1. 개요

콘솔 기반의 서브웨이 샌드위치 주문 프로그램이다. 메인 메뉴는 총 4가지로 샌드위치 주문, 사이드 주문, 장바구니, 관리자 메뉴가 있다.
샌드위치는 주문 시 속 재료와 빵, 야채, 소스를 직접 고를 수 있고, 야채와 소스는 복수 선택이 가능하다. 샌드위치와 사이드 모두 같은 품목을 수량 지정해서 구매 가능하다. 장바구니에서는 결제를 진행하거나 장바구니에 담겨있는 품목을 삭제할 수 있다. 삭제시 일부 수량만 삭제하는 것도 가능하다. 결제는 카드번호를 입력받아 진행하며, 핸드폰 번호를 입력해 포인트를 적립하거나 사용할 수 있다.
관리자 메뉴에서는 판매 품목들을 추가하거나 삭제할 수 있고, 기존 판매 품목의 가격을 변경할 수 있다. 일일결산과 월말 결산을 통해 매출 내역을 확인할 수 있고, 월말결산은 확인 후 삭제할 수 있다.

2.사용된 기술

Java SE, Generic, OOP, Collection

3. 자료형 설계

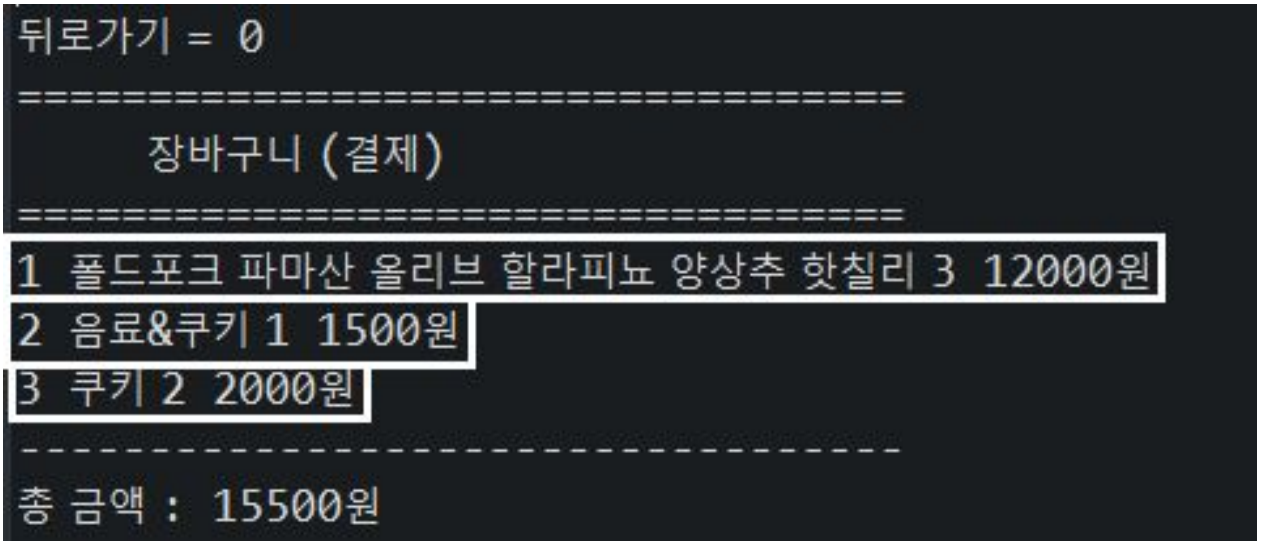
- Item 클래스

서브웨이에서 판매하는 품목에는 여러 종류가 있다. 크게 샌드위치와 사이드 메뉴로 나뉘는데, 샌드위치 안에서도 선택할 수 있는 항목이 샌드위치의 기본 속재료와 빵, 야채, 소스 등으로 다양하다.
이를 각각의 화면에서 선택하도록 설계하였기 때문에 종류별로 구분할 필요가 있다. 또한 각 판매 품목마다 품목명이 있어야 하고, 해당 품목의 가격도 있어야 한다.

```java
public class Item {
    private String category; // 카테고리 이름
    private String name; // 품목 이름
    private int price; // 품목 가격

    // 기본생성자
    public Item() {}

    // 카테고리 이름, 품목 이름, 품목의 가격을 매개변수로 받는 생성자
    public Item(String category, String name, int price) {
        this.category = category;
        this.name = name;
        this.price = price;
    }

    // 카테고리 이름 반환
    public String getCategory() {
        return category;
    }

    // 품목 이름 반환
    public String getName() {
        return name;
    }

    // 품목의 가격 반환
    public int getPrice() {
        return price;
    }
}
```

판매 품목의 카테고리, 이름, 가격 정보를 얻을 수 있도록 각 멤버변수의 getter 메소드를 구현했다.

- SubOrder 클래스



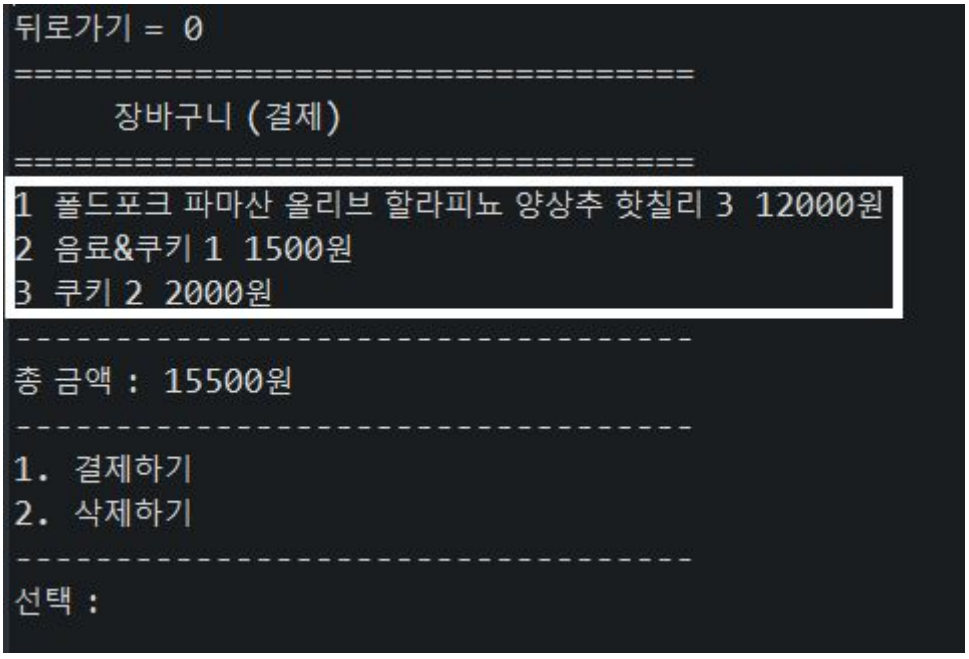
위 그림은 고객이 품목을 여러가지 고른 후 장바구니에 들어갔을 때의 화면이다. 그림에서 네모로 표시된 부분들이 각각 고객이 한번의 주문 액션을 통해 고른 내용이다. 이 각각의 내역들을 저장하기 위한 클래스가 필요한데, 한가지 내역에 여러개의 품목이 들어갈 수 있고, 그 전체 품목을 몇개 구매할 것인지의 정보와 총 가격이 저장되어야 한다.

```java
public class SubOrder {
    private List<Item> item;// 주문 품목 리스트 (샌드위치, 빵, 야채, 소스.. 또는 사이드)
    private int count; // 주문 수량
    private int price; // 전체 가격

    // 기본생성자
    public SubOrder() {}

    // 고객이 선택한 품목의 리스트와 해당 품목들의 수량을 매개변수로 받는 생성자
    public SubOrder(List<Item> item, int count) {
        this.item = item;
        this.count = count;
        this.calculatePrice();
    }

    // 주문 품목의 리스트를 반환
    public List<Item> getItem() {
        return item;
    }

    // 주문 품목의 수량을 반환
    public int getCount() {
        return count;
    }
```

```java
    // 주문 품목의 수량을 설정
    public void setCount(int count) {
        this.count = count;
        this.calculatePrice();
    }

    // 주문 품목의 가격을 반환
    public int getPrice() {
        return price;
    }

    // 주문 품목들의 총 가격을 계산
    private void calculatePrice() {
        int tmp = 0;
        for (Item i : item) {
            tmp += i.getPrice();
        }
        tmp = tmp * count;
        this.price = tmp;
    }

    // 품목이름 품목이름 ... 품목이름 수량 가격 형식으로 문자열 반환
    @Override
    public String toString() {
        String tmp = new String();
        for(Item i : item) {
            tmp = tmp.concat(i.getName() + " ");
        }
        tmp = tmp.concat(this.count + " ");
        tmp = tmp.concat(this.price + "원");
        return tmp;
    }
}
```

하나의 SubOrder에 여러 품목이 들어갈 수 있기 때문에 Item의 리스트로 멤버변수를 넣어줬다. 주문 정보를 알 수 있도록 각 멤버변수의 getter 메소드를 구현해줬고, count의 경우는 주문을 일부 삭제하는 경우 변경될 수 있기 때문에 setter 메소드도 같이 구현해주었다. price는 품목들과 수량에 따라 달라지기 때문에 객체 생성될 때와 count 값이 변경될 때 계산해서 할당할 수 있도록 calculatePrice 메소드를 따로 만들어주었다. 또한 출력을 위해 toString 메소드를 오버라이드 해서 품목명들과 수량, 가격을 하나의 문자열로 만들어서 반환하게 하였다.

- Order 클래스



네모로 표시된 전체가 하나의 주문이고 주문 안에 SubOrder가 들어있다. 장바구니에 들어있는 품목을 저장하거나 결산을 위한 주문 정보를 저장하는데 사용할 클래스가 필요하다.

```
 6 public class Order {
 7     private String date; // 주문 결제 완료된 날짜
 8     private List<SubOrder> item; // 서브 주문 리스트 (샌드위치, 쿠키, 웨지감자 ...)
 9
10     // 기본 생성자
11     public Order() {
12         item = new ArrayList<SubOrder>();
13     }
14
15     // 주문 날짜 반환
16     public String getDate() {
17         return date;
18     }
19
20     // 주문 날짜 설정
21     public void setDate(String date) {
22         this.date = date;
23     }
24
25     // 해당 주문의 상세 내역 반환
26     public List<SubOrder> getItem(){
27         return item;
28     }
```

```
30      // 해당 주문의 상세 내역 설정
31      public void setItem(List<SubOrder> item) {
32          this.item = item;
33      }
34
35      // SubOrder의 toString을 호출해서 장바구니에 들어있는 주문 목록을 문자열 리스트로 받아온다
36      public List<String> listSubOrders() {
37          List<String> tmp = new ArrayList<String>();
38          for(SubOrder so : item) {
39              tmp.add(so.toString());
40          }
41          return tmp;
42      }
43  }
```

결산을 위해서는 날짜 정보가 필요하기 때문에 날짜를 문자열로 저장할 date 변수를 선언해주었다. 또한 하나의 주문에 여러 SubOrder들이 들어있는 것을 구현하기 위해 SubOrder 리스트를 변수로 선언한 후 각 변수의 getter, setter 메소드를 만들어주었다.
listSubOrders 메소드는 SubOrder리스트에 저장된 각각의 SubOrder 정보를 출력하는데 사용하는 메소드이다. 각 SubOrder의 toString 메소드 실행한 결과들을 String 리스트로 만들어 반환해주게 된다.

- Membership 클래스

```
==================================
      결제 방법 선택
==================================
1. 일반 카드 결제
2. 포인트 적립 결제
3. 포인트 사용 결제
----------------------------------
선택 : 2

카드 번호 : 1234 1234 1234 1234
핸드폰 번호 : 010-1111-2222

총 4000원 결제되었습니다.
400 포인트 적립
```

포인트를 적립하며 결제하는 화면이다. 포인트를 적립할 때 고객을 식별하기 위해 핸드폰 번호가 필요하고, 후에 포인트 사용을 위해서는 적립된 포인트 내역을 저장해두어야 한다.

```java
public class Membership {
    private String phone; // 고객의 핸드폰 번호
    private int point; // 적립된 포인트

    // 기본생성자
    public Membership() {}

    // 핸드폰 번호, 적립 포인트를 매개변수로 받는 생성자
    public Membership(String phone, int point) {
        this.phone = phone;
        this.point = point;
    }

    // 핸드폰 번호 반환
    public String getPhone() {
        return phone;
    }

    // 적립된 포인트 반환
    public int getPoint() {
        return point;
    }

    // 적립 포인트 설정
    public void setPoint(int point) {
        this.point = point;
    }
}
```

고객의 핸드폰 번호를 저장할 phone과 적립된 포인트를 저장할 point 변수를 선언해주고 필요한 getter, setter 메소드를 만들어주었다.

- KioskDAO 클래스

```java
public class KioskDAO {
    private List<Item> item = new ArrayList<Item>(); // 판매 품목 저장
    private List<Membership> mem = new ArrayList<Membership>(); // 포인트 적립 저장
    private Order cart = new Order(); // 장바구니에 들어있는 품목 저장
    private List<Order> order = new ArrayList<Order>(); // 실제 결제된 주문들 저장
```

각 자료형 클래스들을 멤버로 가지고 Service 클래스에서 필요로 하는 경우 데이터들을 반환하거나 변경해주는 역할을 한다.

판매 품목들은 Item 리스트, 포인트 적립 정보는 Membership 리스트, 장바구니에 담겨있는 항목들의 정보는 Order형 변수인 cart에, 결제까지 완료한 주문들의 내역은 Order 리스트에 저장해주게 된다.

```
189     // 결제된 주문 내역 중 특정 날짜 (yyyy-MM-dd || yyyy-MM)에 해당하는 리스트 반환
190     public List<Order> listOrder(String date) {
191         List<Order> tmp = new ArrayList<Order>();
192         for(Order o : order) {
193             if(o.getDate().contains(date)) {
194                 tmp.add(o);
195             }
196         }
197         return tmp;
198     }
```

listOrder 메소드는 날짜를 매개변수로 받아 해당 날짜에 결제된 주문건들을 Order 자료형의 리스트로 반환해주는 메소드이다. Order 객체의 date와 매개변수로 받은 날짜를 비교해주게 되는데, 비교할 때 equals 메소드가 아닌 contains 메소드를 사용하기 때문에 일일결산과 월말결산에서 모두 활용할 수 있다.

```
236     // 이미 있는 품목의 가격을 변경하는 메소드
237     // 가격이 바뀌었을 때 이미 결제된 주문들의 품목까지 같이 바뀌지 않도록 삭제 후 재등록으로 진행
238     public void changePrice(String category, String itemName, int price) {
239         for(Item i : item) {
240             if(i.getCategory().equals(category) && i.getName().equals(itemName)) {
241                 int index = item.indexOf(i);
242                 item.remove(index);
243                 item.add(new Item(category, itemName, price));
244                 return;
245             }
246         }
247     }
```

changePrice 메소드는 이미 존재하는 판매 품목의 가격을 변경해주는 메소드이다. 이 경우 기존의 Item 객체의 가격을 변경하면 이미 결제된 주문들의 가격이 변경될 수 있다. 그래서 해당하는 품목을 판매품목 리스트에서 삭제해주고 재등록하는 방식으로 메소드를 구현하였다.