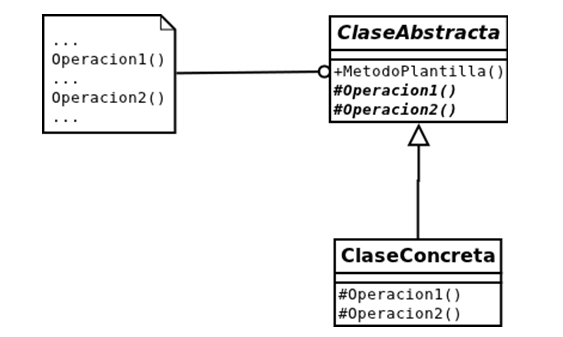
# Descripción del patrón

El patrón de diseño “Método Plantilla” forma parte de la familia de patrones denominados de comportamiento. Este tipo patrones ayudan a resolver problemas de interacción entre clases y objetos.

Este patrón nace de la necesidad de extender determinados comportamientos dentro de un mismo algoritmo por parte de diferentes entidades. Es decir, diferentes entidades tienen un comportamiento similar pero que difiere en determinados aspectos puntuales en función de la entidad concreta.

Una posible solución podría ser copiar el algoritmo en cada de las diferentes entidades cambiando la parte concreta en la que difieren. Esta solución tiene una consecuencia negativa ya que se genera código duplicado.

La solución que propone el patrón “Método Plantilla” es abstraer todo el comportamiento que comparten las entidades en una clase (abstracta) de la que, posteriormente, extenderán dichas entidades. Esta superclase definirá un método que contendrá el esqueleto de ese algoritmo común (método plantilla) y delegará determinada responsabilidad en las clases hijas, mediante uno o varios métodos abstractos que deberán implementar.

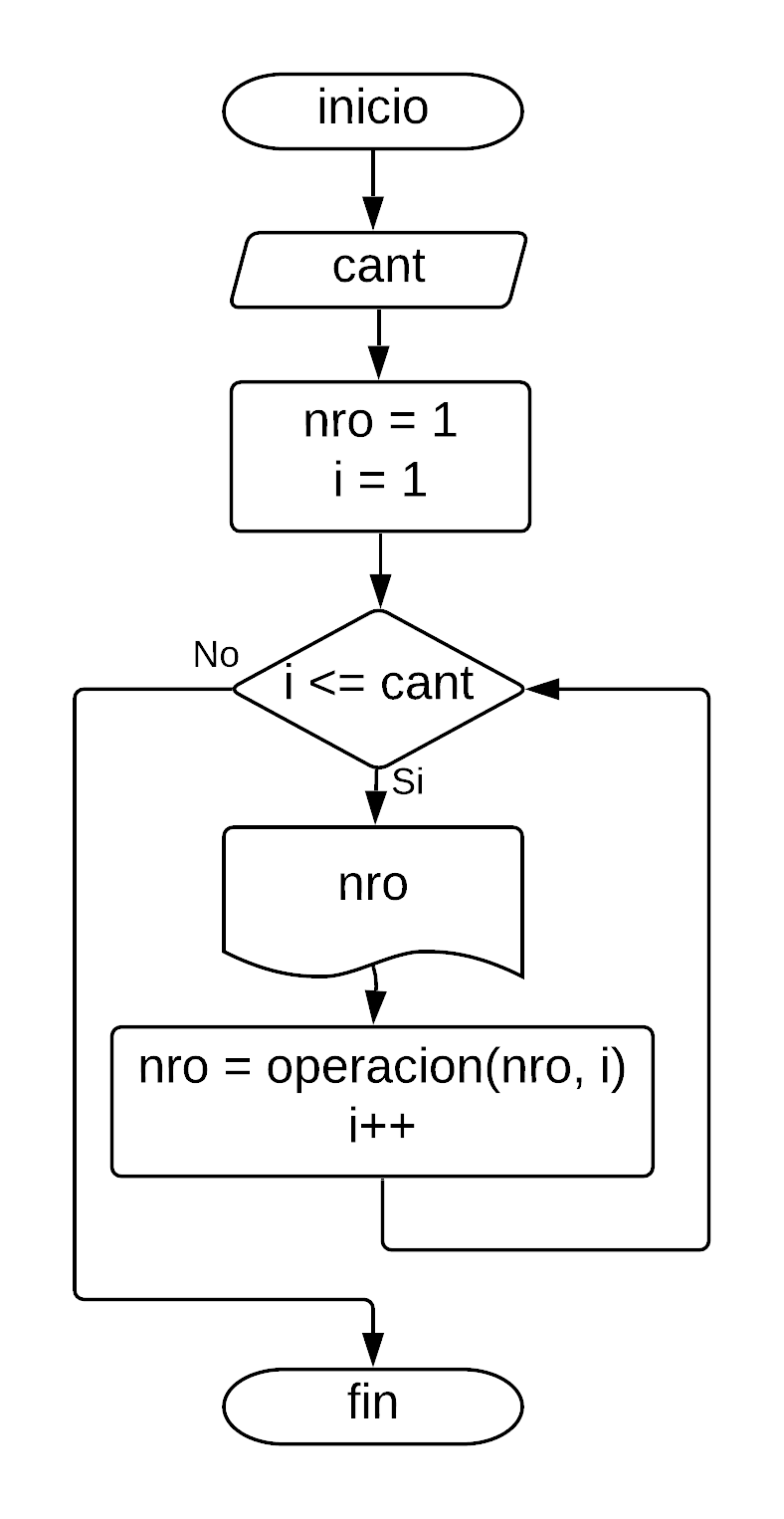


Como puede verse en el anterior diagrama, la superclase contiene el método plantilla, ese método con el algoritmo que comparten las entidades concretas (subclases). Como se puede apreciar, define una o varias operaciones concretas en forma de métodos abstractos que son usados por el método plantilla y que deben ser implementadas por las clases hijas. Dichos métodos abstractos representan los comportamientos concretos de las entidades.

# Aplicando el patrón

Nuestro caso de estudio es sencillo. Se desea crear distintas series de números a partir del número uno hasta una cantidad N de elementos (**S1**: [1, 2, 3, 4], **S2**: [1, 10, 100, 1000], **S3**: [1, 4, 9, 16], **S4**: [1, 3, 5, 7]…). Para ello habría dos soluciones posibles: primera, crear una clase para generar cada tipo de serie, copiando en cada una la misma estructura del código con el único en la forma de hallar el siguiente elemento de la serie numérica. O, podemos reutilizar la estructura genérica de las series con el patrón de diseño Método Plantilla haciendo que cada tipo de serie en vez de copiar el mismo código solo se dedique a modificar la obtención del siguiente elemento de la serie.

Primero crearemos una clase abstracta a la cual llamaremos Serie. Esta clase tendrá dos métodos uno público llamado generarSerie() el cual se encontrará la estructura base de cualquier serie que queramos implementar.



Además también, nuestra clase abstracta tendrá una función abstracta y protegida llamada operacion(int nro, int i) el cual nos devolverá el nuevo elemento de la serie numérica. Esta es la función que debe ser implementada en las subclases y la cual distinguirá a una serie de otra.

Segundo, se crearán las subclases que extenderán o (heredarán) de la clase abstracta Serie. Al heredar de esta clase deben obligatoriamente implementar o sobrescribir la función abstracta operacion(int nro, int i). En esta operación encontraremos un nuevo elemento de la serie a partir de una ecuación matemática usando uno o ambos datos proporcionado.