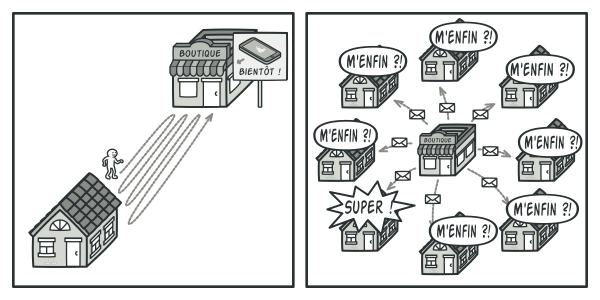
# Intention / Première approche :

L’Observateur est un patron de conception comportemental qui permet de mettre en place un mécanisme de souscription pour envoyer des notifications à plusieurs objets, au sujet d’événements concernant les objets qu’ils observent.

# Problème

Imaginez que vous avez deux types d’objets : un client et un magasin. Le client s’intéresse à un produit spécifique qui sera bientôt disponible dans le magasin. Le client pourrait se rendre sur place tous les jours et vérifier la disponibilité d’un nouveau produit. Mais comme le produit n’est pas encore prêt, ses allées et venues seraient inutiles.



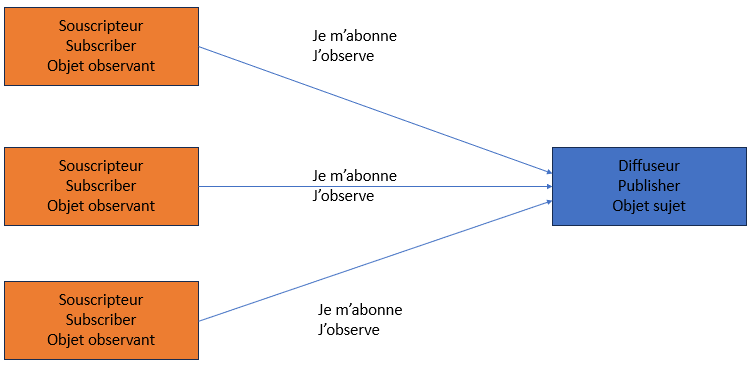
À la place, le magasin pourrait envoyer des tonnes de courriels (ce qui peut être vu comme du spam) à leurs clients chaque fois qu’un nouveau produit est disponible. Cette solution économiserait bien des voyages à leurs clients. En contrepartie, le magasin risque de se mettre à dos ceux qui ne sont pas intéressés par les nouveaux produits.

Nous nous retrouvons dans une situation conflictuelle. Soit les clients perdent leur temps à venir vérifier la disponibilité des produits, soit le magasin gâche des ressources pour prévenir des clients qui ne sont pas concernés.

# Solution

L’objet que l’on veut suivre est en général appelé sujet, mais comme il va envoyer des notifications pour prévenir les autres objets dès qu’il est modifié, nous l’appellerons diffuseur (publisher). Tous les objets qui veulent suivre les modifications apportées au diffuseur sont appelés des souscripteurs (subscribers).

Le patron de conception Observateur vous propose d’ajouter un mécanisme de souscription à la classe diffuseur (publisher) pour permettre aux objets individuels de s’inscrire ou se désinscrire de ce diffuseur. Pas d’inquiétude ! Ce n’est pas si compliqué que cela en a l’air. En réalité, ce mécanisme est composé 1) d’un tableau d’attributs qui stocke une liste de références vers les objets souscripteur et 2) de plusieurs méthodes publiques qui permettent d’ajouter ou de supprimer des souscripteurs de cette liste.



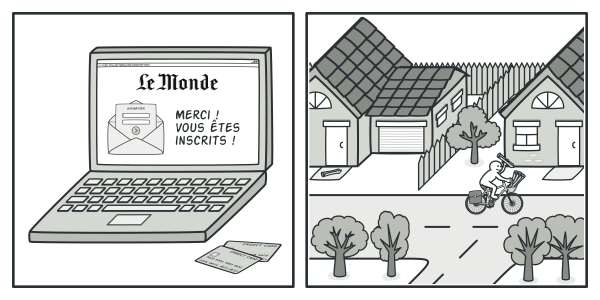
Quand un événement important arrive au diffuseur, il fait le tour de ses souscripteurs et appelle la méthode de notification sur leurs objets.

Les applications peuvent comporter des dizaines de classes souscripteur différentes qui veulent être tenues au courant des événements qui affectent une même classe diffuseur. Vous n’avez sûrement pas envie de coupler le diffuseur à toutes ces classes. De plus, certaines ne seront peut-être pas connues à l’avance, dans les cas où votre classe diffuseur est censée pouvoir être utilisée par d’autres personnes.

C’est pourquoi il est crucial que tous les souscripteurs implémentent la même interface et qu’elle soit le seul moyen utilisé par le diffuseur pour communiquer avec eux. Elle doit déclarer la méthode de notification avec un ensemble de paramètres que le diffuseur peut utiliser pour envoyer des données contextuelles avec la notification.

De plus, les diffuseurs doivent tous suivre la même interface si votre application en comporte plusieurs types et que vous voulez que vos souscripteurs soient tous compatibles avec eux. Cette interface doit contenir quelques méthodes de souscription et elle doit permettre aux souscripteurs d’observer les états du diffuseur sans le coupler avec leurs classes concrètes.

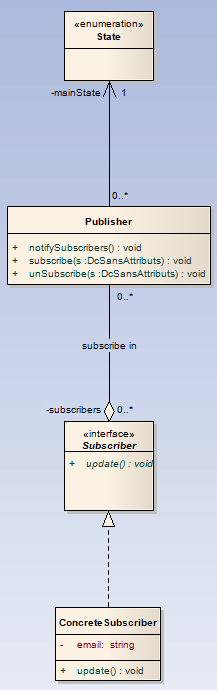
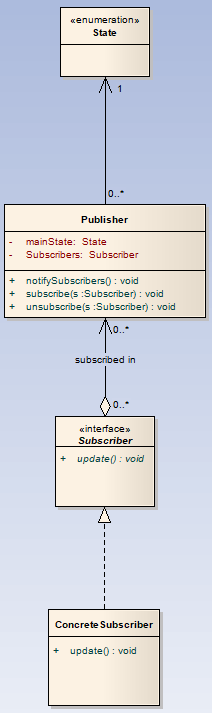
# Analogie



Lorsque vous vous inscrivez à un journal ou à un magazine, vous n’avez plus besoin de vous rendre en magasin pour vérifier si le dernier numéro est sorti. À la place, le diffuseur vous envoie directement les nouveaux numéros dans votre boîte aux lettres dès qu’ils le publient, ou même parfois en avance.

Le diffuseur garde une liste de souscripteurs et connait les magazines qui les intéressent. S’ils ne souhaitent plus recevoir les nouveaux numéros, les souscripteurs peuvent quitter la liste à n’importe quel moment.

# Structure

1. Le Diffuseur envoie des événements intéressants à d’autres objets. Ces événements se produisent quand le diffuseur change d’état ou exécute certains comportements. Le diffuseur possède une infrastructure d’inscription qui permet aux nouveaux souscripteurs de rejoindre la liste et aux souscripteurs actuels de la quitter.
2. Le Diffuseur envoie des événements intéressants à d’autres objets. Ces événements se produisent quand le diffuseur change d’état ou exécute certains comportements. Le diffuseur possède une infrastructure d’inscription qui permet aux nouveaux souscripteurs de rejoindre la liste et aux souscripteurs actuels de la quitter.
3. L’interface Souscripteur déclare les méthodes de notification. Dans la majorité des cas, il n’y a qu’une seule méthode update. Elle peut prendre plusieurs paramètres pour que le diffuseur leur envoie plus de détails concernant la modification.
4. Les Souscripteurs Concrets exécutent certaines actions en réponse aux notifications envoyées par le diffuseur. Toutes ces classes doivent implémenter la même interface pour ne pas coupler le diffuseur avec leurs classes concrètes.
5. En général, les souscripteurs ont besoin de détails à propos du contexte afin d’exécuter correctement la mise à jour. C’est pour cela que les diffuseurs passent souvent des données du contexte en paramètre de la méthode de notification. Le diffuseur peut même s’envoyer lui-même en paramètre et laisser les souscripteurs récupérer directement les données nécessaires.
6. Le Client crée des objets diffuseur et Souscripteur séparément et inscrit les souscripteurs aux mises à jour du diffuseur.