# Smelly Mars Rover movement refactoring.

## Description

In this session we'll refactor a particularly smelly implementation of the movement of the rover from the Mars Rover kata.

This is a summary of the behavior of the rover that was implemented:

- It's located on a grid at some point with coordinates (x,y) and facing a direction encoded with a character.

- The meaning of each direction character is:
  - `N` -> North
  - `S` -> South
  - `E` -> East
  - `W` -> West
- The rover receives a sequence of commands (a string of characters) which are codified in the following way:
  - When it receives an `f`, it moves forward one position in the direction it is facing.
  - When it receives a `b`, it moves backward one position in the direction it is facing.

    - When it receives a `l`, it turns left changing its direction (by 90º).
    - When it receives a `r`, it turns right changing its direction (by 90º).

The initial code was created using TDD, but the coders unfortunately skipped the refactoring step of TDD.

For this reason, even though, the code is fully tested and behaves as expected, it’s full of communication and maintanability problems.

## Goals

1. Start to be able to reconigze some basic code smells from the catalog of code smells from *Martin Fowler*'s **Refactoring, Improving the Design of Existing Code** book (we don’t tell them yet so you can try to first guess yourself).

2. Practice the mechanics of several refactoring techniques from the same book which are useful to eliminate those smells, such as, Extract Method, Replace Magic Number with Symbolic Constant, Replace Constructor with Factory Method, Replace Conditional with Polymorphism, Replace Type Code with State/Strategy, Self Encapsulate Field, Move Method, Rename, Inline Method and Decompose Conditional, and some other ones that are not in the book like Reverse Conditional.

3. Learn about the State pattern.

## How we'll do it?

1. Before the session read the description of what the code should do and then have a look at the initial code, then answer the following questions:
    - What do you think is wrong with this code?
    - Does it communicate its intent well? If not, think why?
    - Are its responsibilities explicit and well segregated?
    - Is there any duplication?
    - Which code smells do you recognize?
2. At the beginning of the session we'll discuss a bit about what you've found looking at the code.
3. Then we'll give you some guidelines and you'll start refactoring the code to remove the smells.