|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE | Programación Avanzada | | |
| AREA | Software | REGIMEN | Semestral |

**Tema: Patrones de diseño**

**Método Plantilla**

Define una estructura algorítmica cuya lógica quedará a cargo de las subclases. Para ello, escribe una clase abstracta que contiene parte de la lógica necesaria para realizar su finalidad. En ella se define una estructura de herencia que sirve de plantilla ("Template" significa plantilla) de los métodos en las subclases.

Dicho de otra forma, define un esqueleto de un algoritmo, delegando algunos pasos a las subclases. Permite redefinir parte de dicho algoritmo sin cambiar su estructura.

Se debe utilizar este patrón cuando.

* Se quiera factorizar el comportamiento común de varias subclases.
* Se necesite implementar las partes fijas de un algoritmo una sola vez y dejar que las subclases implementen las partes variables.
* Se busque controlar las ampliaciones de las subclases, convirtiendo en métodos plantillas aquéllos métodos que pueden ser redefinidos.

Este patrón se vuelve de especial utilidad cuando es necesario realizar un algoritmo que sea común para muchas clases, pero con pequeñas variaciones entre una y otras. En este caso, se deja en las subclases cambiar una parte del algoritmo.

**Vista estática: diagrama de clases**

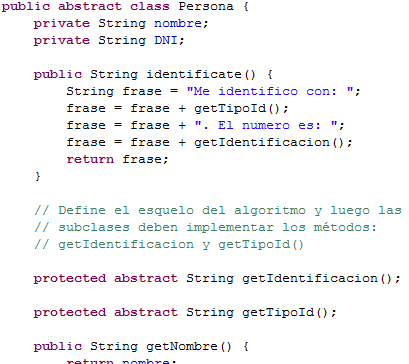


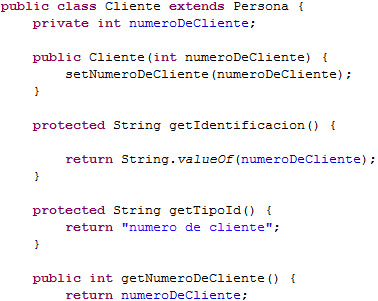
AbstractTemplate o AbstractClass: implementa un método plantilla que define el esqueleto de un algoritmo y define métodos abstractos que deben implementar las subclases concretas

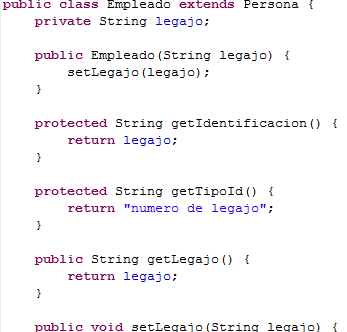
TemplateConcreto o ConcreteClass: implementa los métodos abstractos para realizar los pasos del algoritmo que son específicos de la subclase.

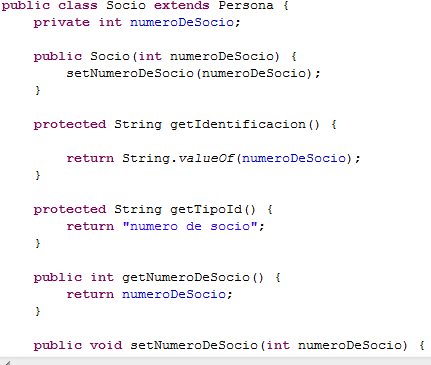
***Caso práctico a desarrollar en clase***

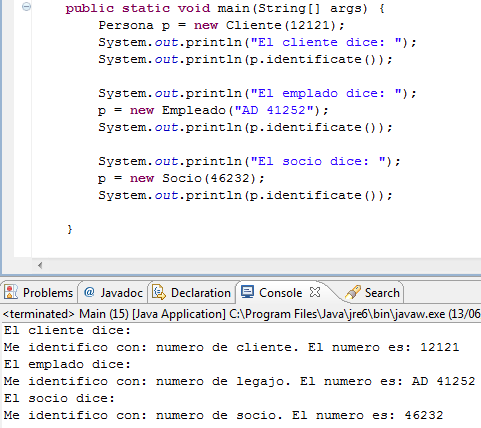
Como ejemplo, imaginemos una empresa que posee socios, clientes, empleados, etc. Cuando se les solicite que se identifiquen, cada uno lo realizará de distinta manera: quizás un empleado tiene un legajo, pero un cliente tiene un número de cliente, etc.











**Consecuencias**

* Favorece la reutilización del código.
* Lleva a una estructura de control invertido: la superclase base invoca los métodos

de las subclases.