Challenge: Place your expert in the puzzle world, and solve the puzzle.

You can now place the expert in a specific *location*, but what if you want the expert to face a specific *direction*, as well? You can call a different version of the `place` method that takes a direction as an additional argument.

> Specifying a direction for your expert
>
> `world.place(expert, facing: .west, atColumn: 6, row: 3)`

But wait—what does `.west` mean? Think of it as a shorter form of dot notation that gives you a group of options to choose from. In this case, you can choose `.west`, `.east`, `.north`, or `.south`, but nothing else!

Choices like this work because each choice is of the same type—an enumeration—that defines that group of related values. You *could* write each choice like `Direction.north`, for example, but you can also leave out `Direction` to make it simpler.

Start by initializing your expert. Then find `world` in the shortcut bar and add it to your code. Use dot notation to call the `place` method that includes the `facing` parameter, and pass in your arguments. Then solve the puzzle.

```
let expert = Expert()
world.place(expert, facing: west, atColumn: 2, row: 8)
var gem = 0
while gem != 9 {
    if expert.isBlocked && gem == 4 {
        expert.turnLockDown()

    } else if expert.isBlocked && gem == 6 {
        expert.turnLockUp()

    }
    if expert.isBlocked {
        if expert.isBlockedLeft {
            expert.turnRight()

        } else {
            expert.turnLeft()
```

```
        }
    }
    expert.moveForward()
    if expert.isOnGem {
        expert.collectGem()
        gem += 1
    }
}
```