Cours d'introduction à la Programmation Orientée Objet

1 Contenu du TP

- Le but de ce TP est de vous permettre d'entrer de façon très simple et guidée dans le développement Java, au travers d'une application simple démontrant les principaux mécanismes objets, et la grammaire des classes Java les implémentant.

  A l'issue de ce TP, vous détiendrez un exemple sur lequel vous baser pour mettre en œuvre vos propres classes, et ainsi démarrer vos premiers programmes.

- Le code source de ce projet (à importer dans eclipse) est structuré autour de trois packages :

  - **model** : Il présente l'implémentation d'un « *set d'entités*».

    Ces composants homogènes, au sein d'une problématique métier, permettent de prendre en charge fonctionnellement la vente simple de produits - rattachés à des comptes clients (cas usuel et nominal dans le monde industriel).

    On y trouve les *classes* standards *Product, Dvd, Game, Category, Order, OrderLine*, *Client, Address*, ainsi que l'*Interface IBonusProducer*.

  - **utils** : un package dédié à stocker les « Helpers » classes (Utilitaires, orientée *static*) :

    Ici très simple, seule une classe *DateHelper* sera en charge de formater les Dates de nos commandes (à l'intérieur de nos traces consoles).

  - **test** : contenant une classe unique Main (et son point d'entrée *main*) :

    Permet de lancer l'application, et d'observer sur la sortie standard les messages du déroulement de la séquence d'appels de l'application.

- Le modèle met en place :

  - une arborescence objet simple (autour des classes *Product, Dvd, Game*), montrant les mécanismes *d'héritage* et de *polymorphisme* entre la classe mère, et ses classes filles.

    En outre on y trouve un exemple *d'overload*, *(surcharge)*, dans les constructeurs publics de ces classes.

  - Les principes *d'encapsulation objet*, de par la mise à disposition de *getters/setters* (ou encore appelé *accessors* et *mutators* en terminologie Java) pour les *champs* composant sa structure de classe.

    Ces éléments suivent la norme des *JavaBeans*, pilier fondateur de la norme de développement du langage Java.

  - Un « comportement » métier sous la forme de l'interface *IBonusProducer*, qui est l'approche « orientée objet » en Java (sans héritage multiple) :

    Celle de contrats d'interfaces définies à implémenter, permettant d'enrichir la définition fonctionnelle des classes.

    Les classes étendent en cela leur structuration au travers des *Types* (*cast/transtypage*) employables, et l'implémentation s'oriente ainsi vers un meilleur niveau d'abstraction !

  - Une arborescence *Composite* d'objets, référencés entre eux, qui proposent un *override*, une *redéfinition*, de la méthode *toString* (renvoyant la définition d'une instance dans la *Machine Virtuelle*), de la Classe racine *java.lang.Object.*

    De plus ici, on peut voir un exemple de chainage classique de la méthode *toString*, exploitée dans chacune des classes concernées.

- La classe Main fournit un point d'entrée pour l'application, on y retrouve :

  - l'instanciation des différents exemples d'objets relatifs aux classes précédemment présentées.

  - Un appel vers la classe *Singleton java.lang.System* (*Design Pattern* de la famille « *création* » : qui contrôle sa population d'objets produits, et n'est pas instanciable directement).

    Entité unique dans le contexte de la *Machine Virtuelle*, afin qu'elle nous décrivent, via la méthode *toString* la composition de nos objets.