오프라인 코딩테스트에 임하는 자세

5시간 동안 모든 요구사항을 구현하는 것에 너무 집중하지 않았으면 한다.

하나를 구현하더라도 제대로 구현하는 것에 집중했으면 한다.

프리코스 과정을 스스로의 힘으로 구현했다는 것을 증명하는데 집중해라.

기능을 하나 더 구현하는 것보다

- **기능 목록, 커밋 로그를 잘 작성하는 것**
- **코드 컨벤션, 읽기 좋은 코드 등 프리코스 과정에서 지켜야할 원칙을 지키는 것**
- **구현한 기능이 모두 동작하는 것**

이 더 중요하다.

오프라인 코딩테스트 진행방식

진행 방식

- 오프라인 코딩테스트 진행 방식은 프리코스와 동일하다.
- 미션은 **기능 요구사항, 프로그래밍 요구사항, 과제 진행 요구사항** 세 가지로 구성되어 있다.
    - 세 개의 요구사항을 만족하기 위해 노력한다. 특히 기능을 구현하기 전에

      기능 목록을 만들고, 기능 단위로 commit 하는 방식으로 진행한다.

미션 제출 방법

- 미션 구현을 완료한 후 **GitHub**을 통해 제출해야 한다.
  - GitHub을 활용한 제출 방법은 **프리코스 과제 제출** 문서 참고해 제출한다.
- GitHub에 미션을 제출한 후 woowa_course@woowahan.com 로 메일을 발송한다.

email 템플릿

- 제목 양식
- 본문 양식
  - Github Id
  - Pull Request URL

email 예시

새 메일

받는사람 **우아한테크코스**

제목 **[숫자야구게임] 박재성 미션 제출합니다.**

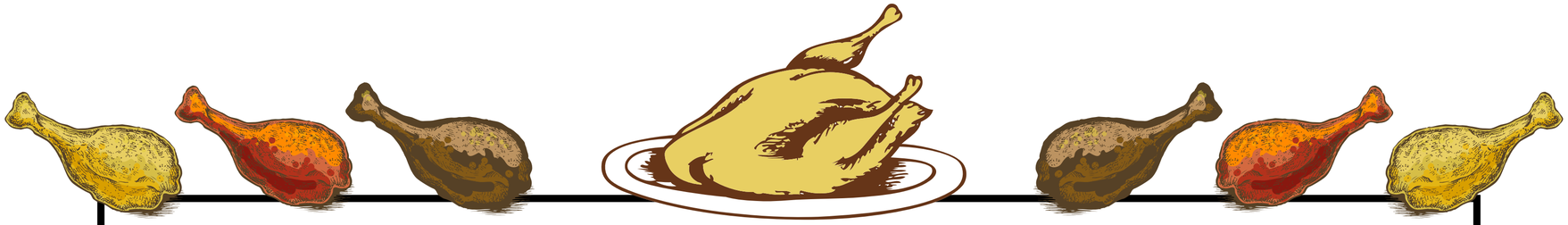
안녕하세요.

이번 미션 생각보다 쉽지 않네요.

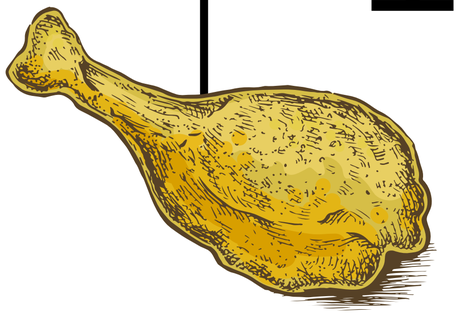
특히 기능을 분리하고 기능 단위로 commit 하는 것이 쉽지 않다는 것을 느꼈어요.

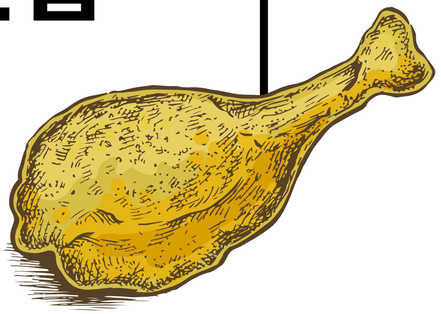
하지만 기능을 분리하고 구현했더니 더 명확하게 구현할 수 있다는 것을 느낄 수 있는 기회가 된 것 같아요.

* **Github ID**: pobiconan
* **Pull Request URL**: https://github.com/woowacourse/java-baseball/pull/1



오프라인 코딩테스트 - 치킨집





기능 요구사항

- 치킨집 사장님이 사용하는 간단한 포스(POS) 프로그램을 구현한다.

  주문등록, 결제하기, 프로그램 종료 기능을 가진다.
- 메뉴 기본 정보가 주어지며 **메뉴 번호, 종류, 이름, 가격**을 가진다.
- 테이블 기본 정보가 주어지며 **테이블 번호**를 가진다.
- 한 테이블에서 주문할 수 있는 **한 메뉴의 최대 수량은 99개**이다.
- 주문이 등록된 테이블은 **결제가 이루어지기 전까지 테이블 목록에 별도로 표시**한다.

기능 요구사항

- 주문 내역에 대한 계산을 할 때는 **결제 유형에 따라 할인율**이 달라진다.
  - 치킨 종류 메뉴의 수량 합이 **10개가 넘는 경우 10,000원씩 할인**된다.
    - e.g. 10개는 10,000원 할인, 20개는 20,000원 할인
  - **현금 결제는 5%가 할인**되며 **할인된 금액에서 한 번 더 할인**이 가능하다.
- 주문 혹은 결제가 불가능한 경우 그 이유를 보여 주고, 다시 주문 혹은 결제가 가능하도록 해야 한다.
- 최종 결제 금액을 보여준다.

프로그램 실행 결과

Run: Application

메인화면
1 - 주문등록
2 - 결제하기
3 - 프로그램 종료

원하는 기능을 선택하세요.
1

테이블 목록
```
┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐
| 1 || 2 || 3 || 5 || 6 || 8 |
└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘
```

테이블을 선택하세요.
1
[치킨] 1 - 후라이드 : 16000원
[치킨] 2 - 양념치킨 : 16000원
[치킨] 3 - 반반치킨 : 16000원
[치킨] 4 - 통구이 : 16000원
[치킨] 5 - 간장치킨 : 17000원
[치킨] 6 - 순살치킨 : 17000원
[음료] 21 - 콜라 : 1000원
[음료] 22 - 사이다 : 1000원

등록할 메뉴를 선택하세요.
1

메뉴의 수량을 입력하세요.
1

메인화면
1 - 주문등록
2 - 결제하기
3 - 프로그램 종료

원하는 기능을 선택하세요.
1

테이블 목록
```
┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐
| 1 || 2 || 3 || 5 || 6 || 8 |
└ ₩ ┘└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘
```

프로그램 실행 결과

Run: Application

테이블을 선택하세요.

1

[치킨] 1 - 후라이드 : 16000원

[치킨] 2 - 양념치킨 : 16000원

[치킨] 3 - 반반치킨 : 16000원

[치킨] 4 - 통구이 : 16000원

[치킨] 5 - 간장치킨 : 17000원

[치킨] 6 - 순살치킨 : 17000원

[음료] 21 - 콜라 : 1000원

[음료] 22 - 사이다 : 1000원

등록할 메뉴를 선택하세요.

21

메뉴의 수량을 입력하세요.

1

메인화면

1 - 주문등록

2 - 결제하기

3 - 프로그램 종료

원하는 기능을 선택하세요.

2

테이블 목록

```
┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐┌ ─ ┐
| 1 || 2 || 3 || 5 || 6 || 8 |
└ ₩ ┘└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘└ ─ ┘
```

테이블을 선택하세요.

1

주문 내역

메뉴 수량 금액

후라이드 1 16000

콜라 1 1000

1번 테이블의 결제를 진행합니다.

신용 카드는 1번, 현금은 2번

1

최종 결제할 금액

17000원

메인화면

1 - 주문등록

2 - 결제하기

3 - 프로그램 종료

...

프로그래밍 요구사항

```
public class Application {
    // TODO 구현 진행
    public static void main(String[] args) {
        final List<Table> tables = TableRepository.tables();
        OutputView.printTables(tables);

        final int tableNumber = InputView.inputTableNumber();

        final List<Menu> menus = MenuRepository.menus();
        OutputView.printMenus(menus);
    }
}
```

- src/main/java/Application의 main()을 실행해 프로그램을 시작한다.
- Application의 main()에서 구현을 시작한다.

프로그래밍 요구사항 - 클래스 1

```
public class Menu {
    private final int number;
    private final String name;
    private final Category category;
    private final int price;

    public Menu(final int number, final String name, final Category category, final int price) {
        this.number = number;
        this.name = name;
        this.category = category;
        this.price = price;
    }
}
```

- 다음 **Menu** 클래스를 활용해 구현해야 한다.
- **Menu**에 기본 생성자를 추가할 수 없다.
- **Menu**의 필드(인스턴스 변수)를 추가할 수 없다.
  - 단, 기존 필드(인스턴스 변수)의 데이터 타입은 변경할 수 있다.
- 필드(인스턴스 변수)의 접근 제어자는 **private**으로 구현해야 한다.

프로그래밍 요구사항 - 클래스 2

```
public class MenuRepository {
    private static final List<Menu> menus = new ArrayList<>();

    static {
        menus.add(new Menu(1, "후라이드", Category.CHICKEN, 16_000));
        menus.add(new Menu(2, "양념치킨", Category.CHICKEN, 16_000));
        menus.add(new Menu(3, "반반치킨", Category.CHICKEN, 16_000));
        menus.add(new Menu(4, "통구이", Category.CHICKEN, 16_000));
        menus.add(new Menu(5, "간장치킨", Category.CHICKEN, 17_000));
        menus.add(new Menu(6, "순살치킨", Category.CHICKEN, 17_000));
        menus.add(new Menu(21, "콜라", Category.BEVERAGE, 1_000));
        menus.add(new Menu(22, "사이다", Category.BEVERAGE, 1_000));
    }

    public static List<Menu> menus() {
        return Collections.unmodifiableList(menus);
    }
}
```

- 다음 **MenuRepository** 클래스를 활용해 구현해야 한다.
- 데이터를 조회하는 DB 역할을 한다.
- **MenuRepository**의 기존 코드는 수정할 수 없다.
  - 단, 추가는 가능하다.

프로그래밍 요구사항 - 클래스 3

```
public class Table {
    private final int number;

    public Table(final int number) {
        this.number = number;
    }
}
```

- 다음 **Table** 클래스를 활용해 구현해야 한다.
- **Table**에 기본 생성자를 추가할 수 없다.
- 필드(인스턴스 변수)의 접근 제어자는 **private**으로 구현해야 한다.

프로그래밍 요구사항 - 클래스 4

```
public class TableRepository {
    private static final List<Table> tables = new ArrayList<>();

    static {
        tables.add(new Table(1));
        tables.add(new Table(2));
        tables.add(new Table(3));
        tables.add(new Table(5));
        tables.add(new Table(6));
        tables.add(new Table(8));
    }

    public static List<Table> tables() {
        return Collections.unmodifiableList(tables);
    }
}
```

- 다음 **TableRepository** 클래스를 활용해 구현해야 한다.
- 데이터를 조회하는 DB 역할을 한다.
- **TableRepository**의 기존 코드는 수정할 수 없다.
  - 단, 추가는 가능하다.

프로그래밍 요구사항 - 제약1

- **자바 코드 컨벤션을 지키면서 프로그래밍**한다.
  - 참고 문서: https://naver.github.io/hackday-conventions-java/
- **indent(인덴트, 들여쓰기) depth를 3이 넘지 않도록 구현한다. 2까지만 허용**한다.
  - 예를 들어 while문 안에 if문이 있으면 들여쓰기는 2이다.
  - 힌트: indent(인덴트, 들여쓰기) **depth를 줄이는 좋은 방법은 함수(또는 메서드)를 분리**하면 된다.
- **3항 연산자를 쓰지 않는다.**
- **함수(또는 메서드)가 한 가지 일**만 하도록 최대한 작게 만들어라.

프로그래밍 요구사항 - 제약2

- **함수(또는 메서드)의 길이가 15라인을 넘어가지 않도록 구현**한다.
  - **함수(또는 메서드)가 한 가지 일만 잘 하도록 구현**한다.
- **else 예약어를 쓰지 않는다.**
  - 힌트: if 조건절에서 값을 return하는 방식으로 구현하면 else를 사용하지 않아도 된다.
  - else를 쓰지 말라고 하니 switch/case로 구현하는 경우가 있는데 switch/case도 허용하지 않는다.
- public/protected/private/package 접근 제어자를 용도에 적합하게 사용해 구현한다.
- **함수(또는 메소드)의 인자 수를 3개까지만 허용**한다. 4개 이상은 허용하지 않는다.
  - 단, 생성자는 제외한다.

과제 진행 요구사항

- 미션은 https://github.com/woowacourse/java-chicken-2019 저장소를 fork/clone 해 시작한다.
- **기능을 구현하기 전에 README.md 파일에 구현할 기능 목록을 정리**해 추가한다.
- **git의 commit 단위는 앞 단계에서 README.md 파일에 정리한 기능 목록 단위로 추가**한다.
- **프리 코스 과제 제출** 문서 절차를 따라 과제를 제출한다.

오프라인 코딩 테스트 마감 및 기준

- **2019년 12월 21일(토) 18시 10분까지** GitHub을 통한 미션 제출과 메일 전송까지 완료해야 한다
- **2019년 12월 21일(토) 18시 10분 이후 추가 push도 허용하지 않는다.**
- **2019년 12월 21일(토) 18시 10분 이후 제출할 경우 코딩 테스트를 제출하지 않은 것**으로 한다.