1 Original TripService 에서 개선이 필요한 점들을 기술하세요.
- Singleton을 사용하여 타 클래스와 dependency가 존재
- TripDAO에서 사용한 Static함수 때문에 테스트가 힘든점
- 과도한 if문과 for문 사용으로 가독성 저하

2 레거시 코드에 테스트를 작성할 때에 인덴트가 얕은 부분부터 작성하는 이유는 무엇인가요?
- 인덴트가 얕은 부분이 다른 코드에 대한 디펜던시가 적다 반면 깊은 부분은 얕은 부분이 선처리 되어야 진행될수 있는 경우가 잦다

3 리팩토링 할 때에 코드의 인덴트가 깊은 부분부터 하는 이유는 무엇인가요?
- 재사용할 수 있는부분이 깊은 부분일 수록 많다.
- 위에서 말했듯이 깊은 부분은 얕은부분이 선처리 되어야 진행될 수 있는 경우가 있는데 얕은 부분부터 시작하게 되면 깊은 부분에서 사용되는 부분을 고려하지 못한체 리팩토링을 진행하게 될 수 있다.

4 레거시 코드에 단위 테스트를 작성하기 어려운 이유는 어떤 것들이 있을까요?
- 클래스들간 Decoupling이 원활하게 되어있지 않은 경우
- Final 과 static처럼 mock을 하기 어려운 경우
- 이미 짜여진 소스를 토대로 이를 이해해가며 테스트를 짜는건 굉장히 많은 시간이 걸린다
- 반대로 테스트를 선수행하고 소스를 작성한다면 반대보다 훨씬 적은 시간이 걸린다

5 단위테스트에서 DAO를 직접 호출하지 않는 이유는 어떤 것들이 있을까요?
- 단위테스트는 말그대로 단위테스트, DAO테스트는 따로 수행되어야한다
- 다른 클래스를 단위 테스트 하는데 DAO가 포함되어있다면 DAO와 해당 클래스 사이의 존재할 수 있는 모든 로직 경우의 수를 테스트 해줘야 하는데 단위를 클래스와 DAO를 나눠놓으면 해당 클래스에서 필요한 로직만 DAO를 mock으로 처리한다음에 테스트를 진행할 수 있기때문에 테스트가 훨씬 간결해진다.

6 시연 중에 코드 커버리지 툴은 어떤 용도로 사용되고 있나요?
- 의미있는 로직인데 미처 테스트처리하지 못한 부분이 있나 확인하는 용도

7 시연 중에 변경 범위가 큰 위험한 리팩토링을 한 이유는 무엇일까요? (50:20~57:04)
- UserSession이라는 싱글톤 객체에 대해 tripService클래스와 디펜던시를 분리하고 대신 해당 정보를 변수로 받기 위해서

8 시연에서 사용된 단위 테스트의 명명 규칙에 대해 기술하세요
- 먼저 should로 시작해 어떤결과가 발생하는지 기술하고 when이나 if같은 문을 써서 condition을 설명한다
- 되도록 실제 소스코드와 동일한 위에서 아래로 순서를 맞춰준다

9 클린 코드는 “잘 작성된 산문”과 같은 코드라고도 말합니다. 시연에서 잘 작성된 산문과 같은 코드의 사 례를 찾아서 기술하세요.
- 어느 특정 유저의 친구 리스트를 for loop에서 돌면서 로그인 한 유저가 특정 유저와 친구인지를 판별하는 로직은 isFriendWith라는 함수로 간결하게 처리해 주었다.
- 두 사람이 친구면 친구에게 속한 여행지를 return하는 tripsBy(user)라는 함수와 친구가 아니라면 빈 어레이를 return하는 noTrips()함수를 통해 가독성을 높였다
- 로그인 유저가 null인지 확인하는 로직대신 vaildate(user)라는 함수로 이해하기 쉽게 해주었다.