



**ENSICAEN**

6, bd maréchal Juin  
F-14050 Caen cedex 4

Spécialité Informatique

2<sup>e</sup> année

**ENSICAEN**

Rapport de projet

---

# Logiciel de gestion de caisse et d'ardoises

---

BOIRLEAUD Pierre-Jean

MOLINS Nicolas

REHFELD Vincent

# Table des matières

<b>INTRODUCTION</b> .....	<b>3</b>
<b>1. CAHIER DES CHARGES</b> .....	<b>4</b>
1.1. BASE DE DONNEES RELATIVE AUX OPERATIONS ET AUX ARDOISES DES CLIENTS .....	4
1.1.1. <i>Table des opérations (table 'transaction')</i> .....	4
1.1.2. <i>Table des ardoises des clients (table 'ardoise')</i> .....	5
1.2. L'APPLICATION .....	5
1.2.1. <i>Fenêtre principale : la caisse</i> .....	5
1.2.2. <i>Fenêtre de configuration des boutons prédéfinis</i> .....	6
1.2.3. <i>Fenêtre de gestion des ardoises</i> .....	6
1.2.4. <i>L'utilitaire de gestion des comptes</i> .....	6
<b>2. ENVIRONNEMENT DU PROJET - OUTILS UTILISES</b> .....	<b>7</b>
2.1. LE JAVA ET L'ENVIRONNEMENT DE DEVELOPPEMENT JBUILDER X .....	7
2.2. MYSQL .....	7
2.3. L' HTML POUR LA CONSULTATION DES COMPTES .....	8
<b>3. REALISATION DE L'APPLICATION</b> .....	<b>8</b>
3.1. LA BASE DE DONNEES .....	8
3.1.1. <i>Les deux tables</i> .....	8
3.1.2. <i>Connexion en JAVA à la base de données, requêtes</i> .....	9
3.1.3. <i>Installation de la Base de Données</i> .....	9
3.2. MODELE UML SIMPLIFIE DE L'APPLICATION .....	9
3.3. AUTOUR DE LA CAISSE .....	10
3.4. AUTOUR DE LA FENETRE DE CONFIGURATION DES 21 BOUTONS PREDEFINIS .....	12
3.5. GESTION DES ARDOISES DES CLIENTS .....	13
3.6. GENERATION DES COMPTES .....	14
<b>CONCLUSION</b> .....	<b>16</b>

## Introduction

La tenue d'un bar restaurant implique un certain nombre de contraintes notamment dans la mise à jour des comptes et des ardoises. En effet, il est impératif de calculer chaque soir les recettes du jour et le gérant n'a pas le temps de noter toutes les transactions réalisées, principalement aux heures de pointes. Bien que seul le résultat du jour soit absolument nécessaire, il peut aussi être intéressant pour le commerçant d'avoir des traces plus complètes des opérations qu'il a effectuées. De plus, dans le cas d'un bar-restaurant, nombreux sont les clients qui se permettent de payer en retard, ce qui implique un fastidieux travail de gestion des ardoises sur papier avec tous les désagréments liés à ce support: fragilité, problème de mise à jour...

Le passage à l'informatique permettrait aux gérants de ne plus avoir à se soucier de ce type de problème en mettant à leurs dispositions une mémorisation automatique de ces transactions ainsi qu'une liste informatique de ces ardoises. De même, l'instauration d'un logiciel efficace de gestion de caisse permettrait de réduire notablement l'attente du client.

Un projet de deuxième année, proposé par Mr Laurent Cousin et Mr Sandor Gardiner, consistait à développer une application de gestion de caisse et d'ardoises simple et performante, qui pourrait sans problème être utilisée dans un Bar-Restaurant.

Une première rencontre avec Mr Cousin et Mr Gardiner nous a permis de mettre sur papier le cahier des charges. Sa présentation constituera la première partie du rapport ; nous décrirons ensuite les outils retenus pour développer l'application puis nous présenterons et commenterons sa réalisation.

# 1 Cahier des Charges

Ce projet consistait à programmer un logiciel de gestion de caisse et d'ardoises pour un bar.

Initialement, le client désirait une telle application en résolution d'écran 1024 \* 768 pixels, pour des écrans tactiles. Lors de notre dernière rencontre, ce dernier nous a demandé de faire une deuxième version pour un affichage en 800 \* 600. Deux Cd-rom (un pour chaque résolution d'écran) ont été conçus pour permettre au barman d'installer avec la plus grande simplicité possible l'ensemble des composants nécessaires au bon fonctionnement de notre application. Ces Cd-rom comportent aussi une notice d'installation et d'utilisation complète.

Les trois principales caractéristiques à développer dans ce projet sont :

- Une interface permettant à un individu n'ayant aucune connaissance en informatique de tenir la caisse. L'interface doit être dessinée pour un écran tactile.
- Un système de gestion d'ardoises accessible depuis la caisse.
- Un logiciel permettant d'imprimer chaque soir les transactions du jour, au besoin depuis un ordinateur différent de la caisse mais relié par un réseau local.

Nous avons pris contact avec Mr Gardiner et établi le cahier des charges :

## **1.1 Base de données relative aux opérations et aux ardoises des clients**

### **1.1.1 Table des opérations (table 'transaction')**

Toutes les opérations effectuées au cours d'une journée doivent être stockées dans cette table, dont voici les champs :

Le numéro de l'opération : ce numéro est auto incrémenté.

Le montant en euros : positif si créditeur pour le barman, négatif sinon.

La date de l'opération.

L'heure de l'opération.

Le type de paiement : espèces, carte bancaire, chèque, ou remboursement d'une ardoise.

Lieu de paiement : restaurant ou bar.

### **1.1.2 Table des ardoises des clients (table 'ardoise')**

L'application doit permettre de créer, supprimer, rembourser (totalement ou en partie) des ardoises pour les clients du bar. Voici les champs de la table correspondante :

Le numéro de l'ardoise : ce numéro est auto incrémenté.

Le nom du client : chaîne de 80 caractères maximum.

Le montant : somme positive en euros que doit actuellement le client.

## **1.2 L'application**

Elle doit se composer de fenêtres (notamment une pour la caisse et une pour la gestion des ardoises) composées de boutons de taille suffisante pour que l'utilisateur puisse taper à l'écran aisément et rapidement. Elle doit par ailleurs être la plus intuitive possible, mais aussi la plus robuste possible. Enfin, cette application doit fonctionner avec autant d'ordinateurs que le barman souhaite installer : l'un exécute alors un serveur de base de données, et les autres sont des clients de cette base. Par ailleurs, un utilitaire à part doit permettre la consultation des comptes.

Voici les éléments de l'application :

### **1.2.1 Fenêtre principale : la caisse**

Celle-ci, à la demande du client doit contenir :

- Un pavé numérique proposant aussi les opérations de multiplication et d'addition (la multiplication restant prioritaire par rapport à l'addition par soucis d'intuitivité), et une virgule. Les nombres sont manipulés avec deux chiffres après la virgule, et sont affichés dans une zone de texte (zone de paiement) à côté du pavé. Cet ensemble doit être placé à droite de l'écran afin de le rendre plus maniable pour les droitiers.

- Un jeu de boutons pour informer sur le type et le lieu du paiement. Un appui tout d'abord sur l'un des boutons « BAR » ou « RESTO » doit provoquer la simplification du calcul en cours ; puis un appui sur l'un des boutons « CB », « espèces » ou « chèque » doit rentrer le montant de cette opération dans la base de données 'transaction'.

- Un jeu de boutons prédéfinis permettant de rentrer des valeurs rapidement dans la zone de paiement : chaque bouton a comme texte le nom d'une boisson ou autre (souvent commandée), et un appui sur celui-ci provoque l'ajout du tarif de la boisson

correspondante dans la zone de paiement. Ces boutons doivent être entièrement reconfigurables.

- Une liste déroulante des 50 dernières opérations effectuées : chaque ligne dans cette liste doit préciser le type de paiement et le montant. A la demande de Mr Gardiner lors de sa dernière visite, cette liste affiche aussi pendant chaque saisie de commande un détail de celle-ci, c'est-à-dire la liste des consommations saisies à l'aide des boutons prédéfinis.

### **1.2.2 Fenêtre de configuration des boutons prédéfinis**

Cette fenêtre doit permettre à tout moment de modifier facilement (sur l'ordinateur où on l'ouvre) les couples de valeurs « texte / tarif » de chaque bouton. Un retour vers la fenêtre principale doit mettre à jour ses valeurs.

Pour cela, un clavier et un pavé numérique virtuels doivent se trouver dans la fenêtre pour rentrer les valeurs.

Chaque ordinateur comporte sa propre liste de valeurs pour ces boutons, stockée dans un fichier.

### **1.2.3 Fenêtre de gestion des ardoises**

Elle doit permettre de gérer la création, la suppression, et le remboursement total ou partiel des ardoises des clients. Pour cela, la fenêtre doit proposer un clavier alphanumérique pour permettre de saisir les noms et les montants directement à l'écran tactile.

La facilité d'utilisation est primordiale dans une telle fenêtre. Seuls les éléments utiles doivent apparaître dans la fenêtre au bon moment. La sélection d'un client par son nom doit être facilitée par une liste affichant tous les noms commençant par la saisie dans le champ.

### **1.2.4 L'utilitaire de gestion des comptes**

Cet utilitaire doit permettre de générer la liste des opérations effectuées au cours de la journée ainsi que les totaux et les sous totaux repartis par type de paiement. Le but principal de ce programme est de générer une liste présentable des transactions avec l'objectif de les imprimer tous les soirs.

Il doit également permettre d'effectuer une sauvegarde des ardoises en cours en cas de défaillance technique.

## 2 Environnement du projet - Outils utilisés

On nous a laissé le choix sur les outils et langages à utiliser. Nous avons décidé d'utiliser le JAVA avec Borland Jbuilder X pour le côté application et interface graphique, MySQL pour ce qui est de la base de données, et enfin l'HTML pour la consultation des comptes du jour.

Nous avons utilisé chacun de ses langages et/ou outils afin de créer une application de manière efficace ; chacun d'entre eux présente en effet ses propres avantages :

### 2.1 Le JAVA et l'environnement de développement Jbuilder X

JAVA est un langage robuste qui peut être exploité pour développer un large éventail de programmes utilisant une interface utilisateur graphique, pouvant être appliqués en réseau et se connecter à des bases de données, et offrant d'autres fonctionnalités toutes plus sophistiquées les unes que les autres.

Avec JAVA et ses bibliothèques graphiques AWT (Abstract Window Toolkit) et Swing, la conception d'interfaces est aisée (beaucoup plus qu'avec la plupart des autres langages qui souvent proposent des librairies peu « maniables »), et malgré cette simplicité de mise en œuvre, les applications sont agréables à l'œil, et portables. Leur conception est encore plus facilitée grâce, notamment au concepteur inclus dans Jbuilder X.

Java est enfin nativement doté d'un ensemble complet de primitives de gestion du multitâche simplifiant grandement l'écriture de programmes par exemple devant exécuter des requêtes sur une base de données. L'API JDBC (Java DataBase Connectivity), apparue dès la JDK 1.1, permet de développer des applications capables de se connecter à des serveurs de bases de données (SGBD), par le biais d'un pilote. Nous avons utilisé un tel pilote (com.mysql.jdbc.Driver) de manière à pouvoir se connecter à un serveur MySQL.

Pour faire fonctionner l'application, il est nécessaire d'installer sur les postes une machine virtuelle JAVA (JRE).

### 2.2 MySQL

L'emploi de bases de données était absolument indispensable afin de sauvegarder de manière efficace l'ensemble des opérations et les ardoises des clients, qui, à long terme peuvent constituer un bloc très volumineux de données.

Nous avons opté pour le langage MySQL (My Structured Query Language) pour plusieurs raisons :

- la première est que nous l'avons étudié en cours cette année.

- la seconde est qu'il offre un système optimisé de gestion de base de données, et ses commandes sont plutôt faciles d'emploi.
- la troisième raison est qu'il existe un driver JDBC pour se connecter à partir d'une application JAVA sur un serveur MySQL. Et plusieurs clients MySQL peuvent être connectés sur un même serveur MySQL.
- Enfin, MySQL AB, l'éditeur de MySQL, propose ce produit en licence libre « open source ».

### **2.3 l' HTML pour la consultation des comptes**

Nous créons à partir de l'utilitaire de gestion des comptes des fichiers HTML (HyperText Markup Language) recensant l'ensemble des opérations d'une période (période = durée entre deux créations de pages HTML).

Nous avons utilisé ce langage plutôt qu'un autre car il permet de facilement mettre en forme du texte, lui mettre de la couleur, et de disposer les éléments dans des tableaux. Il n'est par ailleurs, du fait de la simplicité de sa syntaxe (langage à base de balises), pas difficile de construire des pages HTML à l'aide de programmes, ou même de scripts.

L'HTML est par ailleurs extrêmement répandu grâce à l'Internet, cela assure la lisibilité de ce type de fichier sur la machine, grâce à un quelconque navigateur Web par exemple.

## 3 Réalisation de l'application

### 3.1 La Base de Données

#### 3.1.1 Les deux tables

- La table contenant les opérations (table 'transaction') se présente de la manière suivante :

```
mysql> show columns from transaction;
```

Field	Type	Null	Key	Default	Extra
num	int(11)		PRI	NULL	auto_increment
montant	float	YES		NULL	
date	date	YES		NULL	
heure	time	YES		NULL	
typePaiement	varchar(10)				
type	varchar(10)	YES		NULL	

6 rows in set (0.00 sec)

Précisons que 'typePaiement' correspond à « espèces », « CB », « chèque », ...  
Type est soit « bar », soit « resto ».

- La table contenant les ardoises des clients du bar (table 'ardoise') se présente quant à elle de cette manière :

```
mysql> show columns from ardoise;
```

Field	Type	Null	Key	Default	Extra
num	int(11)		PRI	NULL	auto_increment
nom	varchar(80)	YES		NULL	
montant	float	YES		NULL	

3 rows in set (0.05 sec)

#### 3.1.2 Connexion en JAVA à la base de données, requêtes

Il faut au préalable charger le driver MySQL :

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

Ensuite, on se connecte de cette manière (hote, log et pass sont des chaînes de caractère) :

```
String url = "jdbc:mysql://" + hote;  
Connection con = DriverManager.getConnection(url, log, pass);  
Statement stmt = con.createStatement();
```

Enfin on peut envoyer les requêtes si aucune exception n'a été préalablement interceptée, de cette manière :

```

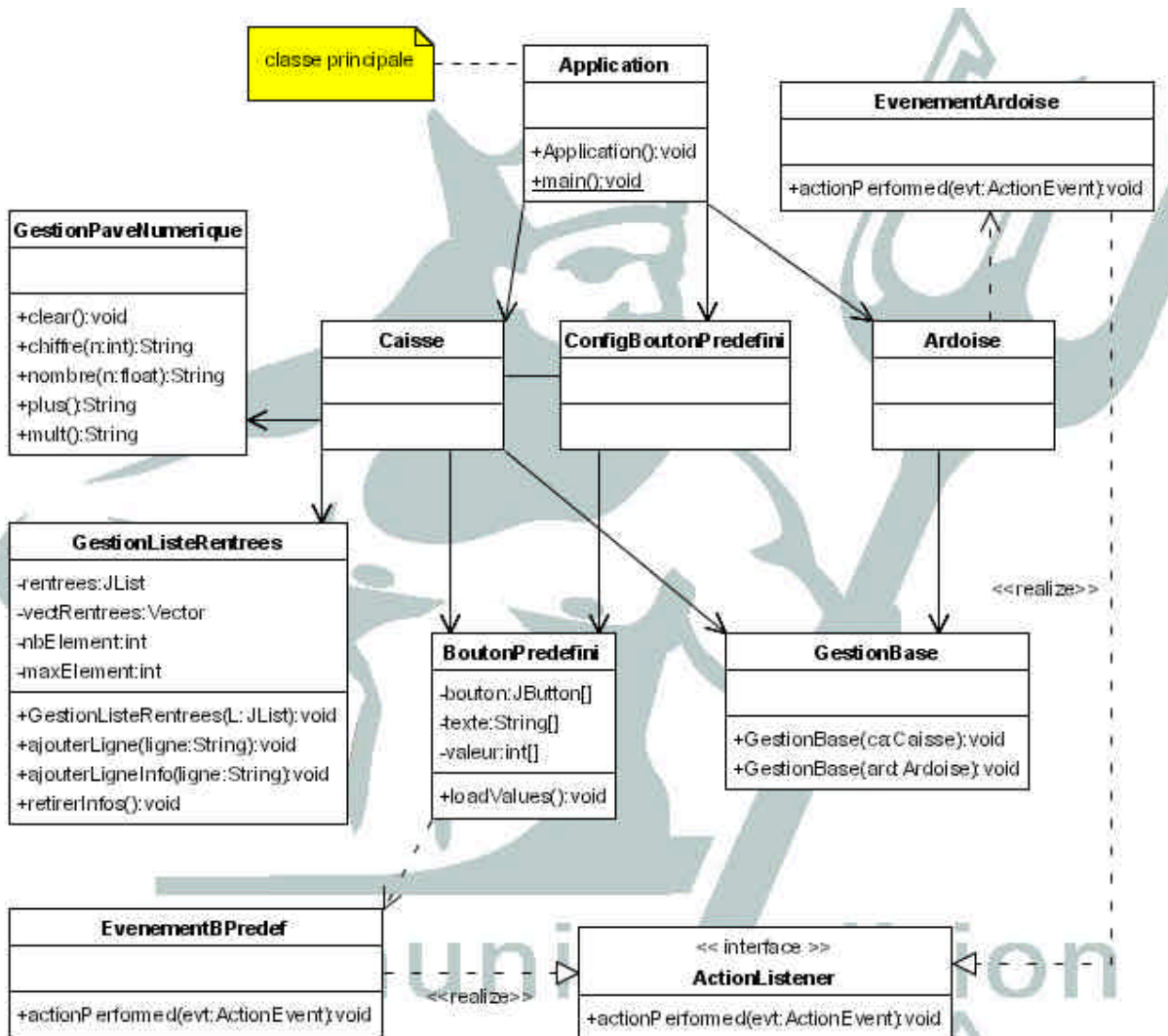
// exemple de récupération du montant que doit un client en ardoise
PreparedStatement requete = con.prepareStatement(
    "SELECT montant FROM ardoise WHERE nom LIKE ?");
requete.setString(1, Nom + "%"); // Nom est une chaine...
ResultSet resultatRequete = requete.executeQuery();
resultatRequete.next();
float f = resultatRequete.getFloat("montant");

```

### 3.1.3 Installation de la Base de Données

Pour que tout fonctionne, il faut installer un serveur MySQL sur un ordinateur, et le pilote JDBC sur chacune des autres machines (clientes de MySQL).

### 3.2 Modèle UML simplifié de l'application



Par souci de clarté, toutes les associations, méthodes, et champs ne figurent pas sur ce schéma (Une large minorité des champs y sont présents).

### 3.3 Autour de la Caisse



Six classes (dont Caisse.java) permettent le fonctionnement de la caisse :

**GestionPaveNumerique.java** gère les actions à effectuer lorsqu'un des boutons du pavé numérique ou l'un des boutons BAR, RESTO, CHEQUE, ESPECES, CB est pressé. Entre autres, c'est grâce à cette classe que les opérations s'effectuent, tout en gérant la priorité de la multiplication par rapport à l'addition. Grâce à cette classe, la saisie peut s'effectuer (on rentre les nombres chiffre par chiffre à l'aide du pavé numérique ou directement grâce aux boutons prédéfinis).

**GestionListeRentrees.java** gère la liste déroulante des opérations (située en haut à gauche de la fenêtre) : elle permet de ne garder que les 50 (ou un autre nombre) dernières opération pour ne pas consommer trop de mémoire ; elle permet de plus d'afficher des lignes « d'informations » (voir la capture d'écran : les 6 dernières lignes en sont) lorsqu'on appui sur l'un des 21 boutons prédéfinis, et de les effacer à chaque fin d'opération.

La classe **BoutonPredefini** génère à sa construction les 21 boutons du bas de la fenêtre de la caisse et de la fenêtre utilisée pour les configurer. Elle gère aussi la lecture et la sauvegarde des textes des boutons et des tarifs correspondants à partir d'un fichier de sauvegarde 'savval.txt' (s'il est absent les textes affichés sont des « ? » et les tarifs nuls).

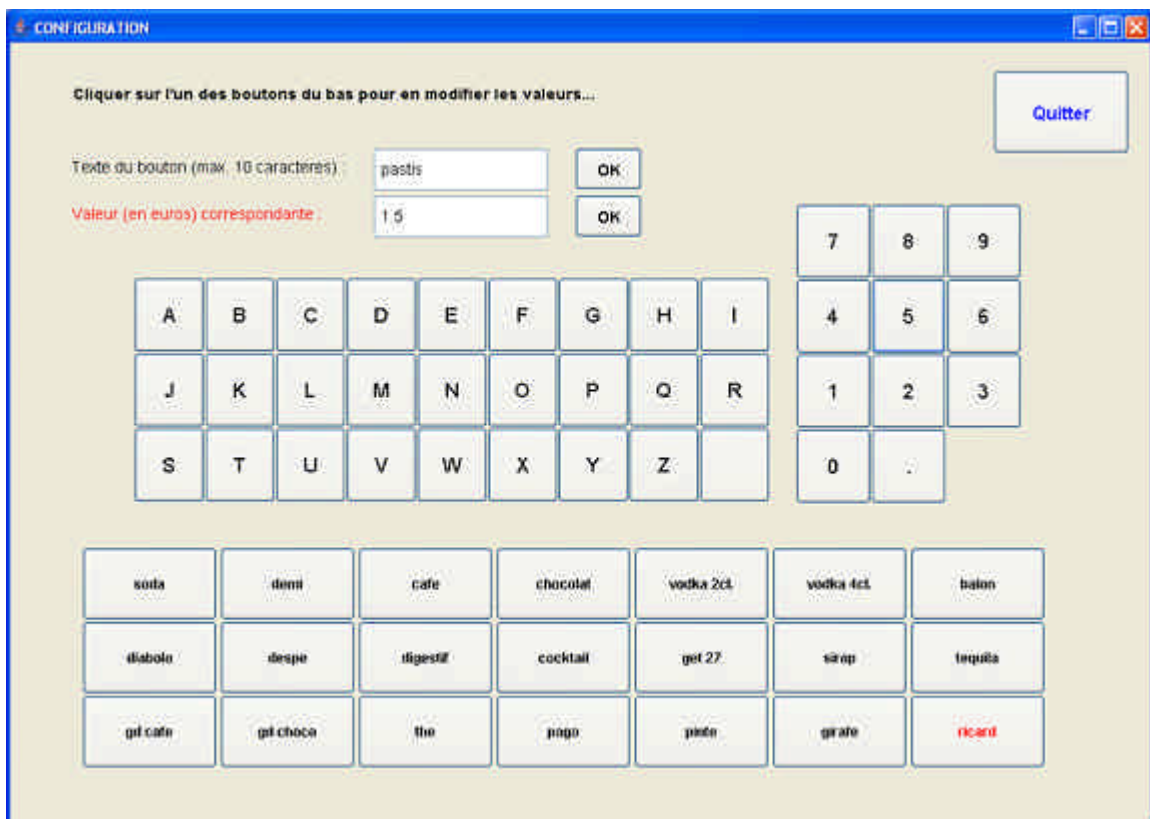
La classe **EvenementBPredef** sert à capturer les évènements (causés par clics) des boutons prédéfinis et à effectuer les bonnes actions ; son comportement est évidemment différent suivant que l'évènement provient de la classe Caisse ou qu'il provient de la classe ConfigBoutonPredefini. Elle gère aussi les évènements du pavé numérique et du clavier de la classe ConfigBoutonPredefini.

La classe **GestionBase** permet de se connecter au serveur MySQL (un message d'erreur apparaît dans la fenêtre en cas d'erreur), et d'effectuer toutes les requêtes utiles pour la sauvegarde des opérations : la méthode ajouterCommande() rajoute une opération (ou commande) dans la base de données.

Enfin, la classe **Caisse** sert à créer la fenêtre et à y afficher les divers éléments la composant. Elle sert par ailleurs de contrôleur : elle gère entre autres les actions à effectuer en cas d'évènement provenant de tous les boutons non gérés par EvenementBPredef.

### 3.4 Autour de la fenêtre de configuration des 21 boutons prédéfinis

Le cahier des charges stipulait que les boutons prédéfinis devaient être reconfigurables, nous avons donc choisi de réaliser une fenêtre à cet effet.



Trois Classes permettent le fonctionnement de cette partie de l'application :

Les classes **BoutonPredefini** et **EvenementBPredef** sont utilisées, comme pour la caisse, sauf que **EvenementBPredef** n'effectue (évidemment) pas les mêmes actions qu'avec la caisse ; notamment on « sélectionne » en rouge le dernier bouton cliqué en appelant une méthode de **BoutonPredefini** prévue à cet effet. **EvenementBPredef** capture aussi les événements du clavier et du pavé numérique de cette fenêtre.

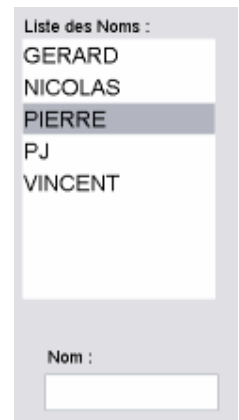
Enfin, la classe **ConfigBoutonPredefini** crée la fenêtre correspondante, y place les éléments, et définit tout un tas de méthodes permettant le fonctionnement des demandes à l'utilisateur de modification des valeurs du bouton actuellement sélectionné.

### 3.5 Gestion des ardoises des clients

La classe **Ardoise** étendant la classe **JFrame**. Elle comporte tous les éléments et boutons permettant de gérer les ardoises.

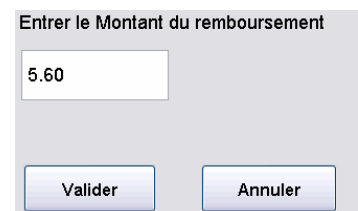
Un objet de type **JList** permet d'afficher tous les éléments de la base de données correspondants au nom ou début de nom entré dans l'objet de type **TextField** en dessous. Chaque nom affiché dans cette liste est sélectionnable et le nom choisi permet d'afficher le montant et de gérer l'ardoise lui correspondant. Un bouton permet d'effacer le nom entré dans cette zone de texte. La liste est remise à jour à chaque caractère ajouté ou retiré.

Un objet **TextField** non éditable affiche le montant correspondant au nom sélectionné.



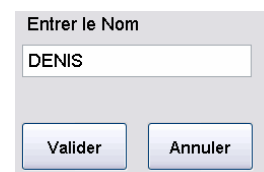
Trois boutons principaux permettent d'accéder aux trois fonctions de cette classe **Ardoise**.

- Le bouton "Remboursement " permet, par le biais d'un **JPanel** apparaissant lors du clic, de diminuer le montant correspondant au nom sélectionné.



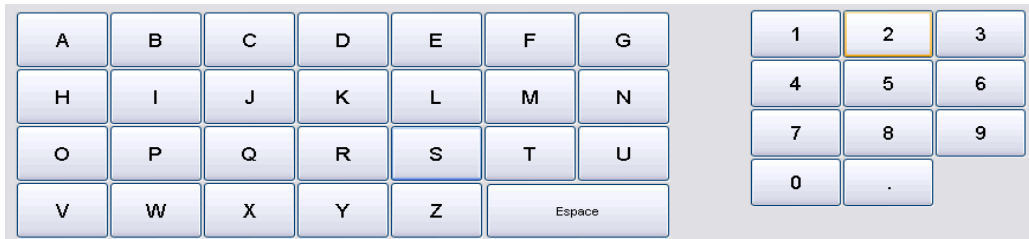
- Le bouton "Ajouter sur Ardoise" permet, par le biais d'un autre **JPanel** apparaissant, d'augmenter le montant correspondant au nom sélectionné.

- Le bouton "Ajouter un nom", faisant aussi apparaître un **JPanel**, permet d'ajouter un nom à la base de données.



Tous ces JPanel sont situés au même endroit du JFrame mais un seul est visible à chaque fois.

Conformément au cahier des charges requérant un clavier virtuel, deux JPanel sont présents, l'un pour les touches alphabétiques, l'autre pour les touches numériques.



Ces claviers virtuels permettent d'écrire le caractère choisi dans le champ du JPanel activé ou dans le champ " Nom " si aucun n'est actif.

Le JPanel correspondant aux touches alphabétiques disparaît si ses caractères ne sont pas autorisés dans le champ actif (augmentation ou diminution du montant). Les événements provoqués par l'activation d'une de ces touches sont gérés par la classe *Evenement*, décrite ci-dessous.

La classe **Evenement** gère tout clic sur l'une des touches des panels alphanumériques en récupérant le caractère choisi et en le renvoyant à la classe *Ardoise* qui l'ajoutera au champ de texte approprié.

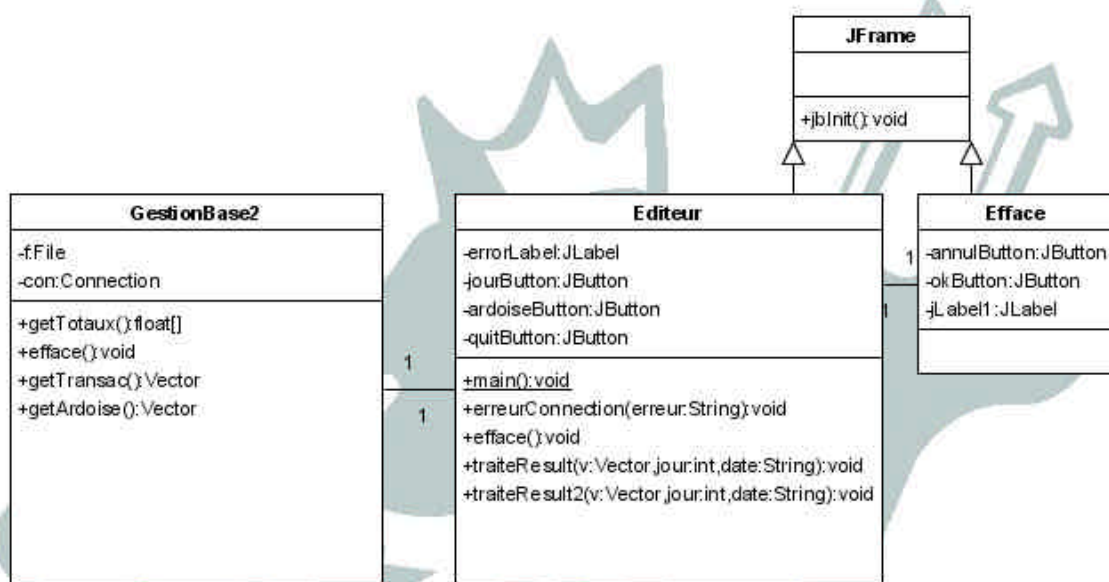
La classe **GestionBase** est utilisée pour accéder à la base de données MySQL. Ses méthodes permettent de récupérer la liste des noms ou un montant, de mettre à jour un montant, d'ajouter ou de supprimer un nom.

### 3.6 Génération des comptes

Nous avons choisi de réaliser un programme indépendant pour l'impression des comptes. En effet, il nous paraissait plus logique de séparer le logiciel gérant la caisse et celui permettant d'effectuer la comptabilité. Notamment pour éviter tout risque de fausse manipulation.



Nous avons créé trois classes :



La classe **Editeur** étendant la classe JFrame comporte tous les éléments et boutons permettant l'écriture des comptes. Elle contient trois JButton :

- Le bouton "Editer compte" permet de créer un fichier html présentant les comptes. Il appelle une méthode de la classe GestionBase2 qui lui retourne les transactions de la base de données dans un Vector. A partir du vecteur, le fichier html est créé dans le répertoire compte créé au préalable par l'installateur grâce à une méthode de la classe. Enfin le navigateur par défaut du système d'exploitation est automatiquement ouvert sur la page nouvellement créée. Le fichier html comporte les diverses opérations de la journée ainsi que le total des montants. Il contient également les totaux répartis par type de paiement.

- Le bouton "sauvegarde des ardoises" permet d'enregistrer la base de données des ardoises dans un fichier html. Sa création est identique à celle du fichier précédent. Le fichier html final contient les noms et les montants associés.

- Le bouton "quitter" permet de quitter l'application

La classe **GestionBase2** est utilisée pour accéder à la base de données MySQL. Ses méthodes permettent de récupérer la liste des transactions, les différents totaux du jour, la liste des ardoises et de vider la base de données des transactions.

La classe **Efface** étendant la classe JFrame est une simple fenêtre permettant de demander si l'utilisateur souhaite purger la base de données. Elle est affichée après la création du fichier html des transactions. Pour une utilisation normale l'utilisateur doit toujours cliquer sur OK, en effet le logiciel n'est censé être utilisé qu'une fois par jour, ce qui permet d'avoir seulement les comptes de la journée en mémoire.

## **CONCLUSION**

Le développement d'une application de gestion de caisse et d'ardoises destinée à être utilisée régulièrement nous a permis de sortir du contexte scolaire (travaux pratiques) en nous confrontant aux problèmes de la vie professionnelle. En effet nous avons découvert l'importance de la communication avec le client : notre cahier des charges n'a cessé de s'enrichir au fil des rencontres. De plus, nous avons dû au cours du développement apporter de nombreuses corrections dans le but de faire correspondre notre logiciel à la vision qu'en avait Mr Gardiner.

Ce projet nous a permis d'approfondir nos connaissances en JAVA, notamment en interface graphique mais aussi en base de données. Cela nous a également permis d'expérimenter les étapes, le fonctionnement, et aussi les problèmes de la création d'un logiciel en équipe. En effet le travail en trinôme est sensiblement différent du travail en binôme, notamment au niveau de la répartition des tâches. De plus le délai important attribué nous a posé de nombreux problèmes d'organisation.