목표

학습 목표

- 관련 함수를 묶어 클래스를 만들고, 객체들이 협력하여 하나의 큰 기능을 수행하도록 한다.
- 클래스와 함수에 대한 단위 테스트를 통해 의도한 대로 정확하게 작동하는 영역을 확보한다.
- 2주 차 공통 피드백을 최대한 반영한다.

프리코스 진행 방식

진행 방식

- 미션은 **과제 진행 요구 사항**, **기능 요구 사항**, **프로그래밍 요구 사항** 세 가지로 구성되어 있다.
- 세 개의 요구 사항을 만족하기 위해 노력한다. 특히 기능을 구현하기 전에 기능 목록을 만들고, 기능 단위로 커밋 하는 방식으로 진행한다.
- **기능 요구 사항에 기재되지 않은 내용은 스스로 판단하여 구현한다.**
- 매주 진행할 미션은 화요일 오후 3시부터 확인할 수 있으며, 다음 주 월요일까지 구현을 완료하여 제출해야 한다. **제출은 일요일 오후 3시부터 가능하다.**
  - **정해진 시간을 지키지 않을 경우 미션을 제출하지 않은 것으로 간주한다.**
  - 종료 일시 이후에는 추가 푸시를 허용하지 않는다.

미션 제출 방법

- 미션 구현을 완료한 후 GitHub을 통해 제출해야 한다.
  - GitHub을 활용한 제출 방법은 프리코스 과제 제출 문서를 참고해 제출한다.
- GitHub에 미션을 제출한 후 우아한테크코스 지원 플랫폼에 PR 링크를 포함하여 최종 제출한다.
  - 자세한 안내는 제출 가이드를 참고한다.
  - 과제를 수행하면서 느낀 점, 배운 점, 많은 시간을 투자한 부분 등 자유롭게 작성한다.

과제 제출 전 체크 리스트

- 기능을 올바르게 구현했더라도 **요구 사항에 명시된 출력 형식을 따르지 않으면 0점**을 받게 된다.
- 기능 구현을 완료한 후 아래 가이드에 따라 모든 테스트가 성공적으로 실행되는지 확인한다.
- **테스트가 실패하면 점수가 0점**이 되므로 제출하기 전에 반드시 확인한다.

테스트 실행 가이드

- 터미널에서 `java -version`을 실행하여 Java 버전이 21인지 확인한다. Eclipse 또는 IntelliJ IDEA와 같은 IDE에서 Java 21로 실행되는지 확인한다.

- 터미널에서 Mac 또는 Linux 사용자의 경우 `./gradlew clean test` 명령을 실행하고, Windows 사용자의 경우 `gradlew.bat clean test` 또는 `.\gradlew.bat clean test` 명령을 실행할 때 모든 테스트가 아래와 같이 통과하는지 확인한다.

```
BUILD SUCCESSFUL in 0s
```

로또

과제 진행 요구 사항

- 미션은 로또 저장소를 포크하고 클론하는 것으로 시작한다.
- **기능을 구현하기 전 `README.md` 에 구현할 기능 목록을 정리**해 추가한다.
- Git의 커밋 단위는 앞 단계에서 `README.md` 에 정리한 기능 목록 단위로 추가한다.
  - AngularJS Git Commit Message Conventions을 참고해 커밋 메시지를 작성한다.
- 자세한 과제 진행 방법은 프리코스 진행 가이드 문서를 참고한다.

기능 요구 사항

간단한 로또 발매기를 구현한다.

- 로또 번호의 숫자 범위는 1~45까지이다.
- 1개의 로또를 발행할 때 중복되지 않는 6개의 숫자를 뽑는다.
- 당첨 번호 추첨 시 중복되지 않는 숫자 6개와 보너스 번호 1개를 뽑는다.
- 당첨은 1등부터 5등까지 있다. 당첨 기준과 금액은 아래와 같다.
  - 1등: 6개 번호 일치 / 2,000,000,000원
  - 2등: 5개 번호 + 보너스 번호 일치 / 30,000,000원
  - 3등: 5개 번호 일치 / 1,500,000원
  - 4등: 4개 번호 일치 / 50,000원
  - 5등: 3개 번호 일치 / 5,000원
- 로또 구입 금액을 입력하면 구입 금액에 해당하는 만큼 로또를 발행해야 한다.
- 로또 1장의 가격은 1,000원이다.
- 당첨 번호와 보너스 번호를 입력받는다.
- 사용자가 구매한 로또 번호와 당첨 번호를 비교하여 당첨 내역 및 수익률을 출력하고 로또 게임을 종료한다.
- 사용자가 잘못된 값을 입력할 경우 `IllegalArgumentException` 을 발생시키고, "[ERROR]"로 시작하는 에러 메시지를 출력 후 그 부분부터 입력을 다시 받는다.
  - `Exception` 이 아닌 `IllegalArgumentException` , `IllegalStateException` 등과 같은 명확한 유형을

처리한다.

입출력 요구 사항

입력

- 로또 구입 금액을 입력 받는다. 구입 금액은 1,000원 단위로 입력 받으며 1,000원으로 나누어 떨어지지 않는 경우 예외 처리한다.

```
14000
```

- 당첨 번호를 입력 받는다. 번호는 쉼표(,)를 기준으로 구분한다.

```
1,2,3,4,5,6
```

- 보너스 번호를 입력 받는다.

```
7
```

출력

- 발행한 로또 수량 및 번호를 출력한다. 로또 번호는 오름차순으로 정렬하여 보여준다.

```
8개를 구매했습니다.
[8, 21, 23, 41, 42, 43]
[3, 5, 11, 16, 32, 38]
[7, 11, 16, 35, 36, 44]
[1, 8, 11, 31, 41, 42]
[13, 14, 16, 38, 42, 45]
[7, 11, 30, 40, 42, 43]
[2, 13, 22, 32, 38, 45]
[1, 3, 5, 14, 22, 45]
```

- 당첨 내역을 출력한다.

```
3개 일치 (5,000원) - 1개
4개 일치 (50,000원) - 0개
5개 일치 (1,500,000원) - 0개
```

```
5개 일치, 보너스 볼 일치 (30,000,000원) - 0개
6개 일치 (2,000,000,000원) - 0개
```

- 수익률은 소수점 둘째 자리에서 반올림한다. (ex. 100.0%, 51.5%, 1,000,000.0%)

```
총 수익률은 62.5%입니다.
```

- 예외 상황 시 에러 문구를 출력해야 한다. 단, 에러 문구는 "[ERROR]"로 시작해야 한다.

```
[ERROR] 로또 번호는 1부터 45 사이의 숫자여야 합니다.
```

실행 결과 예시

```
구입금액을 입력해 주세요.
8000

8개를 구매했습니다.
[8, 21, 23, 41, 42, 43]
[3, 5, 11, 16, 32, 38]
[7, 11, 16, 35, 36, 44]
[1, 8, 11, 31, 41, 42]
[13, 14, 16, 38, 42, 45]
[7, 11, 30, 40, 42, 43]
[2, 13, 22, 32, 38, 45]
[1, 3, 5, 14, 22, 45]

당첨 번호를 입력해 주세요.
1,2,3,4,5,6

보너스 번호를 입력해 주세요.
7

당첨 통계
---
3개 일치 (5,000원) - 1개
4개 일치 (50,000원) - 0개
5개 일치 (1,500,000원) - 0개
5개 일치, 보너스 볼 일치 (30,000,000원) - 0개
6개 일치 (2,000,000,000원) - 0개
총 수익률은 62.5%입니다.
```

프로그래밍 요구 사항 1

- JDK 21 버전에서 실행 가능해야 한다.
- 프로그램 실행의 시작점은 `Application`의 `main()`이다.
- `build.gradle` 파일은 변경할 수 없으며, **제공된 라이브러리 이외의 외부 라이브러리는 사용하지 않는다.**
- 프로그램 종료 시 `System.exit()`를 호출하지 않는다.
- 프로그래밍 요구 사항에서 달리 명시하지 않는 한 파일, 패키지 등의 이름을 바꾸거나 이동하지 않는다.
- 자바 코드 컨벤션을 지키면서 프로그래밍한다.
  - 기본적으로 Java Style Guide를 원칙으로 한다.

프로그래밍 요구 사항 2

- indent(인덴트, 들여쓰기) depth를 3이 넘지 않도록 구현한다. 2까지만 허용한다.
  - 예를 들어 while문 안에 if문이 있으면 들여쓰기는 2이다.
  - 힌트: indent(인덴트, 들여쓰기) depth를 줄이는 좋은 방법은 함수(또는 메서드)를 분리하면 된다.
- 3항 연산자를 쓰지 않는다.
- 함수(또는 메서드)가 한 가지 일만 하도록 최대한 작게 만들어라.
- JUnit 5와 AssertJ를 이용하여 정리한 기능 목록이 정상적으로 작동하는지 테스트 코드로 확인한다.
  - 테스트 도구 사용법이 익숙하지 않다면 아래 문서를 참고하여 학습한 후 테스트를 구현한다.
    - JUnit 5 User Guide
    - AssertJ User Guide
    - AssertJ Exception Assertions
    - Guide to JUnit 5 Parameterized Tests

프로그래밍 요구 사항 3

- 함수(또는 메서드)의 길이가 15라인을 넘어가지 않도록 구현한다.
  - 함수(또는 메서드)가 한 가지 일만 잘 하도록 구현한다.
- else 예약어를 쓰지 않는다.
  - else를 쓰지 말라고 하니 switch/case로 구현하는 경우가 있는데 switch/case도 허용하지 않는다.
  - 힌트: if 조건절에서 값을 return하는 방식으로 구현하면 else를 사용하지 않아도 된다.
- Java Enum을 적용하여 프로그램을 구현한다.
- 구현한 기능에 대한 단위 테스트를 작성한다. 단, UI(System.out, System.in, Scanner) 로직은 제외한다.
  - 단위 테스트 작성이 익숙하지 않다면 `LottoTest`를 참고하여 학습한 후 테스트를 작성한다.

라이브러리

- `camp.nextstep.edu.missionutils`에서 제공하는 `Randoms` 및 `Console` API를 사용하여 구현해야 한다.
  - Random 값 추출은 `camp.nextstep.edu.missionutils.Randoms`의 `pickUniqueNumbersInRange()`를 활용한다.

- 사용자가 입력하는 값은 `camp.nextstep.edu.missionutils.Console`의 `readLine()`을 활용한다.

사용 예시

- 1에서 45 사이의 중복되지 않은 정수 6개 반환

```
Randoms.pickUniqueNumbersInRange(1, 45, 6);
```

Lotto 클래스

- 제공된 `Lotto` 클래스를 사용하여 구현해야 한다.
- `Lotto`에 `numbers` 이외의 필드(인스턴스 변수)를 추가할 수 없다.
- `numbers`의 접근 제어자인 `private`은 변경할 수 없다.
- `Lotto`의 패키지를 변경할 수 있다.

```
public class Lotto {
    private final List<Integer> numbers;

    public Lotto(List<Integer> numbers) {
        validate(numbers);
        this.numbers = numbers;
    }

    private void validate(List<Integer> numbers) {
        if (numbers.size() != 6) {
            throw new IllegalArgumentException("[ERROR] 로또 번호는 6개여야 합니다.");
        }
    }

    // TODO: 추가 기능 구현
}
```