This exercise introduces you to dependency injection and refactoring. You will the snakes and ladders game.

Task 1: Mark methods as for testing only##

If methods are used by tests only mark them with a custom @ForTestingOnly annotation that you need to provide.

- Provide a `ForTestingOnly` annotation.

Hint: use "Open call hierarchy" to learn about the callers of a method—or (as some professional developers do it) just make the method private, hit save and wait for compile time errors. Then act as follows

- If there are compile errors redeclare the method as public.
- If the errors occurred only in tests, mark the method as "for testing" only.
- If there are no compile errors, keep the method private.
- If a warning appears that the method is unused, remove it.

Task 2: Dependency injection

ACME Inc. is between projects and desperately in the need for problems to solve. Their eyes fell upon the Snakes and ladders game, which still isn't using dependency injection.

As their top consultant, you've been called to the rescue. Your new supervisor, a guy who does not even know whether it's pull or push that gets new code from the server, has heard about a wonderful technique that they use at Google. As he informs you, it's called Guice, and what works for Google ought to work for you! In his explanation of Guice, your supervisor clearly mixes up a few terms, but you manage to understand that he wants to inject, as it were, a die into the game. The Game isn't supposed to create its own die. After you asked back a few times, you gather that the advantage is that the `Game#play` method does not need to have a parameter, but still, a mock die can be used in unit tests.

As you go along, you realize that referring to a unit test from within the game, as in the `Game#main` method isn't good style. Probably, there should be a class whose sole responsibility is to create a Game.

Task 2.1: Make yourself familiar with dependency injection

Work your way through the Guice documentation to get familiar with the topic of dependency injection and Guice. You may find the relevant google tech talk helpful.

Task 2.2: Refactor snakes and ladders towards dependency injection

- Rewrite The `Game#play` method such that it does not accept a parameter, but instead uses an instance variable die, which is injected by Google Guice.
- Make a Guice Provider that constructs the Game similar to `SimpleGameTest#newGame`.
- Make a SnakesAndLaddersModule. Modify the code such that the provider is used to assemble the squares of the game.
- Make sure that all unit tests, including the tests that use a mock die, exist and continue to work.

Remarks

- If you did not hand in problemset02, you can ask for somebody else's code to refactor.
- Please solve this exercise in the problemset02 folder.