STEFANOS

Alexandre

21905831

Génie Logiciel - TD11

Partie 1 - Patron de Conception

2) J’ai tiré le patron Composite.

Composite est un patron de conception structurel qui permet d’organiser les objets dans des arborescences pour pouvoir les traiter comme des objets individuels.

Le problème est le suivant : l’utilisation doit être réservé aux applications dont la structure principale peut être représentée sous la forme d’une arbrorescence.

Pour résoudre ce problème, Composite propose de manipuler les feuilles de l’arborescences à l’aide d’une interface qui déclare une méthode de calcul du prix total.

3) En Java :

public class Tableau {

private int nb\_rangee, nb\_colonne ;

public void afficherTableau() {

//afficher Tableau

}

}

public class Rangee {

private int nb\_rangee;

public Rangee(int nb\_rangee) {

this.nb\_rangee = nb\_rangee;

}

public void ajoutRangee(int ajout\_nb\_rangee) {

this.nb\_rangee += ajout\_nb\_rangee;

}

public void ajoutColonne(int ajout\_nb\_colonne) {

this.nb\_colonne += ajout\_nb\_colonne;

}

}

public class Colonne extends Tableau {

private int nb\_colonne;

public Colonne(int nb\_colonne) {

this.nb\_colonne = nb\_colonne;

}

public void ajoutColonne(int ajout\_nb\_colonne) {

this.nb\_colonne += ajout\_nb\_colonne;

}

}

public class GestionTableau extends Tableau {

public void initialisation() {

//initialisation des tableaux : colonnes et rangées

}

public void groupeSelectionne(Tableau tableau) {

/\*Permettant de gérer chaque rangée et chaque tableau dans une seule méthode comme si c’était un tout\*/

for(Rangee rangee : tableau) {

//gestion du tableau rangee par rangee

}

for(Colonne colonne : tableau) {

//gestion du tableau colonne par colonne

}

afficherTableau() ;

}

}

Partie 2 - Patrons de conception

Avec El-Mougharti Hanafi et Cointe Constant

Nous avons séparément étudié les patrons suivant :

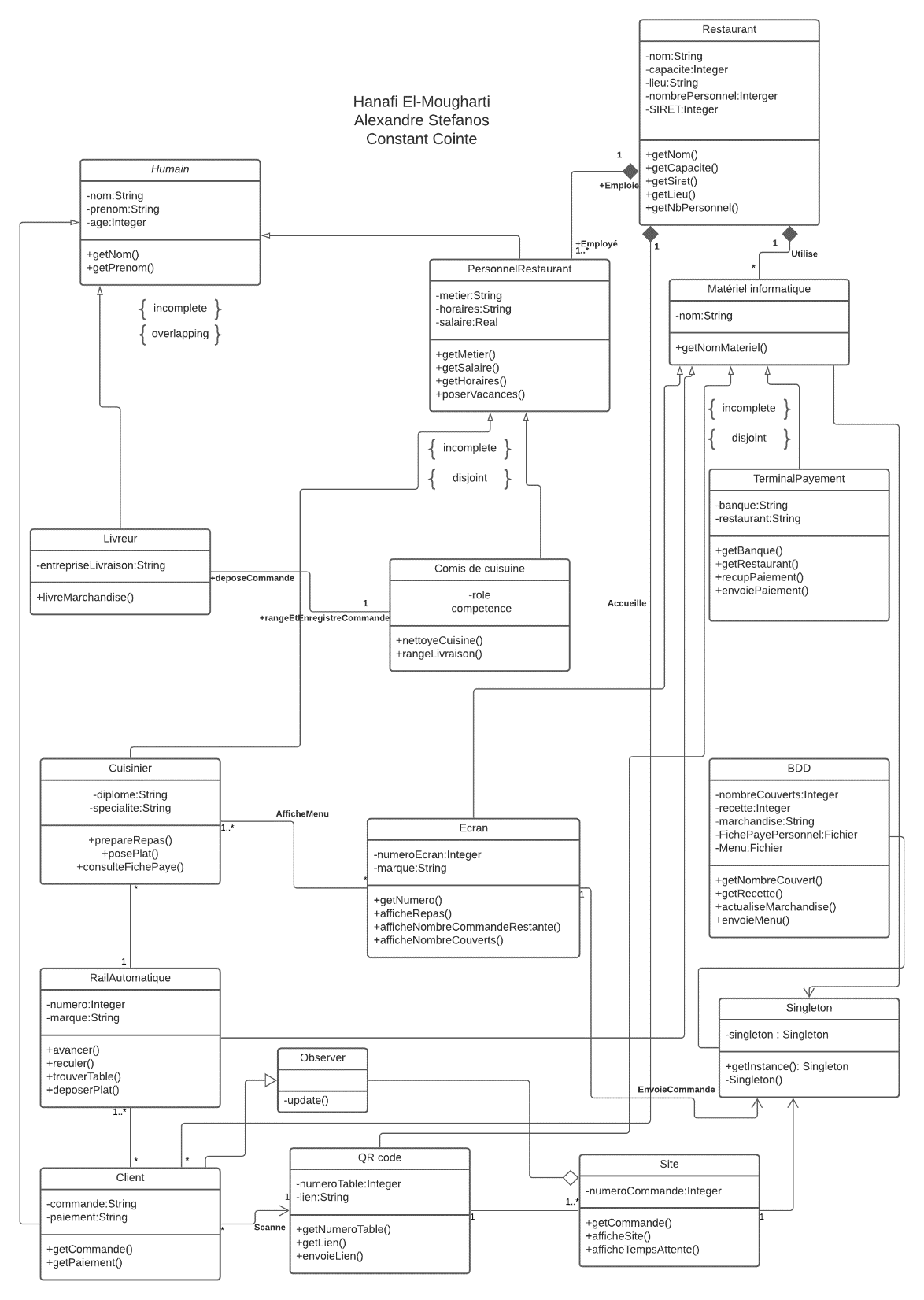
* Alexandre : Composite
* Constant : Stratégie
* Hanafi : Création

Le premier patron de conception que l’on a identifié dans notre projet Fil Rouge est l’Observateur. Ce patron comportemental permet au client soumettant sa commande d’être informé de l’arrivée de cette dernière.

Nous avons ensuite identifié le patron de conception comportemental suivant : la Commande. Ce patron permet de mettre en place une liste ou une file d’attente. Dans notre cas, on l’utilise pour indiquer à un client le temps de commandes, qui peut être gérer en planifiant les différentes exécutions de la liste d’attente, annuler une opération, etc.

Le troisième patron que nous avons identifié est la Stratégie. Ce patron comportemental permet de définir plusieurs algorithmes à appliquer en fonction du contexte. Dans notre cas, en fonction de la commande du client : par exemple, si c’est un menu à entrée + plat, un menu plat + dessert ou un menu entrée + plat + dessert.

Enfin, le dernier patron de conception que nous avons identifié est le Singleton. Ce patron comportemental permet de préciser que nous avons qu’une seule instance d’une classe. Effectivement, nous n’avons qu’une seule base de donnée pour notre restaurant.

Voici notre diagramme de classe mis à jour :