프리코스 진행방식

진행 방식

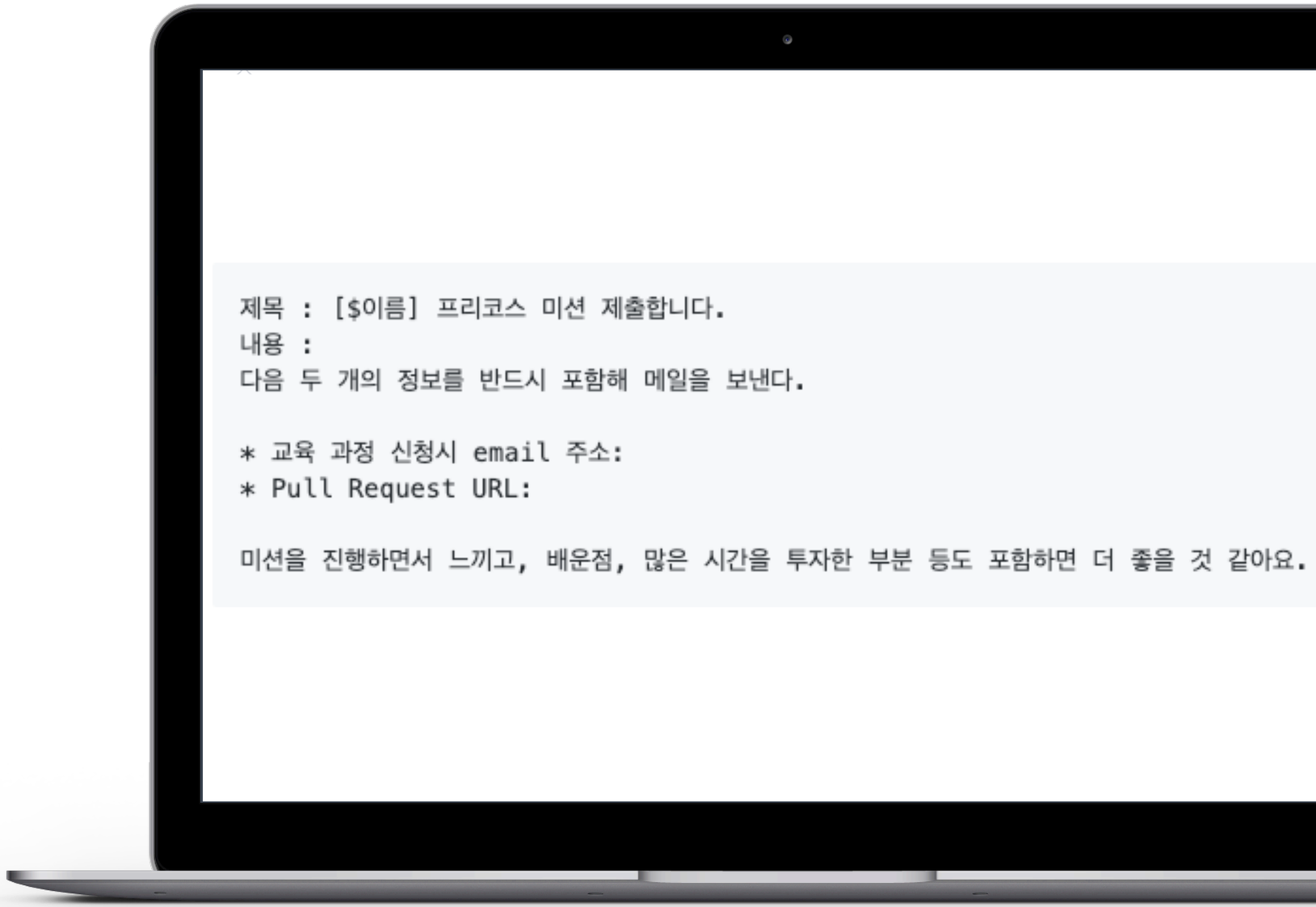
- 미션은 **기능 요구사항, 프로그래밍 요구사항, 과제 진행 요구사항** 세 가지로 구성되어 있다.
- 세 개의 요구사항을 만족하기 위해 노력한다. 특히 기능을 구현하기 전에 **기능 목록을 만들고, 기능 단위로 commit하는 방식**으로 진행한다.

미션 제출 방법

- 미션 구현을 완료한 후 GitHub을 통해 제출해야 한다.
  - GitHub을 활용한 제출 방법은 프리코스 과제 제출 문서 참고해 제출한다.
- GitHub에 미션을 제출한 후 edu.nextstep@gmail.com 로 메일을 발송한다.

email 템플릿

- 제목 양식
- 본문 양식
  - 교육 과정 신청시 이메일 주소
  - Pull Request URL
  - 미션 진행 중 느낀점, 배운점



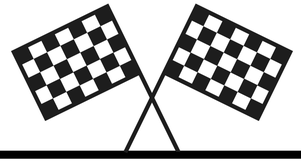
email 예시

새 메일

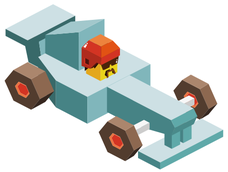
받는사람 edu.nextstep@gmail.com
제목 **[박재성] 숫자야구게임 미션 제출합니다.**

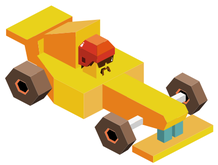
안녕하세요.
이번 미션 생각보다 쉽지 않네요.
특히 기능을 분리하고 기능 단위로 commit하는 것이 쉽지 않다는 것을 느꼈어요.
기능 단위로 분리하고 commit하기 위해 이런 이런 과정으로 연습했어요.
하지만 기능을 분리하고 구현했더니 더 명확하게 구현할 수 있다는 것을 느낄 수 있는 기회가 된 것 같아요.

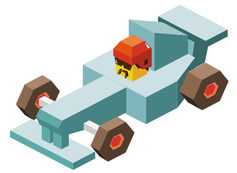
* **Email 주소**: edu.nextstep@gmail.com
* **Pull Request URL**: https://github.com/next-step/java-baseball-precourse/pull/1

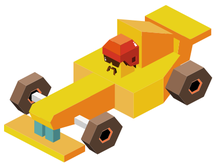


2주차 미션 - 자동차 경주 게임









기능 요구사항

- 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 random 값을 구한 후 random 값이 4 이상일 경우 전진하고, 3 이하의 값이면 멈춘다.
- 자동차 경주 게임을 완료한 후 누가 우승했는지를 알려준다.
- 우승자가 한 명 이상일 경우, 쉼표(,)로 이름을 구분해 출력한다.
- 사용자가 잘못된 값을 입력할 경우 “[ERROR]”로 시작하는 에러 메시지를 출력 후 입력을 다시 받는다.

프로그램 실행 결과

Run: RacingCar

경주할 자동차 이름을 입력하세요.(이름은 쉼표(,) 기준으로 구분)
pobi,crong,honux
시도할 회수는 몇회인가요?
5

실행 결과
pobi : -
crong :
honux : -

pobi : --
crong : -
honux : --

pobi : ---
crong : --
honux : ---

pobi : ----
crong : ---
honux : ----

pobi : -----
crong : ----
honux : -----

최종 우승자는 pobi,honux 입니다.

프로그래밍 요구사항1 - 제약사항

- **자동차 경주 게임을 실행하는 시작점은 src/main/java 폴더의 racinggame.Application의 main()이다.**
- **자동차 경주 게임은 JDK 8 버전에서 실행가능해야 한다. JDK 8에서 정상 동작하지 않을 경우 0점 처리한다.**
- **JDK에서 기본 제공하는 Random, Scanner API 대신 nextstep.utils 패키지에서 제공하는 Randoms, Console API를 활용해 구현해야 한다.**
  - Random 값 추출은 nextstep.utils.Randoms의 pickNumberInRange()를 활용한다.
  - 사용자가 입력하는 값은 nextstep.utils.Console의 readLine()을 활용한다.
- **프로그램 구현을 완료했을 때 src/test/java 폴더의 racinggame.ApplicationTest에 있는 2개의 Test Case가 성공해야 한다.**
  - ApplicationTest에서 제공하는 2개의 Test Case는 자동차 경주 게임을 위한 최소한의 Test Case이다.
  - 필수 요구사항은 아니지만 제공하는 소스 코드를 참고해 자동차 경주 게임을 위한 모든 Test Case를 추가해 보는 것도 테스트에 대한 좋은 연습이 될 수 있다.

프로그래밍 요구사항2 - 1주차와 동일한 기준

- **자바 코드 컨벤션을 지키면서 프로그래밍**한다.
  - https://naver.github.io/hackday-conventions-java/
- **indent(인덴트, 들여쓰기) depth를 2가 넘지 않도록 구현한다. 1까지만 허용**한다.
  - 예를 들어 while문 안에 if문이 있으면 들여쓰기는 2이다.
  - 힌트: indent(인덴트, 들여쓰기) depth를 줄이는 좋은 방법은 함수(또는 메소드)를 분리하면 된다.
- **자바 8에 추가된 stream api를 사용하지 않고 구현해야 한다. 단, 람다는 사용 가능하다.**
- **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.
- **함수(또는 메소드)의 길이가 10라인을 넘어가지 않도록 구현한다.**
  - 함수(또는 메소드)가 한 가지 일만 잘 하도록 구현한다.

프로그래밍 요구사항2 - 2주차 추가

- **일급콜렉션을 활용해 구현한다.**
  - 참고문서: https://developerfarm.wordpress.com/2012/02/01/object_calisthenics_/
- **모든 원시값과 문자열을 포장한다.**
  - 참고문서: https://developerfarm.wordpress.com/2012/01/27/object_calisthenics_4

프로그래밍 요구사항3 - 단위 테스트

- **도메인 로직에 단위 테스트를 구현해야 한다. 단, UI(System.out, System.in, Scanner) 로직은 제외**
  - 핵심 로직을 구현하는 코드와 UI를 담당하는 로직을 분리해 구현한다.
  - 힌트는 MVC 패턴 기반으로 구현한 후 View, Controller를 제외한 Model에 대한 단위 테스트를 추가하는 것에 집중한다.
- JUnit5와 AssertJ 사용법에 익숙하지 않은 개발자는 첨부한 "학습테스트를 통해 JUnit 학습하기.pdf" 문서를 참고해 사용법을 학습한 후 JUnit5 기반 단위 테스트를 구현한다.

미션 저장소 및 진행 요구사항

- 미션은 https://github.com/next-step/java-racingcar-precourse 저장소를 fork/clone해 시작한다.
- **기능을 구현하기 전에 java-racingcar-precourse/README.md 파일에 구현할 기능 목록**을 정리해 추가한다.
- **git의 commit 단위는 앞 단계에서 README.md 파일에 정리한 기능 목록 단위로 추가**한다.
  - AngularJS Commit Message Conventions 참고해 commit log를 남긴다.
- 과제 진행 및 제출 방법은 프리코스 과제 제출 문서를 참고한다.

2차 미션 마감 및 기준

- **2021년 10월 13일(수) 23시 59분**까지 GitHub을 통한 미션 제출과 메일전송까지 완료해야 한다
- **2021년 10월 14일(목) 00시 이후 추가 push도 허용하지 않는다.**
- **2021년 10월 14일(목) 00시 이후 제출한 경우 미션을 제출하지 않은 것**으로 한다.