Subject: Re: [INFO0062-OOP] Projet
From: jean-francois.grailet@uliege.be
Date: 22/03/20 à 11:26
To: f straet <f.straet@student.uliege.be>

Bonjour,

Lorsque nous testerons votre projet, nous utiliserons exclusivement les classes et les méthodes telles que définies dans l'énoncé.

Cependant:
-vous pouvez créer une ou plusieurs classes qui vont stocker l'ensemble des informations des connexions existantes entre les filtres et conserver le(s) sample(s) retardé(s) des étapes précédentes pour gérer les boucles de feedback, de sorte à ne pas modifier les filtres de base que vous avez déjà créés.
-vous pouvez "encapsuler" un des filtres de base dans un autre objet qui va ajouter les variables dont vous avez besoin pour gérer l'état du filtre(par exemple, pour stocker le sample retardé de l'étape précédente, dans le cas d'un DelayFilter) et ses connexions avec d'autres filtres. Cela vous permet de continuer à utiliser la même interface que celle décrite dans l'énoncé pour CompositeFilter tout en étendant les propriétés de vos filtres (vous pouvez alors utiliser une HashMap pour faire correspondre un filtre avec l'objet qui l'encapsule). Avec un peu d'astuce, vous pourriez peut-être même le faire de façon générique, c.-à-d. indépendamment du fait que vous avez à l'intérieur de l'objet un DelayFilter, un GainFilter... ou même un autre CompositeFilter !
-éventuellement, vous pouvez ajouter aussi des méthodes aux classes DelayFilter / AdditionFilter / GainFilter (ou même leur faire implémenter une 2e interface que vous aurez créée) du moment que ça ne nous empêche pas, de notre côté, d'utiliser ces classes tel que prévu par l'énoncé. Entre autres choses, nous devons pouvoir compiler votre librairie avec le programme "CompositeExample.java" que j'ai publié hier sur ma page web:

http://www.run.montefiore.ulg.ac.be/~grailet/INFO0062_proj_19-20.php

(jetez aussi un oeil aux slides project_example.pdf pour avoir quelques commentaires sur ce code)

En espérant que ceci clarifie un peu nos attentes et les possibilités dont vous disposez.

Cordialement,

Jean-François

----- Mail original -----
De: "f straet" <f.straet@student.uliege.be>
À: "jean-francois grailet" <jean-francois.grailet@uliege.be>
Envoyé: Dimanche 22 Mars 2020 09:45:16
Objet: [INFO0062-OOP] Projet

Bonjour Monsieur,

Étant en train de travailler sur la classe CompositeFilter, je me retrouve à devoir gérer le feedback.

Dans les prototypes des méthodes à implémenter, les filtres sont toujours représentés par des objets implémentant l'interface Filter. Ainsi, je me demande si nous devons

n'utiliser que les méthodes de cette interface (afin de ne pas dépendre de notre propre implémentation), ou si nous pouvons définir d'autres méthodes pour les classes représentant les filtres basiques pour les utiliser dans les filtres composites.

En particulier, je voudrais pouvoir faire quelque chose comme :

```
if (filter instanceof DelayFilter) {
    // ...
    filter.myOwnMethod(/*...*/);
    // ...
}
```

Est-ce permis, ou bien devons nous gérer, par exemple, le cas où un utilisateur implémente un autre filtre délai et le connecte en boucle ?

Merci d'avance pour votre réponse,

François Straet