# Patrones de diseño

El principal patrón aplicado en el TP (era uno de los objetivos), fue el **MVC**. Hicimos un esfuerzo enorme para que nos quedaran bien discriminadas las partes en distintos paquetes.

* control: están prácticamente todo aquello que implementa ActionListener. Además, hay unas clases que controlan el desarrollo del juego
* modelo: ahí está nuestro modelo de dominio entero. Hay una clase importante, modelo.Entorno, que representa la relación entre la pista y el auto. Le da un sentido físico al juego. Además, está el paquete modelo.servicio, donde se alocan las clases que solo nos interesan por su funcionalidad.
* vista: paquete compuesto por las ventanas, y por la representación gráfica de cada elemento del dominio (auto, loma de burro, etc.)

También tenemos el paquete titiritero, que tiene una mezcla de control y vista.

Otro patrón aplicado fue el **double dispatch** al momento de chocar con un obstáculo, dado que el desgaste producido a los neumáticos varía dependiendo del tipo de ruedas y del obstáculo. Básicamente, cuando la rueda se entera que chocó, le informa al obstáculo que está siendo chocado, y éste contesta diciéndole a la rueda con quién está chocando. Más formal está mostrado en un diagrama de secuencia.

El último patrón implementado conscientemente fue el de las fábricas. Son dos, en realidad: **abstract** **factory** y **factory**, pero porque la primera se apoya en la segunda. Las usamos para la creación de los autos. Podría haber sido de instancia única (singleton), pero preferimos la idea de que, para crear al auto, no se necesitara más que un método. Conseguimos un auto recién salido del horno con sólo hacer:

Automovil auto = FabricaDeAutos.instanciarAuto(metadataAuto);

Las subfábricas abstractas (de motores y ruedas) tienen un mapa que guarda la clave, y la referencia a una fábrica concreta. Una forma más fácilmente extensible de hacer esto podría haber sido con reflexión, y nos ahorrábamos un par de fábricas.

Indirectamente, usamos patrones similares al **observer** (desde el MVC)**, command** (desde el menú, donde todos implementan ActionListener) e **iterator** (para iterar sobre las listas de terrenos, obstáculo, etc, en la pista), pero no merecen mayor mención.