Design Pattern – Corrigé du TD n°4

LE PATTERN SINGLETON : Y A-T-IL UN CONTRÔLEUR DANS LA TOUR ?

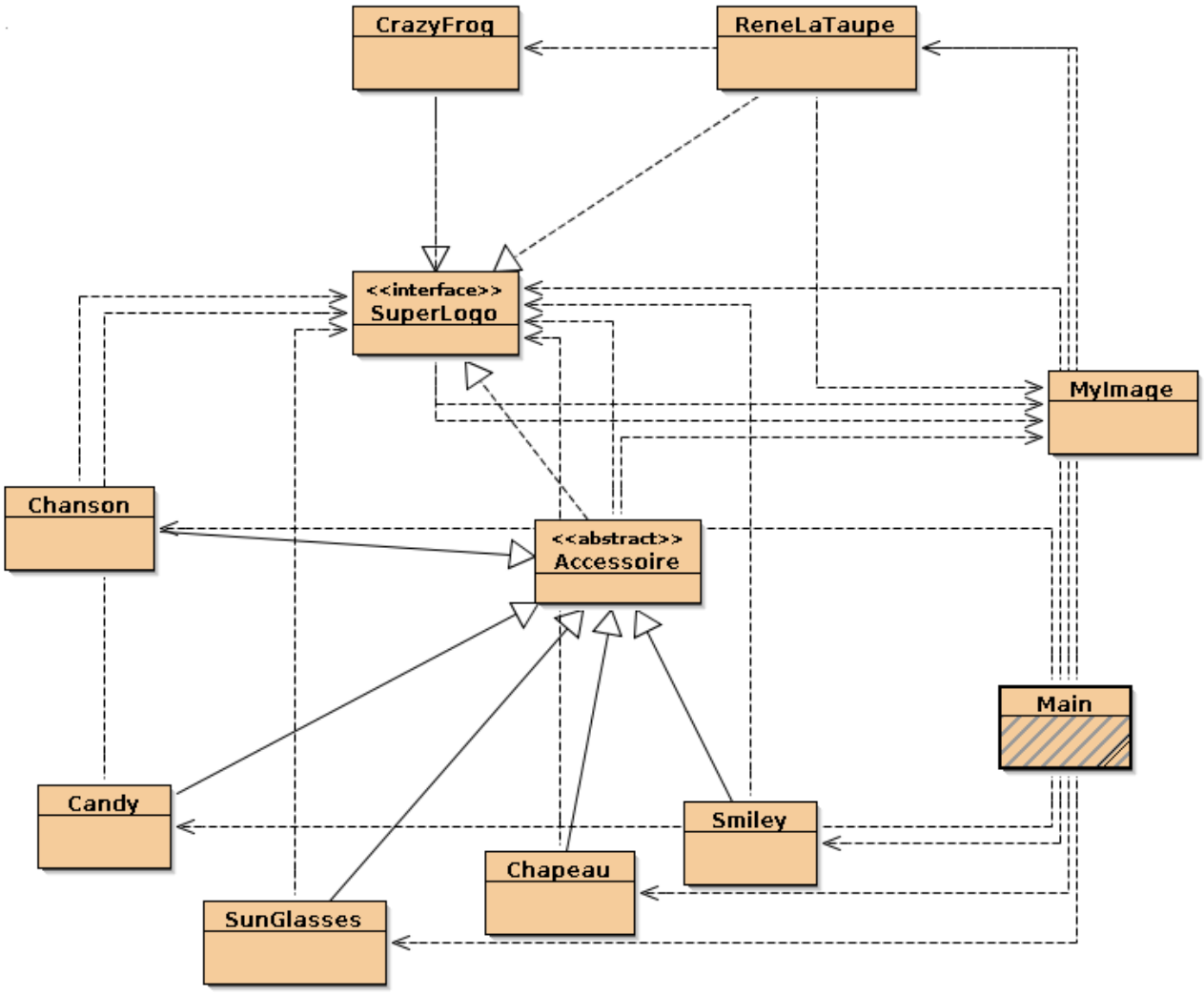
Le pattern Singleton n'est pas compliqué en lui-même : rendre un constructeur privé afin de contrôler le nombre d'instances d'un objet que l'on peut réaliser. La difficulté ici consiste à rendre le programme « thread safe » : si plusieurs threads (programmes tournant en parallèle) demandent à construire un objet Singleton en même temps, un seul objet doit être construit en tout.

> Téléchargez la code source de la solution ici <

LE PATTERN DÉCORATEUR : RENÉ LA TAUPE

Le pattern décorateur permet de superposer des comportements (ici, l'affichage d'objets sur une image de départ, et le calcul de prix par addition successive des prix des composants de l'image) en utilisant la technique de l'encapsulation, qui consiste à placer une instance de la classe X dans la classe X elle-même.

Voici le schéma de classe que vous devriez obtenir à la fin



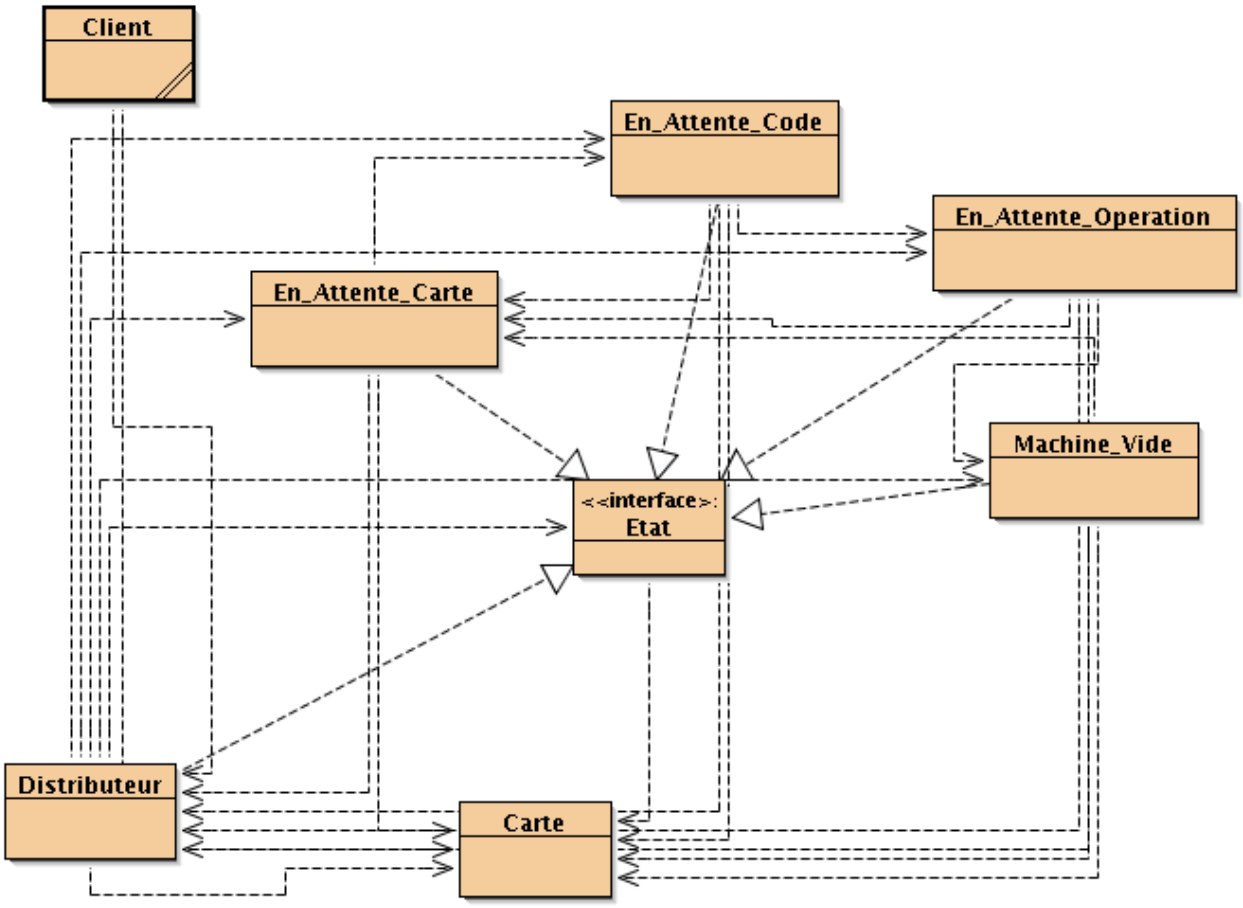
> Téléchargez la code source de la solution ici <

LE PATTERN STATE

Le pattern State permet de décrire les différents états et les transitions d'état d'une machine d'état. Le but n'est pas de gérer en soi les transitions, mais de gérer correctement la machine en elle-même: si une transition est demandée dans un état qui ne lui correspond pas, la machine d'état ne doit pas « planter ». Comme dit précédemment, le but de ce pattern n'est pas de générer les demandes de transitions d'état (ces demandes seront effectuées par l'utilisateur de la machine, à travers un menu et des boutons), mais de gérer correctement la machine afin qu'elle réponde correctement à toutes les demandes que l'on peut effectuer.

Il est important, pour chaque état, de bien coder chacune des fonctions de transition, même si ces dernières ne font pas transiter l'état (par exemple, en mode *En_Attente_Carte*, le fait de *retirer_carte* ne fait rien).

Une fois terminée, votre machine devra ressembler à ceci



> Téléchargez la code source de la solution ici <

On peut proposer une solution plus robuste, où l'on évite de faire des new pour chaque changement d'état, et où un utilisateur extérieur ne pourrait pas changer l'état de notre machine (pas de setter d'état).

> Téléchargez la code source d'une meilleure solution ici <