TD 13 – Premiers Principes de Conceptions Orientée Objet

Exercice 1 Une boîte à outils utile : BAKt

La société ***BAKorp*** développe de plus en plus de logiciels. Pour ne pas réinventer la roue à chaque nouveau projet, l'équipe technique décide de composer sa propre bibliothèque de développement. Ce projet, baptisé ***BAKt*** pour « ***BAKt is Another Kool Toolkit*** », se veut être un cadre de travail complet pour le développement d'applications variées.

Votre rôle dans ce développement est de vous consacrer à réaliser un gestionnaire d'événements. Typiquement, vous devez imaginer une architecture permettant à des applications développées à l'aide du framework de recevoir différents types d'événements. L'application aura la possibilité d'une part de générer des événements et d'autre part d'être notifiée lorsque ces événements ont été générés. Les deux parties de l'application (la partie de génération des événements et la partie de réception des événements) devraient être faiblement couplées de façon à garder le maximum de souplesse dans le développement.

Dans un premier temps, afin de valider la structure, il vous est demandé de gérer des événements clavier (touche enfoncée et touche relâchée), des événements souris (déplacement du pointeur, bouton enfoncé et bouton relâché) et un événement de type temporisation généré à intervalle régulier. Ne sachant pas trop par où commencer, vous êtes aidé par votre chef de projet qui vous propose de réaliser une application de démonstration du gestionnaire d'événement basée sur les trois classes représentées dans la figure 1.

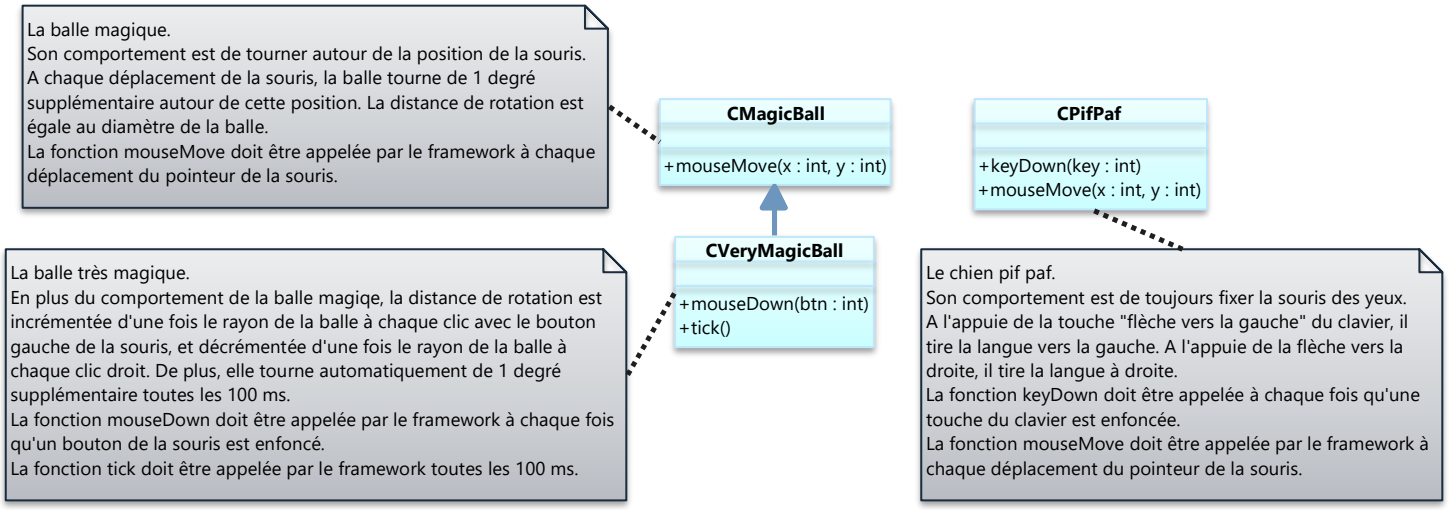

Figure 1 : Classes principales de l'application de démonstration du gestionnaire d'événements

- Tentez de proposer une architecture permettant de résoudre tant la partie de génération des événements que la partie réception.
  - Votre architecture respecte-t-elle les principes de conception précédemment présentés ?

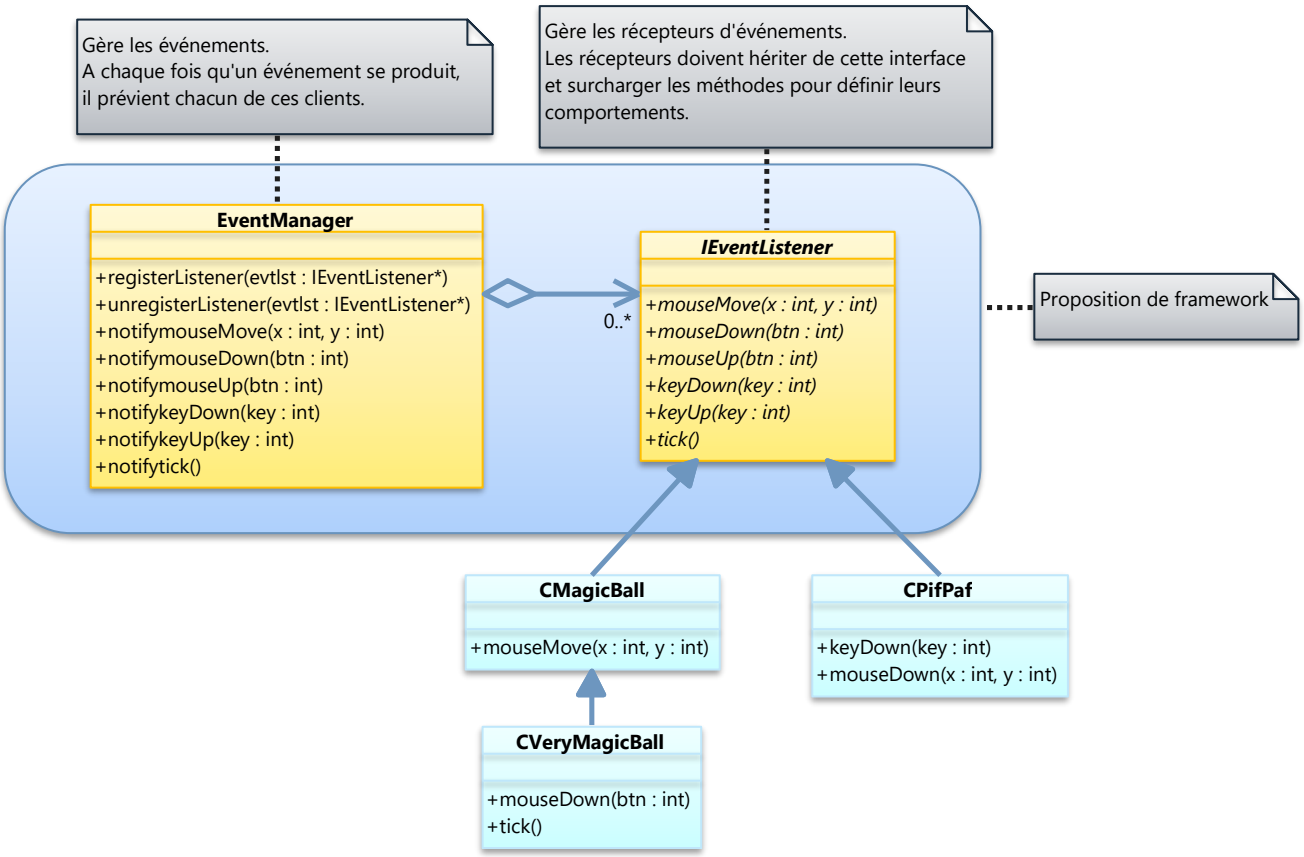

Figure 2 : Mauvaise solution

La solution de la figure 2 est mauvaise car :

1. elle viole manifestement le principe de responsabilité unique car les deux classes du framework s'occupent de tous les types d'événements. Il serait souhaitable de les couper en trois pour les trois types d'événements (clavier, souris, timer)
2. elle viole le principe ouvert-fermé car ajouter de nouveaux événements passera par la modification de ces deux classes
3. Les classes **`CPifPaf`** et **`CMagicBall`** se voient munie par l'héritage de la fonctionnalité du timer, alors qu'elles n'en ont pas besoin.
4. Les classes **`CMagicBall`** et **`CVeryMagicBall`** se voient munie par l'héritage de la fonctionnalité des événements clavier alors qu'elles n'en ont pas besoin.