



# Documentation d'ajout de filtre

Ce document décrit les étapes nécessaires à l'ajout d'un filtre au logiciel lovmi. Le logiciel a été conçu de manière à faciliter cette opération. En java, il est possible de déclarer une interface décrivant un certain nombre de méthodes. Une classe voulant respecter cette interface devra obligatoirement implémenter toutes les méthodes qui y sont décrites.

Les sources et les packages se trouvent dans le répertoire src du dossier lovmi. Tous les packages commencent par `fr.uml.v.lovmi`.

Nous avons donc décrit une interface `Filter`, qui se trouve dans le package `fr.uml.v.lovmi.filter`. De cette manière, si l'on souhaite créer un filtre, il suffit de déclarer une classe implémentant cette interface.

L'interface `Filter` est déclarée de la manière suivante:

```
public interface Filter {  
  
    public void start(Image in, int lineStart, int columnStart, int lineEnd, int columnEnd);  
  
    public void applyToPixel(int line, int column);  
  
    public void terminate();  
  
    public Image getResult();  
  
    public boolean produceResult();  
  
    public boolean canApply(Image image);  
  
    public boolean init() ;  
}
```

La classe implémentant l'interface `Filter` devra se trouver dans le package `fr.uml.v.lovmi.filter`.

Détaillons maintenant chaque méthode de l'interface, ainsi que leur rôle dans l'exécution du filtre.

La méthode `start` démarre le filtre. Par exemple, elle peut allouer l'image résultat aux bonnes dimensions, ou effectuer un calcul de paramètres.

La méthode `applyToPixel` sera appelée pour chaque pixel et devra appliquer le filtre pour ce pixel. Si le filtre ne permet pas d'appliquer le traitement pour un pixel indifféremment des autres, le parcours de l'image et l'application des traitements devront se faire dans la méthode `start`.

La méthode `terminate` sert à terminer le filtre (si des fichiers doivent être fermés par exemple).

`getResult` doit retourner l'image résultant de l'application du filtre.

`produceResult` indique si le filtre produit un résultat affichable ou non (si la méthode `getResult` renvoie une image non `null`).

La méthode `canApply` vérifie si le filtre peut être appliqué à une image (par exemple, le calcul de l'amplitude ne peut s'effectuer que sur une image comportant deux canaux). Elle sera bien sûr appelée avant exécution du filtre.

Enfin la méthode `init` initialise et démarre l'interface graphique, dans le cas où une boîte de dialogue est nécessaire pour que l'utilisateur fournisse des paramètres. Cette méthode doit retourner `true` si tout s'est bien passé et si les arguments sont corrects, dans le cas contraire elle doit retourner `false`.

Pour créer une boîte de dialogue, il faut utiliser une `WaitableFrame` dont la déclaration se trouve dans le package `fr.umlv.lovmi.frame`. Contrairement à la `JFrame` de Swing, la `WaitableFrame` force les autres fenêtres à attendre que son exécution complète se termine avant de continuer.

La méthode `waitUntilClosed` de la `WaitableFrame` oblige la fenêtre mère à attendre la fin de l'exécution de la `WaitableFrame`.

Pour débloquer la `JFrame` mère, il faut appeler la méthode `closed`. Elle permet de notifier sa fermeture.

Il est aussi possible d'entrer des paramètres via la ligne de commande, en utilisant la classe `java.util.Scanner` classique.

Une fois que la classe est implémentée et placée dans le bon package, il reste encore une étape pour que le filtre soit intégré au logiciel. Dans le package `fr.umlv.lovmi.filter`, se trouve également un fichier `filters.xml`. Le fichier a la forme suivante :

```
<filters>
  <filter name = «Bicubique» class= «fr.umlv.lovmi.filters.Bicubic» icon= «default.png» />
  <filter name = «Amplitude» class= «fr.umlv.lovmi.filters.Amplitude» icon= «amplitude.png» />
</filters>
```

On indique le nom du filtre, le chemin de la classe, et l'icône associée au filtre. Ce fichier est chargé au lancement du logiciel, et permet d'indiquer les filtres devant être chargés. Ainsi le nouveau filtre est ajouté au logiciel sans avoir à modifier le code de l'interface graphique, ou d'une autre partie du programme.