Laboratoire de Java (I)

2ème informatique de gestion et 2ème informatique et systèmes (industrielle et réseaux) 2017-2018

Claude Vilvens – Christophe Charlet – Jean-Marc Wagner

Projet "Le gourmet audacieux" - 1ère partie

Contexte de développement

1. Préambule

L'Unité d'Enseignement " **Développement orienté objets et multitâche** " (gestion: 9 ECTS (105 h) / indus & réseaux: 8 ECTS (94 h)) se structure en trois Activités d'apprentissage de la manière suivante :

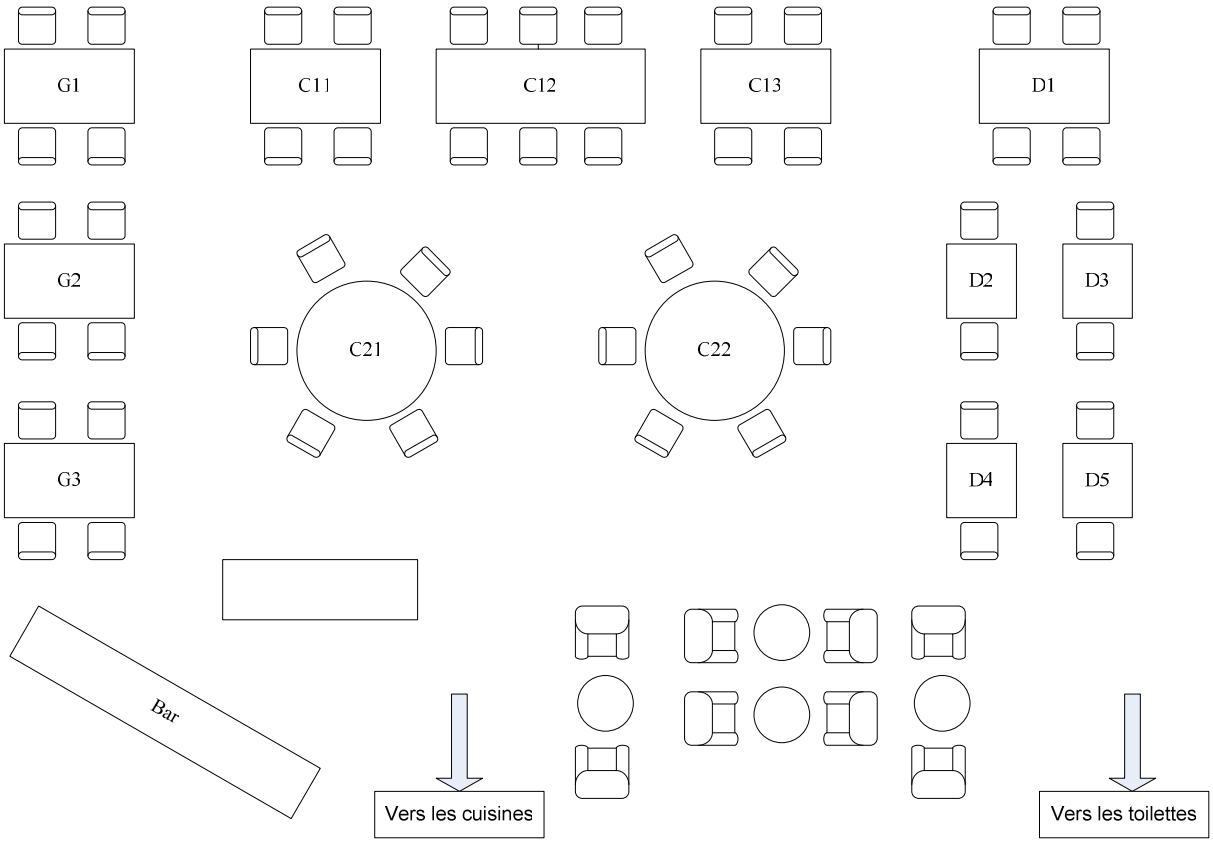
- AA: Programmation du multi tâche léger- Threads (gestion: 30h (29%) / indus & réseaux: 19h (20%))
- AA: Programmation orientée objet Unix et Windows - Java (gestion: 30h (29%) / indus & réseaux: 30h (32%))
- AA: Programmation orientée objet Windows- C# (gestion: 45h (42%) / indus & réseaux: 45h (48%))

Les travaux de programmation réseaux présentés ici constituent la description technique des travaux de laboratoire de l'AA " **Programmation orientée objet Unix et Windows - Java** ".

2. Le projet "Le gourmet audacieux"

Le thème de ce laboratoire de programmation Java est la gestion élémentaire d'un **restaurant** dénommé **"Le gourmet audacieux"**. Ce restaurant propose à une clientèle d'affaires ou familiale des plats qui sortent des sentiers battus, mariant des saveurs de façon parfois … audacieuse (voir la carte dans les copies écran ci-dessous). Vu la consistance des plats proposés, le restaurant ne propose pas d'entrées. Par contre, il propose des desserts … tout aussi "audacieux". La salle du restaurant se veut un endroit convivial :

et comporte des tables réparties selon le schéma suivant :



Le projet **"Le gourmet audacieux"** à développer est destiné principalement aux serveurs qui travaillent en salle (application **ApplicationSalle)**, mais une interaction est prévue avec le personnel qui travaille en cuisine (application **ApplicationCuisine**). Les boissons servies sont gérées par le bar (application **ApplicationBar**), qui travaille de manière autonome et se contente d'envoyer des additions de boissons à l'application principale.

L'idée générale est que l'application **ApplicationSalle** propose un interface permettant aux serveurs (serveuses) d'encoder les plats et desserts commandés et d'être prévenu(e)s de leur disponibilité quand ils sont prêts à être servis.

Il va de soi que les possibilités d'une telle application réellement utilisée dans un restaurant sont plus nombreuses. Dans le contexte de ce laboratoire, seules certaines fonctionnalités auront le temps d'être d'implémentées.

Le développement de ces applications fera intervenir :

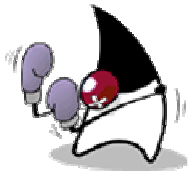
- la conception de classes et interfaces répartis dans des packages structurés;
- les bases de la programmation Java (notamment les packages, interfaces et classes abstraites, exceptions);
- les interfaces graphiques principalement Swing (notamment les JComboBox, JList et JTable);
- les classes utilitaires (containers comme Vector, LinkedList et Hashtable, StringTokenizer, Date-Calendar-DateFormat-Timezone);
- l'utilisation de la ligne de commande élémentaire (javac, java, jar);
- les flux et plus particulièrement les techniques de sérialisation sur fichier, l'utilisation de fichiers Properties et la lecture/écriture de fichiers textes;
- l'utilisation d'une librairie de communication réseau élémentaire (simple échange de chaînes de caractères) disponible sous forme d'un jar;
- les Java Beans avec les chaînes d'événements propriétaires et de type "propriété liée";
- la portabilité Windows-Unix (exécution d'une application Java à distance avec un serveur X-Window);
- une première approche des threads.

3. Les objectifs et les outils

L'ensemble des travaux proposés ici est à réaliser en utilisant l'environnement de développement **NetBeans version 8.***, basé sur le **JDK 1.8**. Comme éditeur de texte plat, on pourra utiliser **JEdit 4.2** tandis **Xming 6.9** sera le serveur X-Window privilégié.

Le dossier est, *à priori*, à réaliser **par équipe de deux étudiants** mais **peut aussi être présenté par un étudiant seul** qui le désirerait, avec des avantages et des inconvénients qui s'échangent selon le choix posé.

Un petit conseil : ***lisez bien l'ensemble de l'énoncé*** avant de concevoir (d'abord) ou de programmer (après) une seule ligne ;-). Ceci vous permettra non seulement d'avoir une vision globale du projet mais aussi de déjà remarquer des traitements communs à des points différents des applications. Prévoyez une schématisation des diverses classes (peut-être des diagrammes de classes UML ?) et élaborez d'abord "sur papier" (donc sans programmer directement) les divers scénarios correspondant aux fonctionnalités demandées.



4. Les règles d'évaluation

1) L'évaluation établissant la note de l'AA de " Programmation orientée objet Unix et Windows - Java" [ex "Réseaux et technologie Internet"] est réalisée de la manière suivante :

- théorie : un examen écrit en juin 2018, sur base des notes de cours ("Java I: programmation de base") et de listes de points de théorie à préparer, listes fournies au fur et à mesure; il sera coté sur 20;

♦ laboratoire : 2 évaluations (une fin avril, non remédiable, et une durant la 1ère session, remédiable), chacune cotée sur 20; la **moyenne pondérée de ces 2 cotes** (poids respectifs de **4/10** et **6/10**) fournit une note de laboratoire sur 20;
♦ note finale : **moyenne de la note de théorie (poids de 5/10) et de la note de laboratoire (poids de 5/10)**.

Cette procédure est d'application tant en 1ère qu'en 2ème session.

2) *Dans le cas où les travaux sont présentés par une équipe de deux étudiants*, chacun d'entre eux doit être capable d'expliquer et de justifier l'intégralité du travail (pas seulement les parties du travail sur lesquelles il aurait plus particulièrement travaillé).

3) En 2ème session, un **report de note** est possible séparément pour **la note de laboratoire** ainsi que pour **la note de théorie pour des notes supérieures ou égales à 10/20**.
Toutes les évaluations (théorie ou laboratoire) ayant des **notes inférieures à 10/20** sont **à représenter dans leur intégralité**.

La première partie des travaux de programmation réseaux sera **évaluée** par l'un des professeurs du laboratoire **à partir de la semaine du 30 avril 2018** (avec rentrée d'un dossier papier - le délai est à *respecter* impérativement).

La deuxième partie sera **évaluée** lors de l'examen de laboratoire en juin 2018 (le dossier papier n'est plus nécessaire).

Evaluation 1

Les applications demandées doivent fonctionner sur une machine **Windows (PC** ou **portable) et** sur une machine **UNIX (Sunray** avec terminal ou émulation de terminal**)** de l'InPrES.

Dossier :
- ♦ schéma UML des classes utilisées dans l'application
- ♦ explication et code correspondant à un changement de table dans l'application.

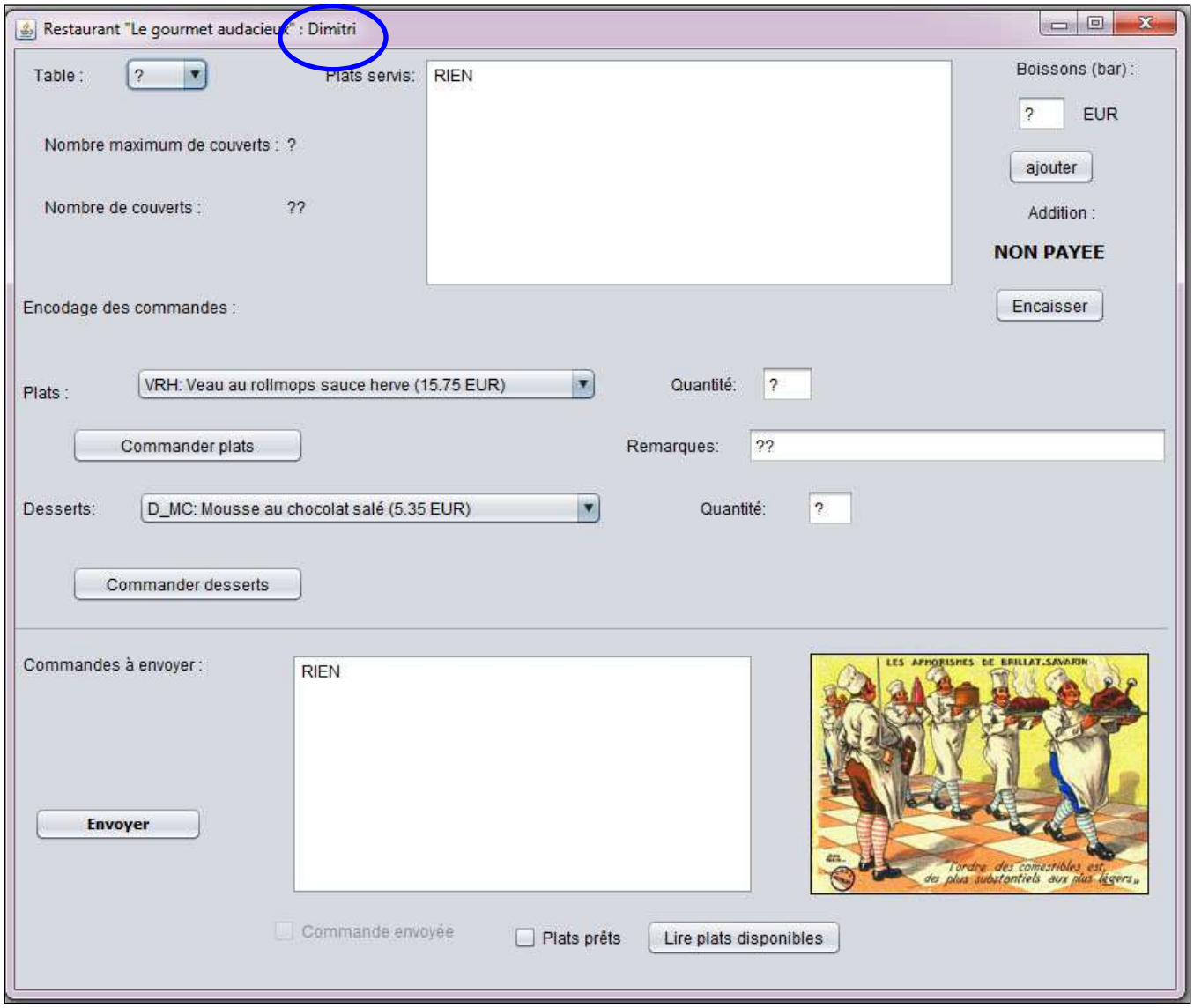
I. Fonctionnalités de base du projet "Le gourmet audacieux"

1. L'application ApplicationSalle

Lorsqu'un serveur a pris la commande d'une table, il entre dans l'application et se fait reconnaître comme utilisateur agréé (procédure classique de login-password) :

Pour l'instant, la validation d'un serveur se base sur une hashtable statique en mémoire (clé=login=prénom, valeur=mot de passe).

En cas de succès, il parvient à l'interface graphique qui se présente, dans un premier temps, sous l'aspect suivant (modifiable selon les souhaits des commanditaires):



Le rôle de cet interface graphique est clairement d'enregistrer une commande de plats et desserts pour une table donnée et d'envoyer cette commande à la cuisine (qui utilise l'application ApplicationCuisine).

2. Au préalable : les classes de base

Une modélisation minimale de l'application considérée implique l'utilisation de classes "Serveur", "Plat" et "Table". Plus précisément :

1) la classe **Serveur** se contente d'encapsuler les nom, prénom et login (en principe son prénom, éventuellement complété des premières lettres du nom si risque d'ambigüité) du serveur ainsi que son numéro de carte d'identité;

2) la classe **Plat** est une classe abstraite qui comporte l'information du prix; elle implémente (partiellement donc) l'interface **Service** qui déclare les méthodes :

```
double getPrix();
String getLibelle();
String getCategorie();
```

En ce qui concerne la catégorie, la classe **CategoriePlat** encapsule un nom de catégorie (un String) et, pour faciliter son emploi, contient des variables membres static instances de cette classe :

static final CategoriePlat PLAT_PRINCIPAL, DESSERT, BOISSON, ALCOOLS, etc.

Les classes **PlatPrincipal** (catégorie CategoriePlat.PLAT_PRINCIPAL) et **Dessert** (catégorie CategoriePlat.DESSERT) implémentent bien sûr la classe Plat. L'une et l'autre contiennent aussi un code (ex: "FE" pour "Filet de boeuf Enfer des papilles", "D_DG" pour le desert "Dame grise").

Dans une première approche (mais cela changera dans la suite), les plats sont référencés dans une hashtable statique (clé=code, valeur=objet PlatPrincipal qui contient le nom et le prix). On pratiquera de même pour les desserts.

3) la classe **Table** est évidemment la pierre d'angle du restaurant. Elle comporte

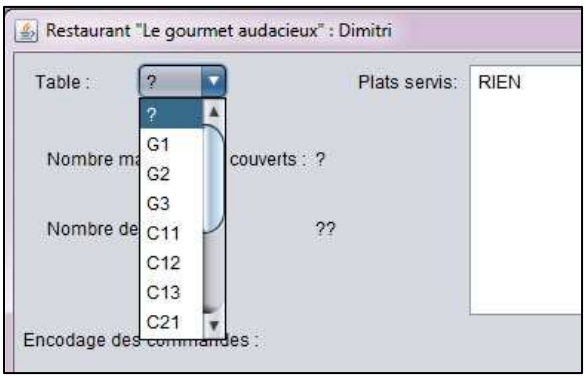
- un numéro de table;
- un container (Vector, ArrayList, etc) des commandes de plats qui y sont servis;
- le nombre maximum de couverts et le nombre effectif de couverts (il peut n'y avoir que trois convives sur une table de 4 places);
- l'addition;
- le fait que l'addition a été payée (synonyme de départ des clients) ou pas;
- le prénom du serveur qui a pris la table en charge (nous supposerons ici que ce serveur ne change pas durant une séance).

La classe **CommandePlat** permet de référencer un Plat (plat principal ou dessert) avec la quantité commandée. Il est ainsi plus facile de mémoriser les commandes d'une table (sinon, il faudrait mentionner chaque plat le nombre de fois qu'il a été commandé :-(…).

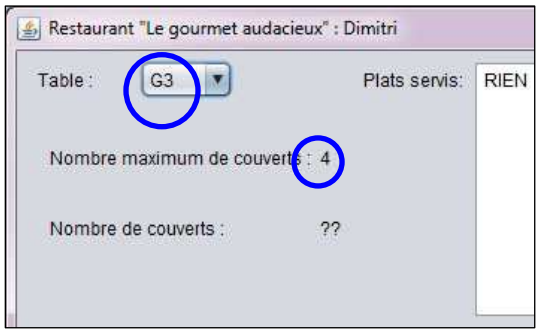
Si les clients changent, les tables ne changent pas en ce qui concerne leur numéro et nombre maximum de couverts. Dans une première approche (mais cela changera dans la suite), les tables sont référencées dans une hashtable statique (clé=numéro de table, valeur=objet Table qui contient donc le nombre maximum de couverts).

3. La prise de commande

Le serveur commence bien sûr par choisir la table :



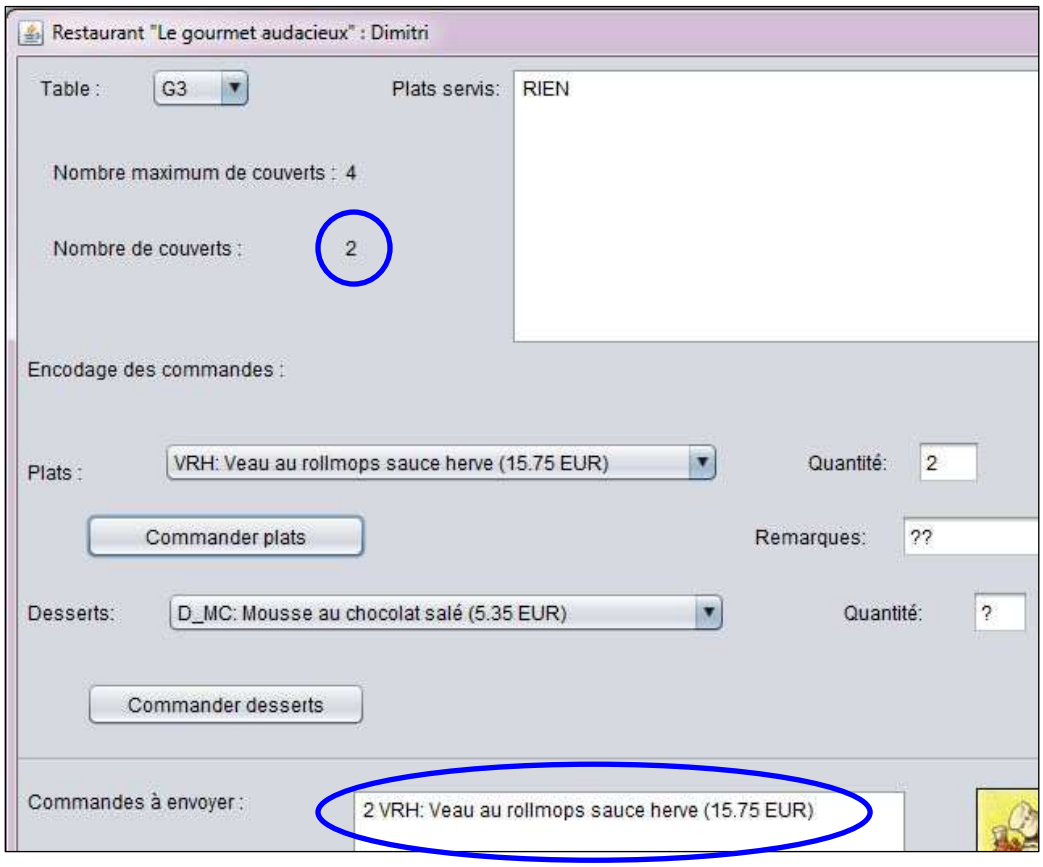
Une fois ce choix effectué, le nombre de couverts possible pour cette table est affiché:



Le serveur peut alors encoder les plats et desserts choisis par cette table ainsi que le nombre de chacun d'entre eux :

puis

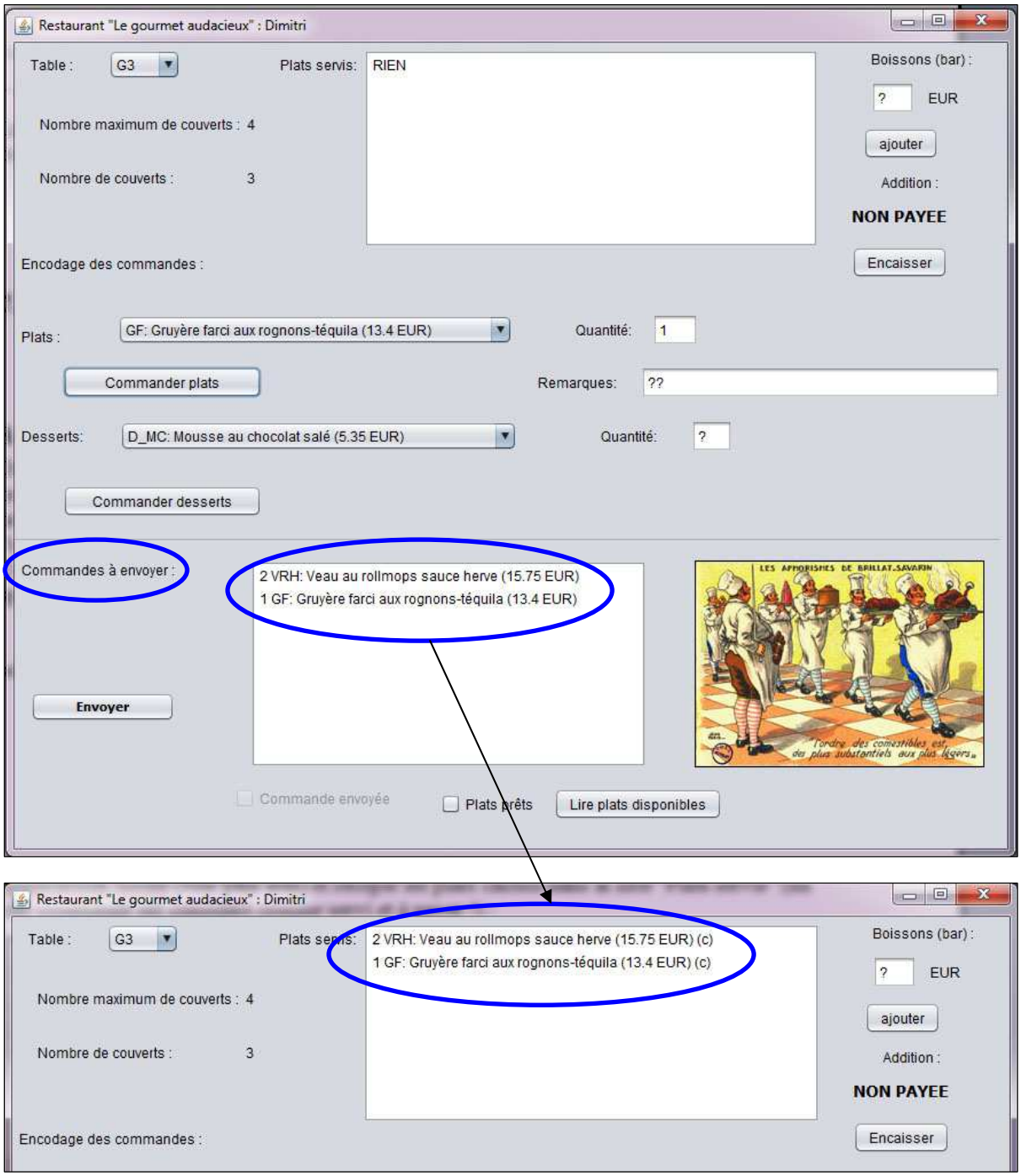
L'appui sur le bouton "Commander plats" a pour effet d'envoyer la commande dans la liste "Commandes à envoyer" et aussi d'incrémenter le nombre effectif de couverts :



Bien sur, ceci a aussi pour effet d'ajouter les plats dans le container de plats de la table. Si le nombre effectif de couverts dépasse le nombre maximum de couverts de la table, l'exception **TooManyCoversException** est lancée. Elle ne signifie pas forcément une erreur (on a pu ajouter un couvert en plus si les clients se serrent un peu ;-)) : ce sera au serveur de choisir dans la boîte de dialogue qui apparaît dans ce cas si on continue ou si on annule tout (donc erreur véritable).

La procédure est similaire pour les desserts, mais sans l'incrémentation et la vérification du nombre effectifs de couverts (certains convives sont des amateurs de desserts ;-)).

Une fois toutes les commandes encodées, on peut envoyer la commande globale des plats par appui sur le bouton "Envoyer" qui bien sûr envoie cette commande à l'application ApplicationCuisine (voir plus loin) et recopie les plats choisis dans la liste "Plats servis" (un plat commandé est considéré comme servi et à payer !) :



L'addition de la table est évidemment incrémentée des prix des différents plats et desserts.

II. Fonctionnalités de changement de table et d'addition

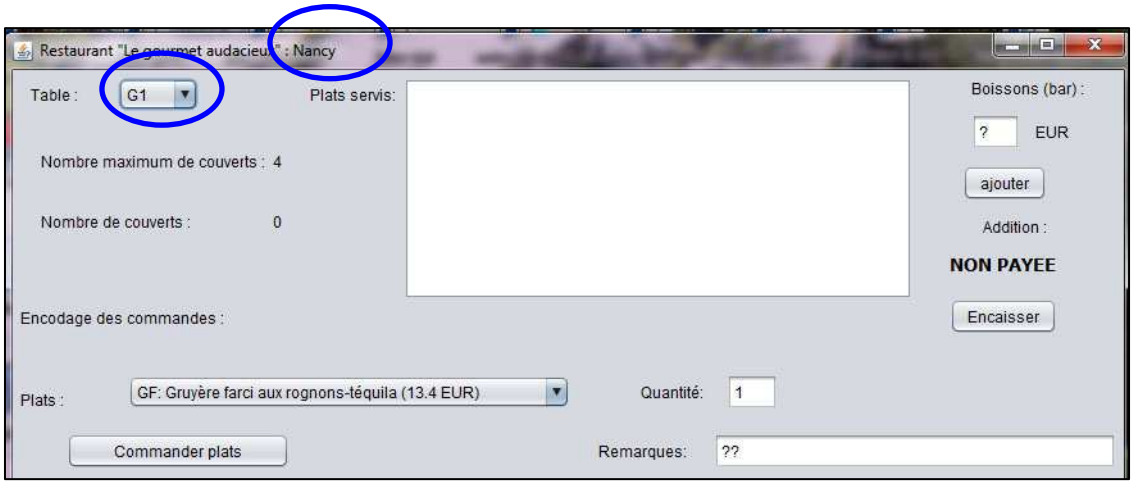
4. Changement de table et de serveur

Jusqu'à présent, tout s'est passé sur une seule table avec intervention du serveur qui s'en occupe. A remarquer : on supposera qu'un serveur qui encode ses commandes le fera en terminant par l'envoi de la commande aux cuisines (cela semble raisonnable - sinon, quel intérêt ?).

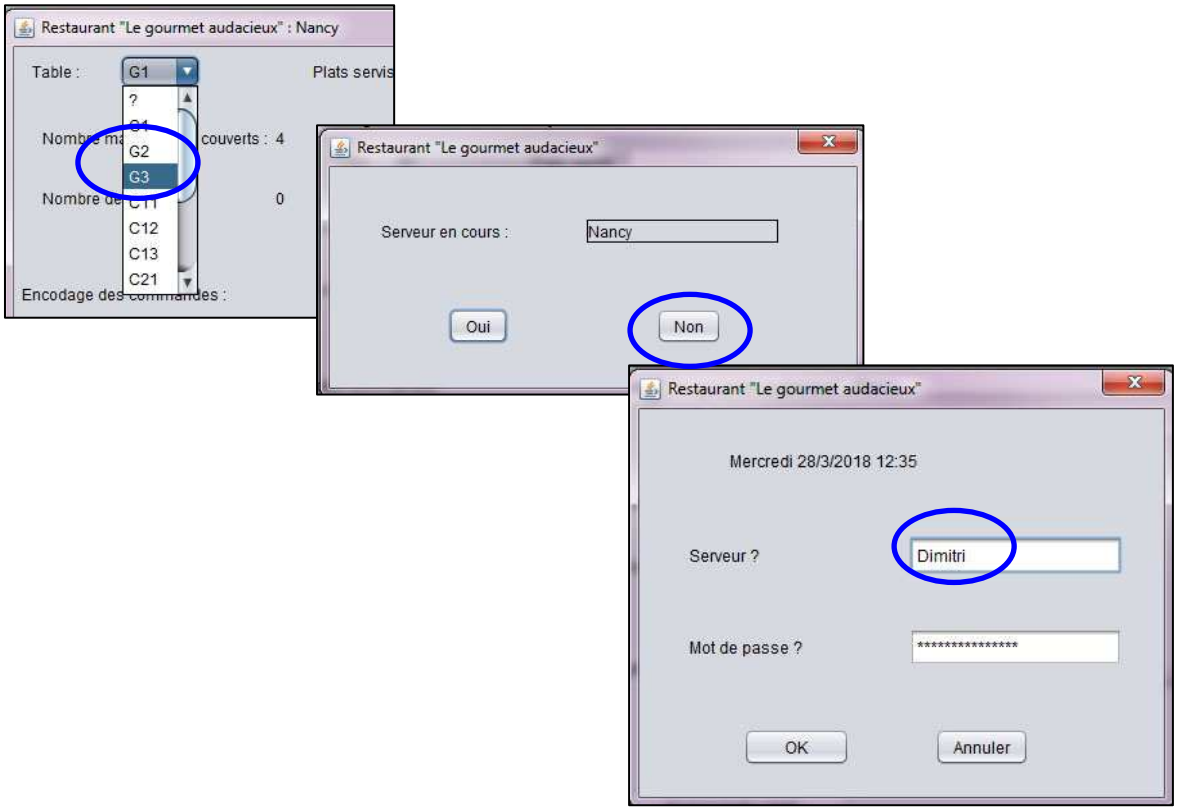
Mais bien sûr, un autre serveur peut vouloir intervenir sur une autre table dont il est (ou va devenir) le responsable de service. Pour ce faire, une sollicitation sur la boîte combo des numéros de table fera apparaître une boîte de dialogue demandant si le serveur change :

Si non, le nouveau serveur (ou la nouvelle serveuse) se fait reconnaître :

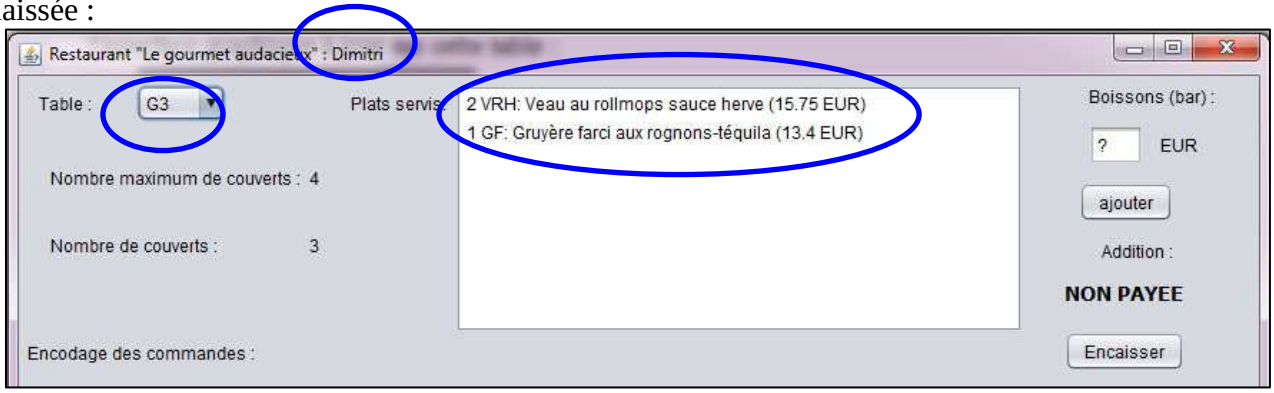
Après quoi ce serveur se voit (ré)attribuer la table et l'interface graphique est simplement mis à jour avec les informations de la table choisie (si il en existe) :



Bien sûr, un retour à une table pour laquelle on a déjà enregistré les commandes mettra l'interface graphique à jour sur cette table :

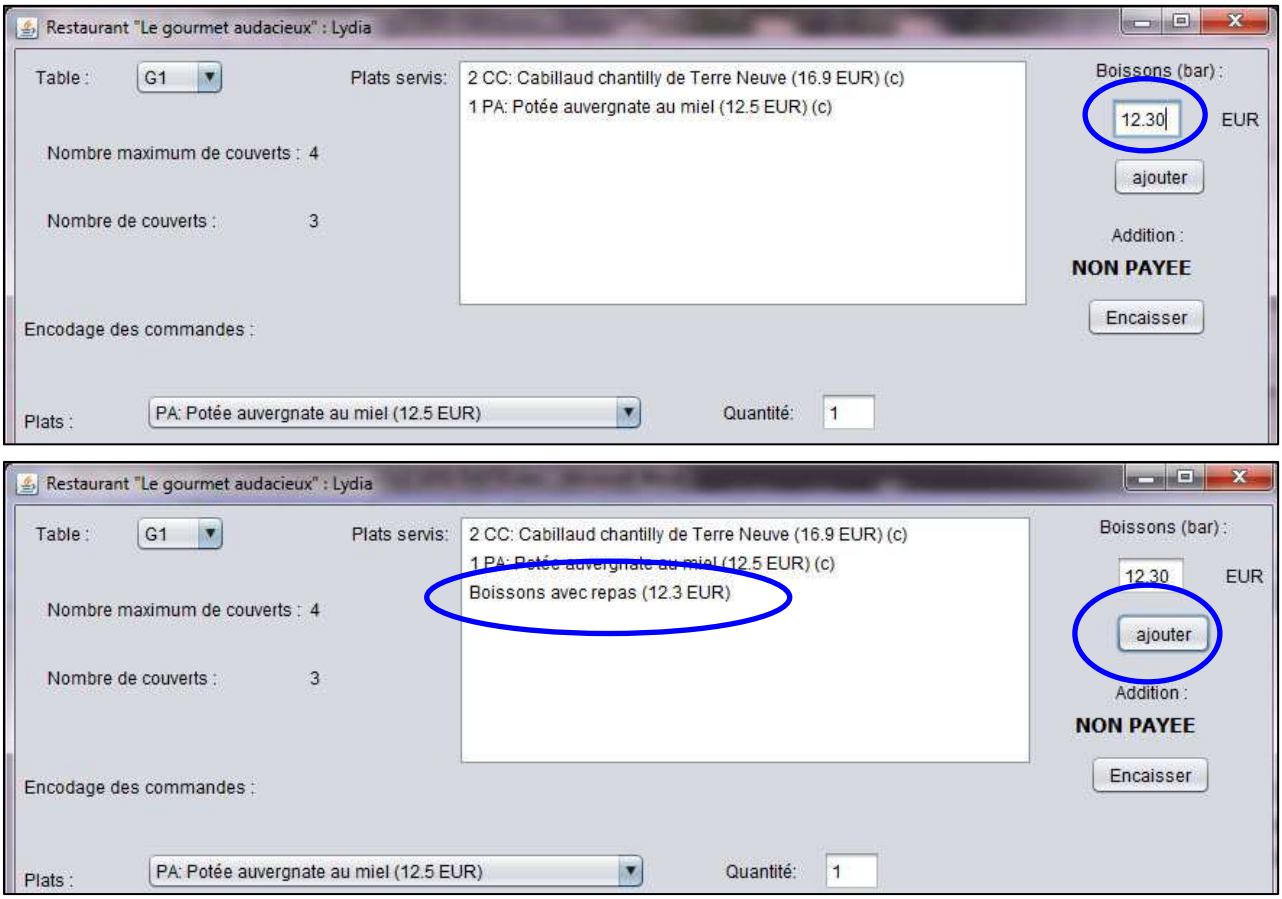


Ici, Dimitri a repris la main pour revenir sur la table G3 qu'il retrouve dans l'état où il l'a laissée :

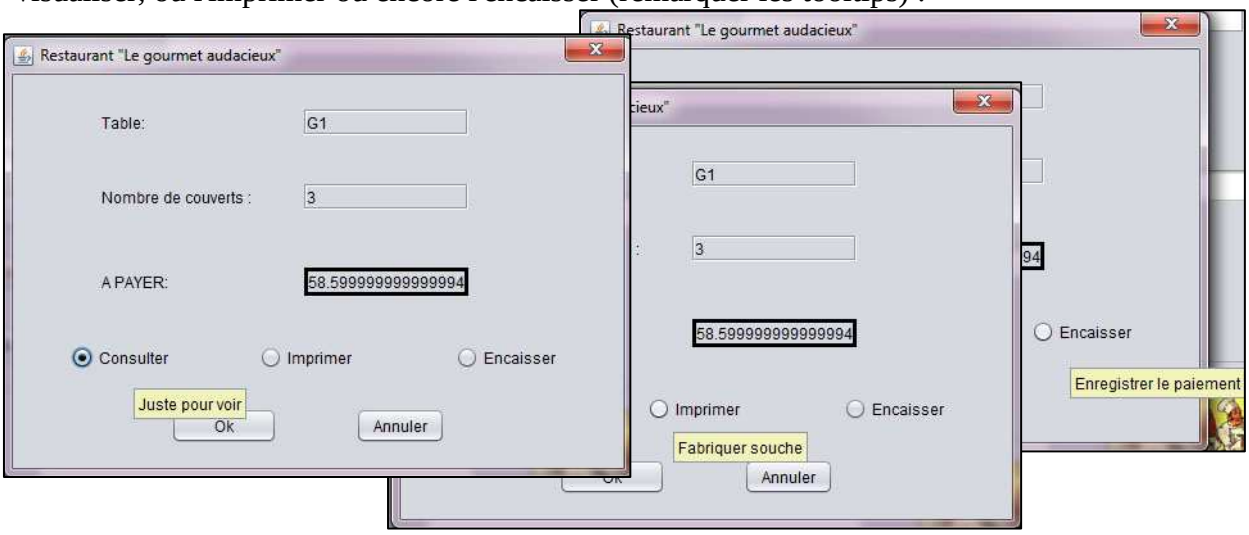


5. Les boissons et l'addition

Le bar fonctionne de manière autonome (il est possible de simplement passer boire un verre - voir plan du restaurant avec les fauteuils près du bar). Le serveur (la serveuse) responsable d'une table se contente donc d'enregistrer le total des consommations de boissons qui sont ajoutées à l'addition :



Quant à l'addition (appui sur le bouton "Encaisser"), on peut vouloir simplement la visualiser, ou l'imprimer ou encore l'encaisser (remarquer les tooltips) :



Pour l'instant, le seul effet d'un choix sera d'afficher dans la console l'opération choisie (voir plus loin pour plus évolué).

6. La portabilité

Pour rappel, afin d'illustrer la portabilité de Java,

l'application demandée doit fonctionner sur une machine **Windows** **et** sur une machine **UNIX** de l'INPRES.

Concrètement, l'application peut être lancée depuis Netbeans sur la machine de développement, mais **par la ligne de commande "java -jar …"** (donc, pas avec Netbeans) sur la machine Unix Sunray. Il faut être capable d'expliquer le contenu du fichier manifeste.

Evaluation 2 *(to be continued)*

Dossier :
- ……..
- …….

III. Persistance et communication des données du projet "Le gourmet audacieux"

6. Properties et sérialisation

7. L'application ApplicationCuisine

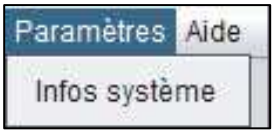
IV. Quelques fonctionnalités supplémentaires

8. Menu, Jtable et/ou JTree

9. L'addition : améliorations

10. Les menus Paramètres et Aide

Dans le menu principal, on remarque aussi l'existence d'items qui sont alignés à droite ("Paramètres" et "Aide").



Pour obtenir cela, le plus simple est d'ajouter dans le constructeur de la fenêtre (si jMenuBar1 désigne la barre de menu) :

jMenuBar1.add(Box.createHorizontalGlue()); //Espace horizontal pour aligner à droite

puis d'ajouter manuellement à cette barre les items de menus et sous-menus.

V. Flux et Java Beans

11. Les fichiers de log et les fichiers serveurs

12. Une chaîne de beans dans la cuisine