목표

학습 목표

- 여러 역할을 수행하는 큰 함수를 단일 역할을 수행하는 작은 함수로 분리한다.
- 테스트 도구를 사용하는 방법을 배우고 프로그램이 제대로 작동하는지 테스트한다.
- 1주 차 공통 피드백을 최대한 반영한다.

회고

아래 질문에 대한 중간 회고를 진행하고 소감에 구체적인 결과를 작성한다. 소감은 텍스트로 작성해야 하며 외부 링크는 허용하지 않는다.

- 지원서에 작성한 목표를 얼마나 달성하고 있다고 생각하나요? 그 이유는 무엇인가요?
- 지원서에 작성한 목표를 변경해야 한다고 생각하시나요? 그렇다면 그 이유와 어떤 목표로 변경하고 싶으신가요?
- 프리코스를 진행하면서 눈에 띄는 변화나 깨달은 점이 있나요?

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
```

자동차 경주

과제 진행 요구 사항

- 미션은 자동차 경주 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

초간단 자동차 경주 게임을 구현한다.

- 주어진 횟수 동안 n대의 자동차는 전진 또는 멈출 수 있다.
- 각 자동차에 이름을 부여할 수 있다. 전진하는 자동차를 출력할 때 자동차 이름을 같이 출력한다.
- 자동차 이름은 쉼표(,)를 기준으로 구분하며 이름은 5자 이하만 가능하다.
- 사용자는 몇 번의 이동을 할 것인지를 입력할 수 있어야 한다.
- 전진하는 조건은 0에서 9 사이에서 무작위 값을 구한 후 무작위 값이 4 이상일 경우이다.
- 자동차 경주 게임을 완료한 후 누가 우승했는지를 알려준다. 우승자는 한 명 이상일 수 있다.
- 우승자가 여러 명일 경우 쉼표(,)를 이용하여 구분한다.
- 사용자가 잘못된 값을 입력할 경우 `IllegalArgumentException`을 발생시킨 후 애플리케이션은 종료되어야 한다.

입출력 요구 사항

입력

- 경주할 자동차 이름(이름은 쉼표(,) 기준으로 구분)

```
pobi,woni,jun
```

- 시도할 횟수

```
5
```

출력

- 차수별 실행 결과

```
pobi : --
woni : ----
jun : ---
```

- 단독 우승자 안내 문구

```
최종 우승자 : pobi
```

- 공동 우승자 안내 문구

```
최종 우승자 : pobi, jun
```

실행 결과 예시

```
경주할 자동차 이름을 입력하세요.(이름은 쉼표(,) 기준으로 구분)
pobi,woni,jun
시도할 횟수는 몇 회인가요?
5

실행 결과
```

```
pobi : -
woni :
jun : -

pobi : --
woni : -
jun : --

pobi : ---
woni : --
jun : ---

pobi : ----
woni : ---
jun : ----

pobi : -----
woni : ----
jun : -----

최종 우승자 : pobi, jun
```

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

라이브러리

- `camp.nextstep.edu.missionutils`에서 제공하는 `Randoms` 및 `Console` API를 사용하여 구현해야 한다.
  - Random 값 추출은 `camp.nextstep.edu.missionutils.Randoms`의 `pickNumberInRange()`를 활용한다.
  - 사용자가 입력하는 값은 `camp.nextstep.edu.missionutils.Console`의 `readLine()`을 활용한다.

사용 예시

- 0에서 9까지의 정수 중 한 개의 정수 반환

```
Randoms.pickNumberInRange(0, 9);
```