderRepository.add(makeOrderForDelivery());
        orderRepository.add(makeOrderForDriveThru());
        orderRepository.add(makeOrderForTakeOut());

        while(orderRepository.hasNext()) {
            System.out.println(orderRepository);
            Order order = orderRepository.next();
            if (order == null) break;
            order.completed();
        }
    }

    private static Order makeOrderForTakeOut() {
        Order order = new Order();
        Order.PickUp pickUp = pickUpFactory.makeTakeOut();
        order.setPickUp(pickUp);
        order.addItem(new OrderItem(new Teavana("Chamomile"), 1)) ;
        order.addItem(new OrderItem(new Coffee("Americano"), 1)) ;
        return order;
    }

    private static Order makeOrderForDriveThru() {
        Order order = new Order();
        Order.PickUp pickUp = pickUpFactory.makeDriveThru("001 가 0000");
        order.setPickUp(pickUp);
        order.addItem(new OrderItem(new Coffee("Americano"), 2)) ;
        return order;
    }

    private static Order makeOrderForDelivery() {
        Order order = new Order();
        Order.PickUp pickUp = pickUpFactory.makeDelivery("Pusan National University");
        order.setPickUp(pickUp);
        order.addItem(new OrderItem(new Blended("MangoBanana"), 1)) ;
        order.addItem(new OrderItem(new Coffee("Americano"), 1)) ;
        return order;
    }
}
```

입력

없음

출력

--- 주문 관리자 화면 ---

현재 주문수는 총 3 입니다.

〈 주문 내역 〉

주문번호: 1 - [name=MangoBanana, Price=6300, size=GRANDE, quantity=1, name=Americano, Price=4100, size=TALL, quantity=1]

주문번호: 2 - [name=Americano, Price=4100, size=TALL, quantity=2]

주문번호: 3 - [name=Chamomile, Price=4100, size=TALL, quantity=1, name=Americano, Price=4100, size=TALL, quantity=1]

--- 배달 관리 화면 ---

배달통으로 주문을 전달합니다!

배송주소: Pusan National University

--- 주문 관리자 화면 ---

현재 주문수는 총 2 입니다.

〈 주문 내역 〉

주문번호: 2 - [name=Americano, Price=4100, size=TALL, quantity=2]

주문번호: 3 - [name=Chamomile, Price=4100, size=TALL, quantity=1, name=Americano, Price=4100, size=TALL, quantity=1]

--- 드라이브 쓰루 화면 ---

차량번호: 001 가 0000

주문하신 음료가 준비 되었습니다!

--- 주문 관리자 화면 ---

현재 주문수는 총 1 입니다.

〈 주문 내역 〉

주문번호: 3 - [name=Chamomile, Price=4100, size=TALL, quantity=1, name=Americano, Price=4100, size=TALL, quantity=1]

--- 테이크 아웃 화면 ---

주문번호: 3

주문하신 음료가 준비 되었습니다!