RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED
FACULTÉ DES SCIENCES EXACTES
Département D'Informatique

Mémoire de Fin D'étude
Présenté pour l'obtention du Diplôme de

LICENCE ACADEMIQUE

Domaine : **Mathématique et Informatique**
Filière : **Informatique**
Spécialité : **Systèmes Informatiques**

Présenté par :

- **SOUALAH Mouhammed Adel**
- **ZEKRI Mouhammed El Hocine**
- **LEKHOUIMES Soltane**

Thème

Réalisation d'une calculatrice de tarification

Proposé et Encadré par : **Madame GUIA Sana Sahar**

Soutenue le 24-05- 2017 Devant le jury :

Mr.	**kertiou Ismail**	MCA	Président
Mr.	**Gherbi Kdour**	MAA	Rapporteur

Année Universitaire : 2016-2017

Remerciement

Nous tenons à remercier avant tout Dieu tout puissant qui nous a donné la volonté, la force et la patience pour élaborer notre travail.

Nous remercions en particulier notre encadreur

Madame GUIA Sana Sahar

qui nous a aidés et conseillés durant Cette année.

Enfin, que tous ceux qui ont participé de près ou de loin à la réalisation de ce travail, trouvent ici l'expression de notre gratitude.

Dédicace

Je dédie ce modeste travail chers
parents ma mère et mon père
Tous me chers frères
Dr Issam Eddine Chaib,
Massaoudi Smail.
Qui ont été toujours à côté de moi,
Et toute la famille sans exception.
Tous les amis qui ont partagé
Mes joies et mes douleurs.
Tous les chers collègues pendant les années d'études.

Abstract

In this work calculation problem. This is to allow an employee to calculate the entrance fee to a show.

The objective of our work is to develop an application to Pricing calculator with Java, and object-oriented modeling by the UML modeling language.

In fact, this application will have to include many advantages namely to facilitate the work of the employer and to save time by reducing the waiting time of the customers.

KEYWORDS: Calculator, Java, object-oriented language, pricing.

Résumé

Dans ce travail problème de calcul. Il s'agit de permettre à un employé de calculer le tarif d'entrée à un spectacle.

L'objectif de notre travail est de développer une application de gestion des guichets « calculatrice tarification » à l'aide du langage de modélisation orienté objet UML, et en utilisant un langage de programmation Java.

En effet, cette application devra comporter beaucoup d'avantages à savoir facilite le travail de l'employeur et gagner le temps en réduisant le temps d'attente des clients.

MOTS CLES : Calculatrice, Java, langage orientée objet, tarification.

ملـــــخص

في هذا العمل هو السماح للموظف لحساب رسوم الدخول إلى المعرض.

الهدف من عملنا هو برمجة تطبيق إدارة البوابة "حاسبة التسعير" بواسطة لغة JAVA، والتحليل بواسطة لغة النمذجة الموحدة UML.

في الواقع، فإن هذا التطبيق يتميز بالإيجابيات التالية، يسهل من العمل بالإضافة إلى توفير الوقت عن طريق تقليل وقت الانتظار للعملاء.

الكلمات الرئيسية هي: **آلة حاسبة، لغة البرمجة الموجهة، التسعير.**

Liste des figures

Table de matières

1 Introduction générale :

L'exemple que nous allons traiter dans ce chapitre problème de calcul. Il s'agit de permettre à un employé de calculer le tarif d'entrée à un spectacle. La direction souhaite disposer d'une calculatrice permettant en fonction de la catégorie des clients de déterminer le tarif à payer. Dans un premier temps, il n'est pas demandé de garder une trace des calculs effectués. Par la suite, il est envisageable de garder une trace papier ainsi qu'une trace sur une mémoire électronique à des fins de comptabilité journalière, hebdomadaire et mensuelle. La fonction essentielle du système à réaliser il s'agit en fait ici d'une toute petite application est représentée par le cas d'utilisation du système exprimé par la figure.

Nous souhaitons réaliser une calculatrice permettant à un employé de calculer le tarif d'entrer à un spectacle en fonction de la catégorie des clients.

Les spécifications du problème sont les suivantes : Nous considérons trois catégories de clientèle.

- Les enfants dont l'âge est au plus égal à douze ans.
- Les adultes sont les personnes qui n'appartiennent pas à la première catégorie.
- Les groupes de personnes qui sont constitués d'au moins sept adultes.

Le logiciel à concevoir doit permettre d'effectuer des enchaînements de calcul afin de déterminer le tarif à payer.

Pour des tarifs ou des sommes d'argents, il est préférable d'introduire des objets à la place des données élémentaires, ce qui facilite l'ajout d'une nouvelle catégorie ou leur réutilisation dans d'autres applications.

L'objectif de ce rapport est la réalisation d'une calculatrice de tarification qui permet à un employé de calculer le tarif à payer en fonction des catégories des clients, ce qui permet de calculer le coût total rapidement Et programmé par La programmation orientée objet.

Est un modèle de langage de programmation qui s'articule autour d'objets et de données, plutôt que d'actions et de logique. Par le passé, un programme était une procédure logique qui récupérait des données en entrée, les traitait puis produisait des données en sortie. [1]

2 Plan De Mémoire :

Ce rapport se compose de trois (2) chapitres :

Chapitre II : Ce chapitre est destiné à la modélisation de l'application, en se basant sur le besoin de l'utilisateur final de la calculatrice de tarification, et ceci en utilisant le langage de modélisation UML.

Chapitre III : Ce chapitre présente l'implémentation de calculatrice de tarification en se basant sur la modélisation du chapitre précédent.

Chapitre

1

La modélisation

1.1 Introduction :

Dans ce chapitre nous allons modéliser le problème de la réalisation d'une calculatrice de tarification qui permet de calculer le tarif des clients selon leurs catégories. Afin d'arriver à une bonne modélisation et puisque le problème est résolu en utilisant la programmation orientée objet nous avons choisi le langage de modélisation UML.

1.2 Langage de modélisation UML :

1.2.1 Definition UML :

UML (en anglais Unified Mödling Language) ou « langage de modélisation unifié » est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

UML est à présent un standard défini par l'Object Management Group (OMG). L'OMG diffuse depuis novembre 2007 la version UML 2.1.2, et travaille à présent sur la version. [2]

1.2.2 Les classifications des diagrammes :

Les 9 diagrammes de celle-ci se répartissent en trois grands groupes : [2]

1. Diagrammes structurels ou diagrammes statiques (*UML Structure*)

1.1. Diagramme de cas d'utilisation (*Use case diagramme*)

1.2. Diagramme de classes (*Class diagramme*)

1.3**.** Diagramme d'objets (*Object diagramme*)

1.4. Diagramme de composants (*Component diagramme*)

1.5. Diagramme de déploiement (*Déploiement diagramme*)

2. Diagrammes comportementaux ou diagrammes dynamiques (*UML Behavior*)

2.1-diagramme d'activités (*Activité diagramme*)

2.2-diagramme d'états-transitions (*State machine diagramme*)

3. diagrammes d'interaction (*Interaction diagramme*)

3.1. Diagramme de séquence (*Séquence diagramme*)

3.2. Diagramme de communication (*Communication diagramme*)

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à l'occasion d'une modélisation.

1.3 L'analyse des diagrammes UML :

1.3.1 Diagramme de cas d'utilisation :

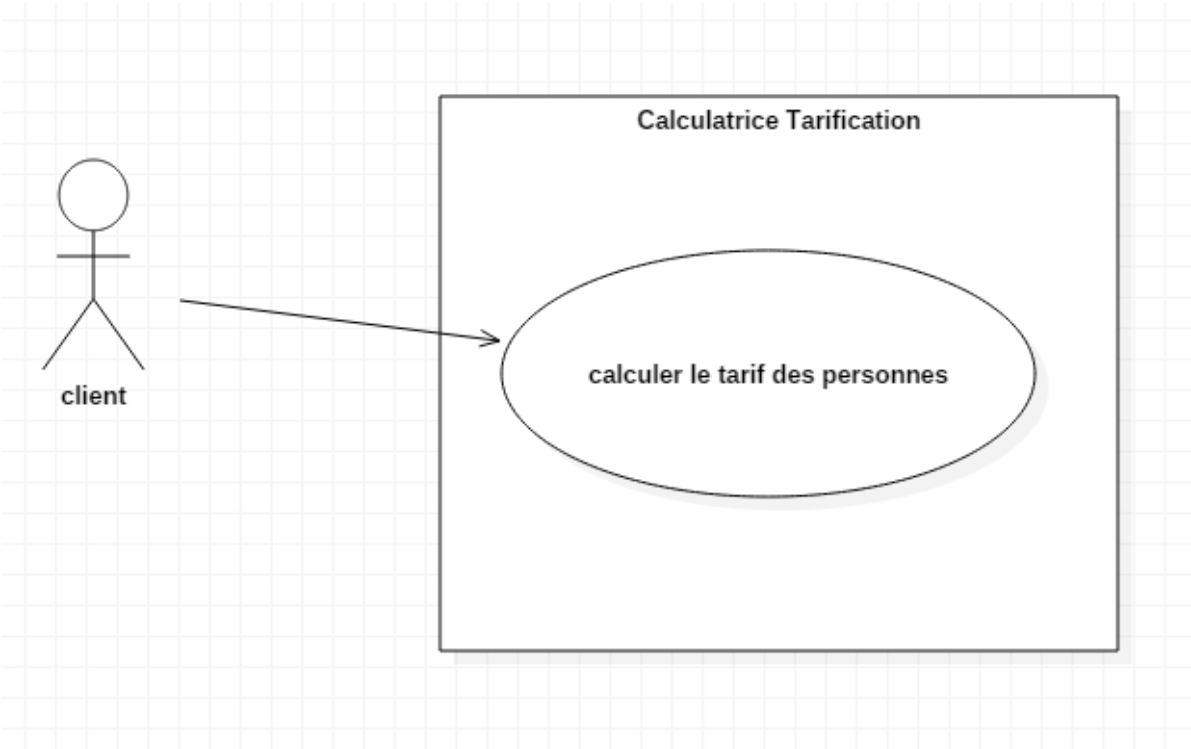

Figure2.1 : diagrammes de Cas d'utilisation.

Cas d'utilisation Calculer le tarif des personnes.

Description du sommaire	
Titre	Calculer le tarif des personnes
But	permet à l'employé de calculer le tarif d'entrée à un spectacle selon les catégories des clients
Acteurs	Utilisateur

1.3.2 Diagramme de séquence :

Les diagrammes des séquences documentent les interactions à mettre en œuvre entre les classes pour réaliser un résultat, tel qu'un cas d'utilisation. UML étant conçu pour la programmation orientée objet, ces communications entre les classes sont reconnues comme des messages. Le diagramme des séquences énumère des objets horizontalement, et le temps verticalement. Il modélise l'exécution des différents messages en fonction du temps.[3]

Voici le diagramme de séquence qui correspondre à notre application, qui illustre le scénario nominal de calcule montant.

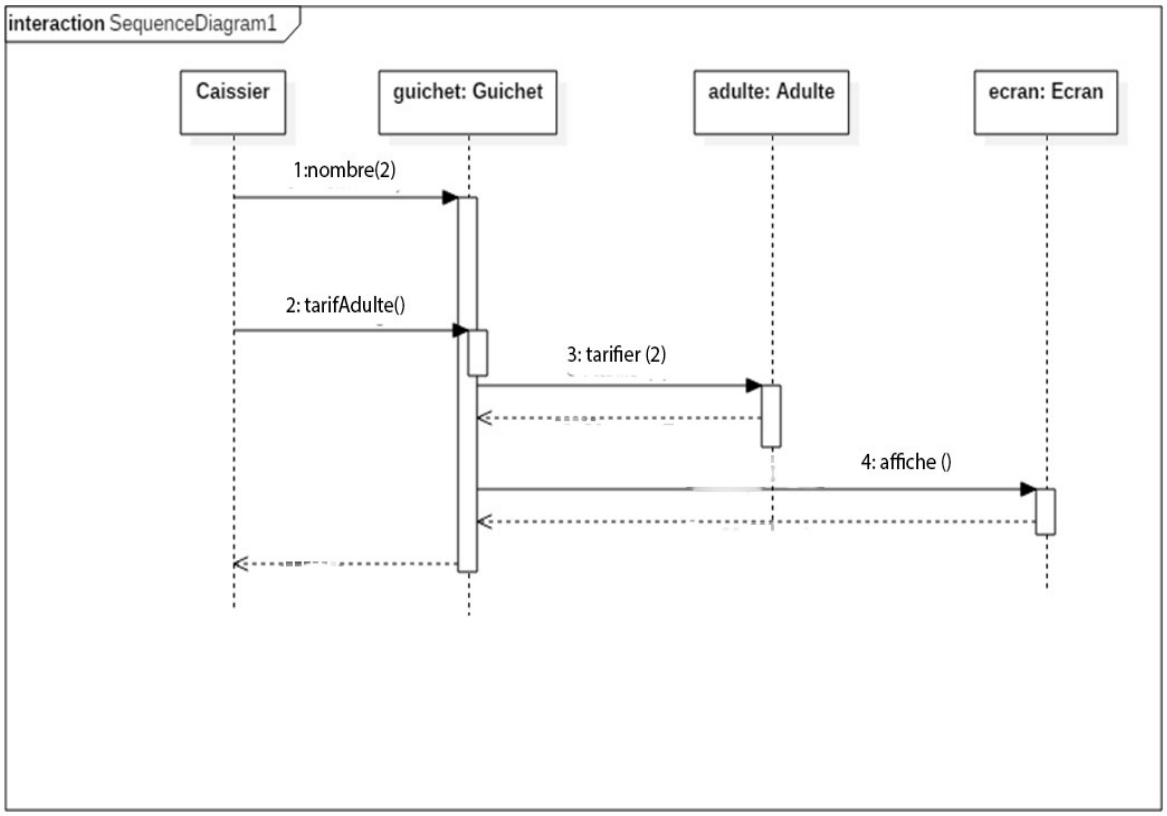

Figure2.3 : diagrammes de séquences.

1.3.3 Diagramme de classes :

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens**.**

La figure ci-dessus représente le diagramme de classe de la Calculatrice de Tarification dans lequel il est décrit les différents composants de l'application :

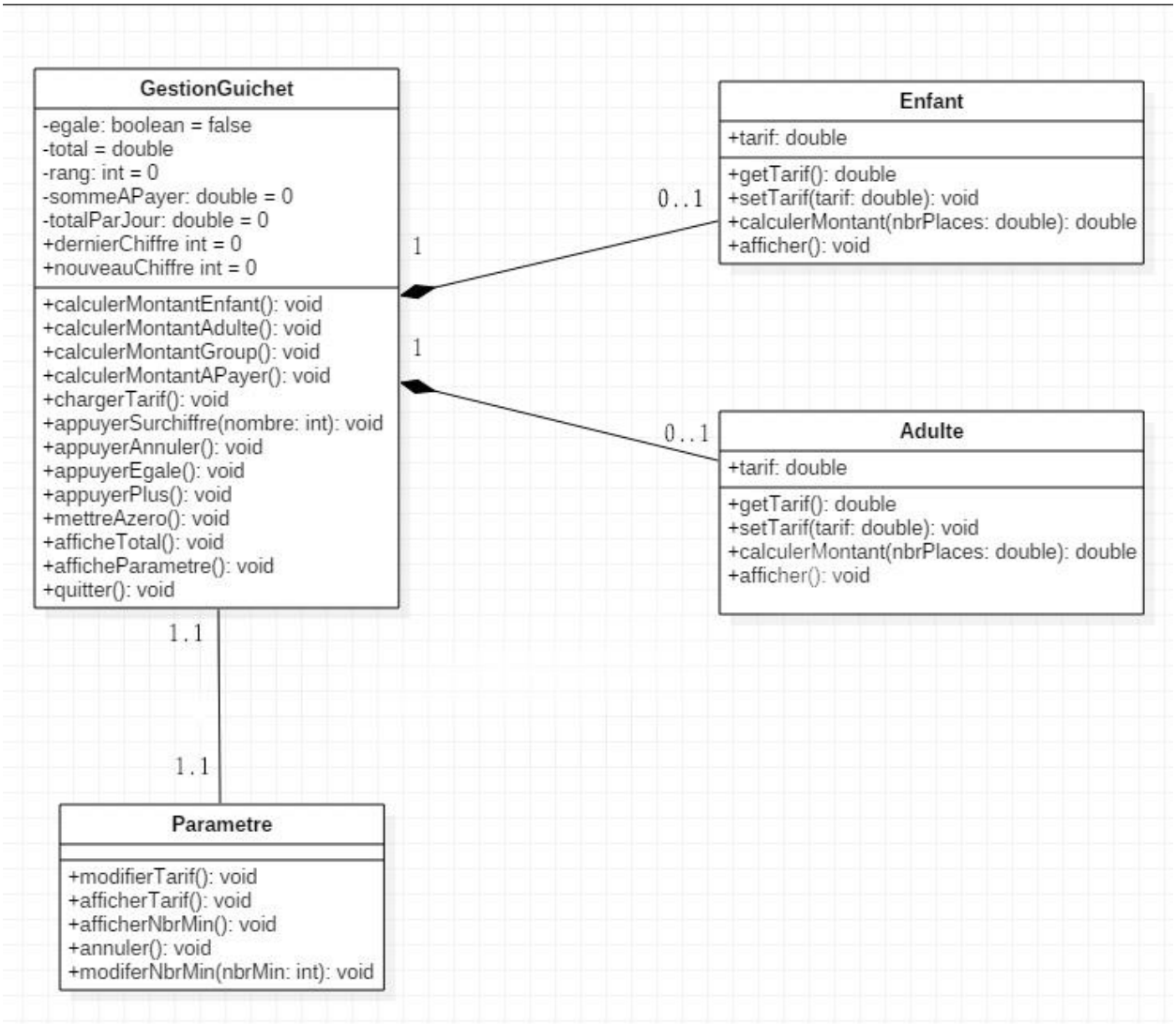

Figure2.2 : diagramme de classes.

1.3.4 Les diagrammes d'états-transitions :

Les diagrammes d'états-transitions d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode).

Ils spécifient habituellement le comportement d'une instance de classeur classe, mais parfois aussi le comportement interne d'autres éléments tels que les cas d'utilisation, les sous-systèmes, les méthodes. [3]

La figure au-dessous illustre le diagramme d'états-transitions pour notre application.

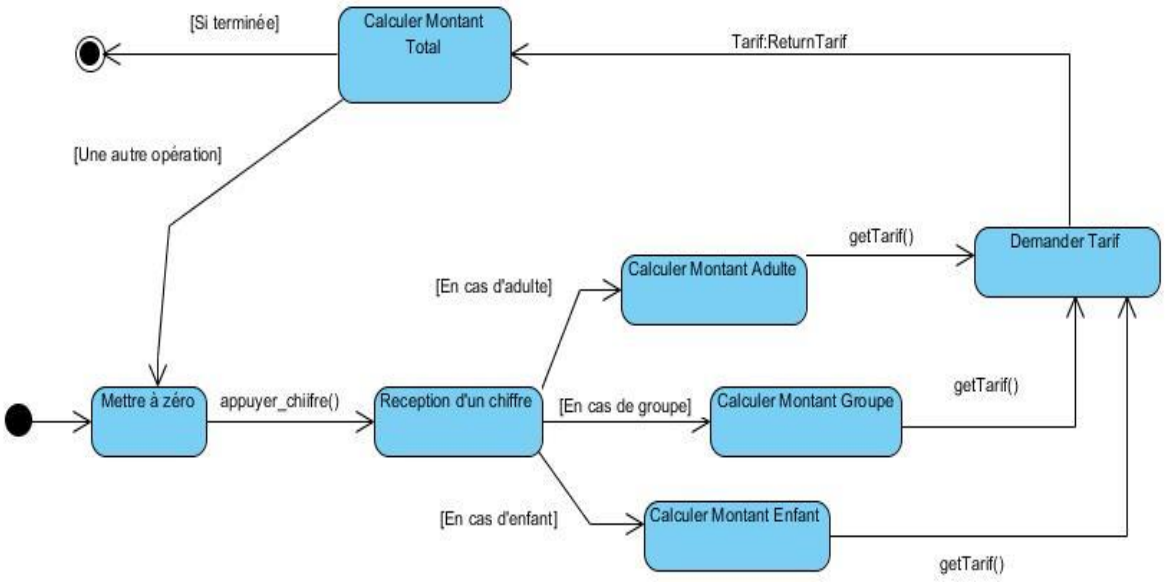

Figure2.4 : Diagramme d'états-transitions.

1.4 Conclusion :

Dans le but de faciliter l'implémentation de notre système d'un calculatrice de tarification, dans ce chapitre, nous avons présenté une capture des diagrammes utilisés lors de la modélisation suivi une démarche de développement UML. Les résultats de ce chapitre seront enrichis par de détails d'implémentation dans le chapitre suivant pour réaliser notre Programmation.

Chapitre 2

Implémentation

2.1 Introduction :

Ce chapitre est consacré à la partie pratique de notre travail, nous allons implémenter la calculatrice de tarification en suivant la modélisation détaillée dans le chapitre précédent. Nous commençons tout d'abord par la description de l'environnement de travail. Ensuite nous présentons notre application.

2.2 Choix du langage de programmation :

2.2.1 Java :

Java est un langage de programmation orienté objet, développé par Sun Microsystems et destiné à fonctionner dans une machine virtuelle, il permet de créer des logiciels compatibles avec des nombreux systèmes d'exploitation.

Java et non seulement un langage de programmation puissant conçu pour être sûr, inter plateformes et international, mais aussi un environnement de développement qui est continuellement étendu pour fournir des nouvelles caractéristiques et des bibliothèques permettant de gérer de manière élégante des problèmes traditionnellement complexes dans les langages de programmation classiques, tels que le multithreading, les accès aux bases des données, la programmation réseau, l'informatique répartie. En plus java est considéré comme un langage adaptable aux plusieurs domaines puisque une application web implémentée par celle-ci peut avoir des extensions ou des modifications dans le futur. De plus, java permet de réduire le temps de développement d'une application grâce à la réutilisation du code développé. [4]

2.3 Environnement de développement :

2.3.1 NetBeans :

NetBeans IDE est un environnement de développement intégré (EDI) modulaire basé sur des normes, écrit dans le langage de programmation Java. Le projet de NetBeans IDE consiste en un EDI Open Source complet écrit dans le langage de programmation Java et en une plateforme

d'application cliente riche, qui peut être utilisée comme structure générique pour créer n'importe quel type d'application. [5]

Voici une liste de choses que l'IDE fait pour simplifier le développement d'Applications Web :

- Fournit un serveur Web Tom cat pour déployer, tester et déboguer vos applications.
- Définit le fichier et la structure de dossier d'une application web pour vous.
- Génère et maintient le contenu des descripteurs de déploiement.
- S'assure que les fichiers de configuration qui apparaissent dans le dossier WEBINF de votre application ne sont pas effacés lorsque vous exécutez la commande Clean pour enlever les résultats des builds précédents.
- Fournit une coloration syntaxique, complétion de code (L'éditeur texte reconnait les mots clés du code source et les colorie selon leur utilisation et leur type : mot clé du langage, variable, fonction, chaine de caractère, commentaire...etc.).
- Fournit un support de débogage compréhensif, incluant le mode pas à pas dans les fichiers JSP et le traçage des requêtes HTTP.

2.3.2 Pourquoi utiliser la Plateforme NetBeans :

Il y aura toujours une place dans le monde pour "les clients lourds". La Plateforme NetBeans apporte aux applications bureautiques les mêmes avantages que l'architecture J2EE apporte aux applications coté-serveur :

- **Un contexte de déploiement** rune time pour des fonctionnalités arbitraires qui simplifient le développement.
- **Une boîte à outils** qui permet de gagner beaucoup de temps en développement et d'effort.
- **Un ensemble d'abstractions** qui permet aux développeurs de se concentrer sur la business logique, et non de réécrire de la logique de routine et des composants requis par la plupart des applications.
- **Un ensemble de Standards** pour rehausser et renforcer la consiste cet et l'interopérabilité entre les applications et les systèmes d'exploitation.

En tirant avantage de cette trousse à outils gratuite, basée sur des standards, les développeurs peuvent concevoir des applications complexes plus rapidement, avec une plus grande assurance de robustesse et de concevoir des applications qui résisteront à l'épreuve du temps. [6]

2.3.3 Interface NetBeans:

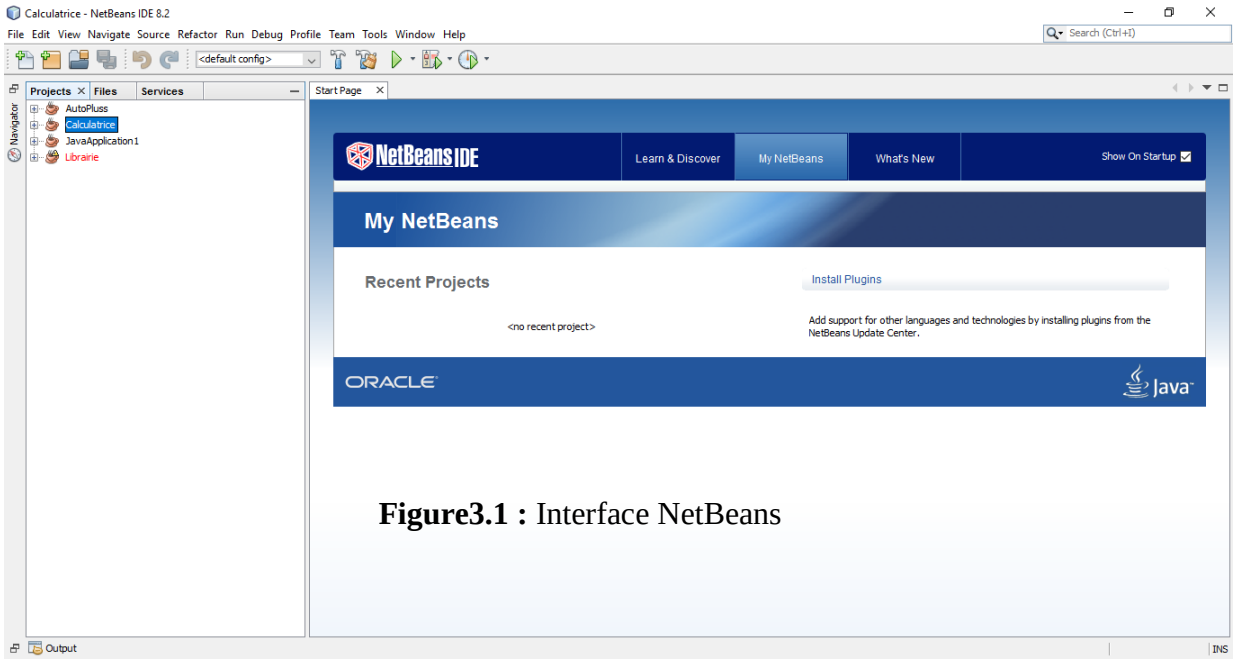

Figure3.1 : Interface NetBeans

Dans la figure suivante représente un exemple de message d'accueil simple en Java NetBeans.

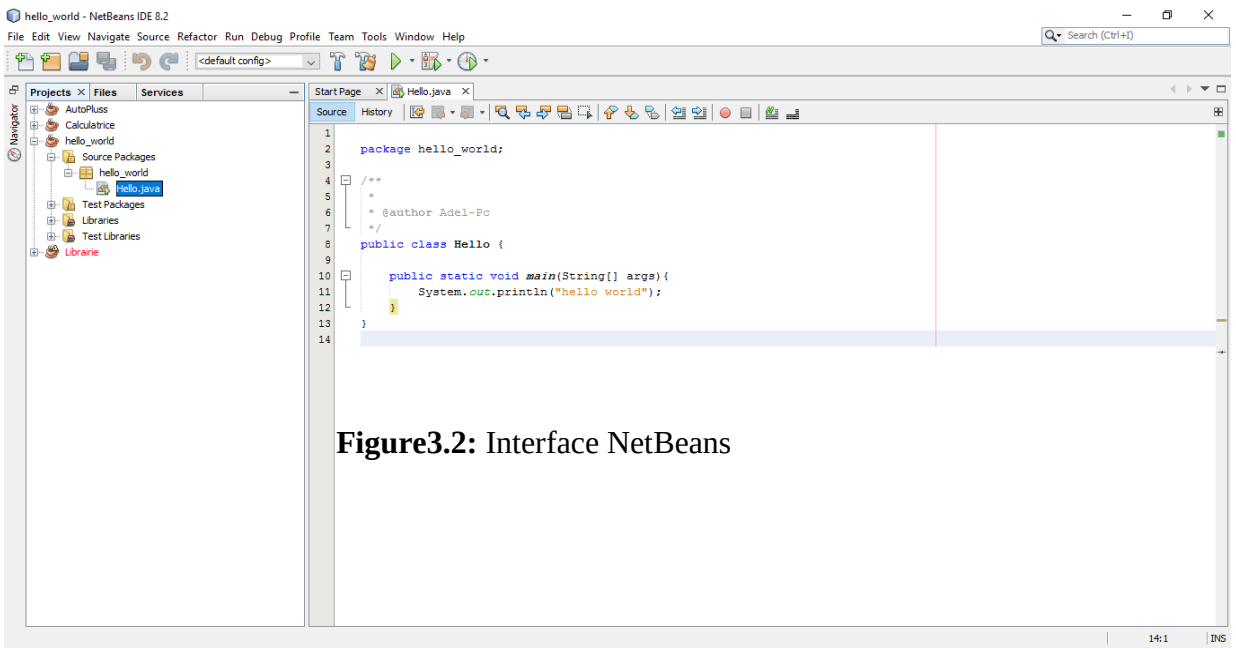

Figure: Exemple de message Hello World

2.4 Lancement du projet :

2.4.1 Menu:

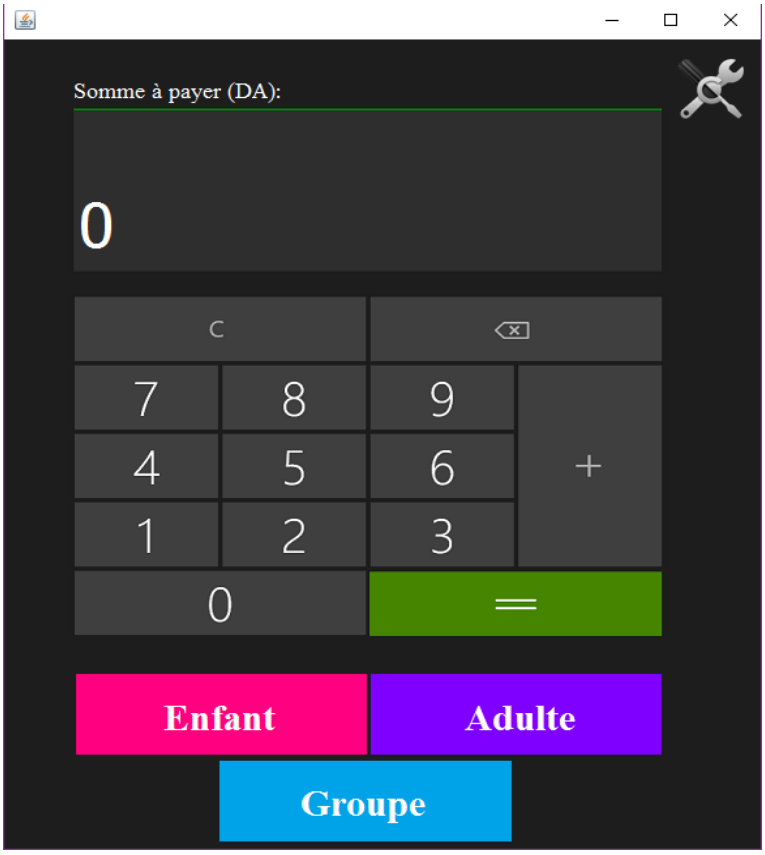

Figure 3.3 : Un aperçu de l'écran de menu

Dans le menu principal, l'' utilisateur peut :

L'utilisateur peut effectuer un enchaînement de calcul pour calculer le tarif à payer, en précisant les catégories des clients :

- choisir enfant en cliquant sur '' Enfant ''.
- choisir adulte en cliquant sur '' Adulte ''.
- choisir groupe en cliquant sur '' Groupe ''.

2.4.2 Ecran de Paramètre :

L'utilisateur peut voir et modification du prix des enfants, adulte, groupe.

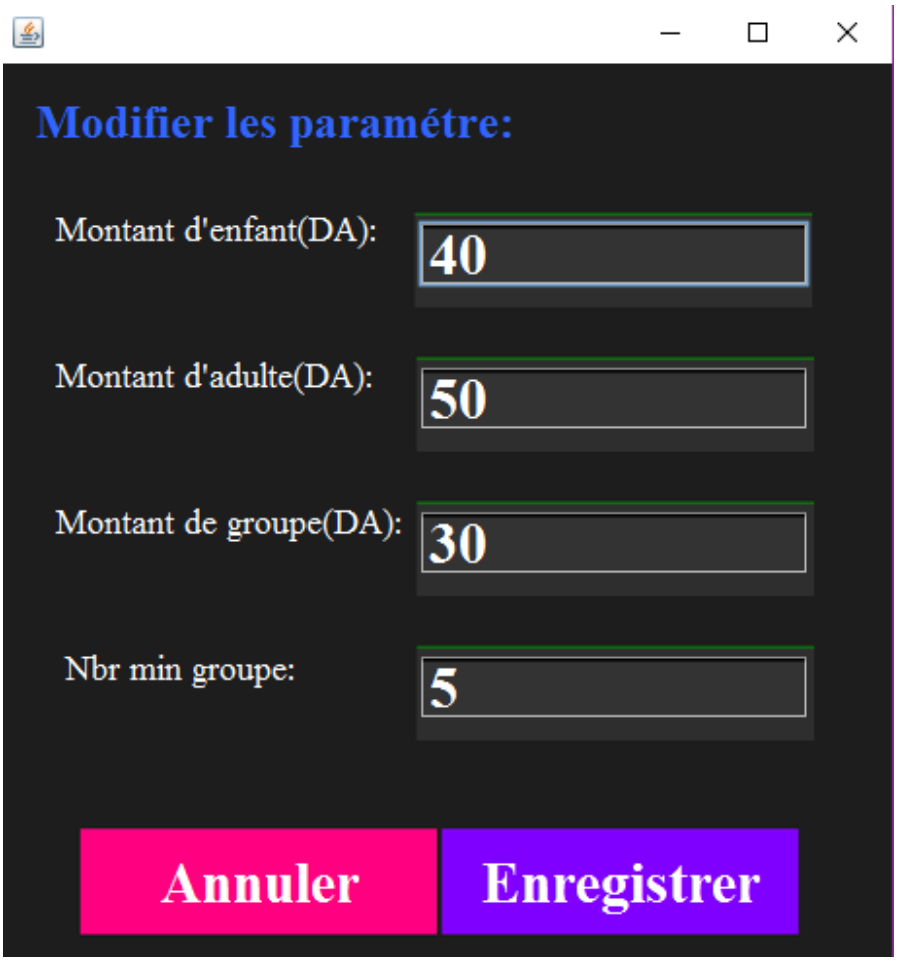

Figure3.4: Le plateau de parameter

2.4.3 Exemple de programme :

Nous voulons calculer le prix deux enfants et deux adultes, Après la fixation des prix de l'utilisateur peut voir et modification du prix.

Montante d'enfant = 40DA.

Montante d'adulte = 50DA.

Montante de groupe = 30DA.

Nous définissons le nombre minimum pour un groupe**.**

Numéro min group = 5.

Clique sur bouton numéro 2 ensuite clique sur bouton enfants, le prix est calculé le résultat est affiché en haut.

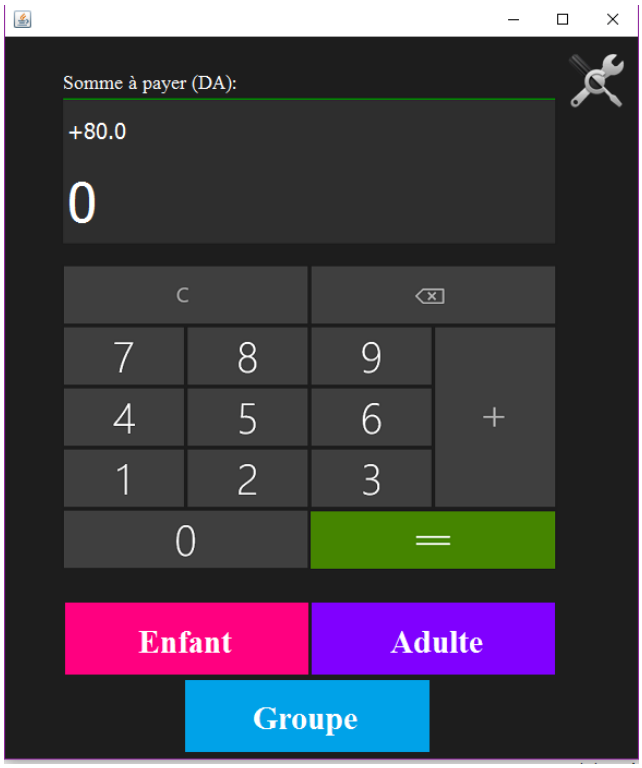

Figure 3.5 : Exemple de programme.

Ensuite en cliquant sur bouton numéro 2 ensuite sur bouton adulte, le prix est calculé, le résultat est réunis directement avec le premier résultat affiché en haut.

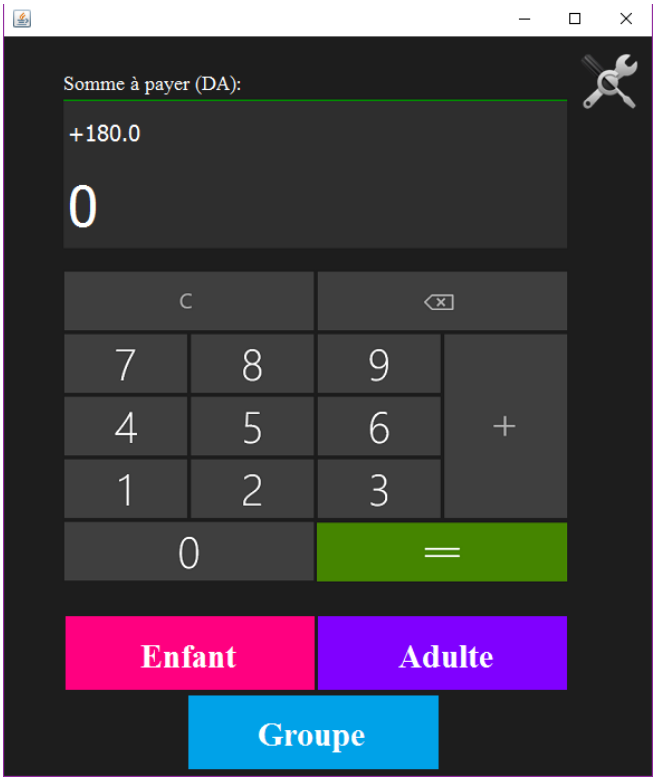

Figure 3.6 : Exemple de programme.

Le résultat de calculer le prix deux enfants et deux adultes est 180 DA.

Nous voulons calculer le prix de cinq personnes en groupe, en cliquant sur le bouton numéro 5 ensuite sur le bouton groupe, le prix est calculé le résultat est affiché en haut.

Figure 3.7 : Exemple de programme.

2.4.4 Code Source :

- ***Interface Tarifs :***

On a défini ici ce qu'on attend d'un objet de type tarifs.

```
package calculatrice;

public interface Tarifs {
    double getTarif();
    void setTarif(double tarif);
    double calculerMontant(int nbrPlaces);
}
```

Figure 3.8 : Interface Tarife.

On peut maintenant donner une ou plusieurs implémentations de cette interface grâce à l'utilisation du mot clef implements :

❖ ***Class Enfant :***

```
package calculatrice;

public class Enfant implements Tarifs {

    private double tarif;

    public Enfant(double tarif) {
        this.tarif = tarif;
    }

    public double getTarif() {
        return tarif;
    }

    public void setTarif(double tarif) {
        this.tarif = tarif;
    }

    public String toString() {
        return "TarifEnfant{" + "tarif=" + tarif + '}';
    }

    public double calculerMontant(int nbrPlaces) {
        return (nbrPlaces * tarif);
    }
}
```

Figure3.9: class Enfant.

❖ ***Class Adulte :***

```
package calculatrice;

public class Adulte implements Tarifs {

    private double tarif;

    public Adulte(double tarif) {
        this.tarif = tarif;
    }

    public double getTarif() {
        return tarif;
    }

    public void setTarif(double tarif) {
        this.tarif = tarif;
    }

    public String toString() {
        return "TarifAdulte{" + "tarif=" + tarif + '}';
    }

    public double calculerMontant(int nbrPlaces) {
        return (nbrPlaces * tarif);
    }
}
```

Figure3.10: class Adulte.

2.4.5 Conséquence :

- Ces 3 objets peuvent être vus comme des tarifés, c'est ce qu'on appelle le polymorphisme.
- Par ce biais, on introduit une couche d'abstraction dans notre programmation ce qui la rend beaucoup plus flexible.

2.5 Conclusion :

Dans ce chapitre, nous avons présenté la partie réalisation de notre application "calculatrice de tarification".

L'implémentation est réalisée en utilisant l'environnement de développement «NetBeans», en respectant la modélisation du chapitre 2, afin d'adopter une solution qui facilite toute modification futur de l'application.

Conclusion Générale :

Nous sommes parvenus, à réaliser un logiciel simple pour la gestion des guichets « calculatrice de tarification », dans lequel nous avons pu voir l'intérêt de l'utilisation de la programmation orientée objet en terme de souplesse, adaptabilité, polymorphisme, qui sont des critères liés aux modifications et aux évolutions d'un logiciel.

Ce rapport présente tous les étapes de réalisation de l'application ; de la modélisation UML à l'implémentation à travers l'environnement de développement "NetBeans"

Nous pouvons améliorer notre travail par le perspectives suivantes :

- Calculer les prix dans différentes devises.
- Paiement par carte de crédit.

Bibliographie :

[2]	Site Web Wikipédia https://fr.wikipedia.org/wiki/UML_(informatique)
[3]	Laurent AUDIBERT, UML 2.0, Institut Universitaire de Technologie de Villetaneuse – Département Informatique, URL : Adresse du document : http ://www-lipn.univ-paris13.fr/ audibert/pages/enseignement/cours.htm.
[4]	KaziAouelBassim et Rostane Zakaria/Suivie des enseignements du LMD par application informatique, 2007.
[5]	Oracle Corporation and/or its affiliates. Sponsored by oracle
[6]	https://nblocalization.netbeans.org/www/about/platform/index4_fr.html