Daix Théo

Dubrulle Allan

Verhoye Victor

# Rapport de modélisation

## Génie Logiciel

## Projet de modélisation et d'implémentation

## Simulateur pour un distributeur de billet de train

Faculté des Sciences

UMONS Université de Mons

Enseignants : Mr Mens et Mr Hauweele
Année Scolaire 2017-2018

# Table des matières


Introduction ..... 4

Diagramme de cas d'utilisations ..... 4

Description du diagramme ..... 4
Diagramme ..... 4

Description des cas d'utilisation ..... 4

Diagramme de classes ..... 5

Description du diagramme ..... 5
Diagramme ..... 6

Diagrammes de séquences ..... 7

1. Acheter billet ..... 7
Description du diagramme ..... 7
Diagramme ..... 7

2. Acheter abonnement ..... 7
Description du diagramme ..... 7
Diagramme ..... 7

3. Renouveler abonnement ..... 8
Description du diagramme ..... 8
Diagramme ..... 8

4. Acheter pass ..... 8
Description du diagramme ..... 8
Diagramme ..... 9

5. Paiement ..... 9
Description du diagramme ..... 9
Diagramme ..... 10

6. Impression ..... 10
Description du diagramme ..... 10
Diagramme ..... 11

7. Sortie de veille ..... 11
Description du diagramme ..... 11
Diagramme ..... 11

8. Vérifier horaire trains ..... 11
Description du diagramme ..... 11
Diagramme ..... 12

9. Créer/gérer une panne ..... 12
Description du diagramme ..... 12
Diagramme ..... 12

10. Recharger/vider nombre d'impression ..... 13
Description du diagramme ..... 13
Diagramme ..... 13

11. Activer/désactiver composant optionnel ..... 14
Description du diagramme ..... 14
Diagramme ..... 14

12. Recharger/vider caisse ..... 14
Description du diagramme ..... 14
Diagramme ..... 15

Diagramme global d'interaction ..... 15

Description du diagramme ..... 15
Diagramme ..... 15

Diagramme d'état ..... 16

Description du diagramme ..... 16
Diagramme ..... 16

# Introduction :

Voici le rapport de modélisation du projet d'un simulateur pour un distributeur de billets de train, à destination des étudiants en deuxième année de bachelier en math-info. Son but est de montrer et d'expliquer les différents diagrammes servant à décrire le fonctionnement de la future application.
Dans l'introduction, nous aimerions parler de certains choix de conception que nous avons fait :

- Lorsqu'une quelconque panne sera détectée, toutes les fonctionnalités qui sont impactées par ces pannes seront désactivés (cela sera fait grâce aux boutons concernés qui seront indisponible dans la fenêtre de simulation).

- De même, si des composants optionnels ne sont pas activés, ils ne seront pas utilisables. Tout ce qui les concerne sera donc indisponible.

# Diagramme de cas d'utilisation :

Ce diagramme (voir Figure 1) représente non pas un simulateur de distributeur, mais bien un distributeur. Nous avons pris cette décision car il est ainsi plus évident de comprendre les réelles interactions entre les utilisateurs et le système. Etant donné que ce projet est en réalité un simulateur, le technicien et le client sont bien sûr la même personne (dans la suite, j'appellerai cette même personne l'utilisateur). En effet, celle-ci pourra à la fois interagir avec le distributeur, mais pourra aussi gérer elle-même les pannes (par exemple, recharger en encre et en papier le distributeur quand il n'y en aura plus). A la suite de ce diagramme, vous pourrez comprendre exactement les cas d'utilisation à l'aide des descriptions qui sont données. Le système bancaire, lui, sera en réalité (dans la suite du projet) la combinaison entre la base de données et le gestionnaire de base de données (dans le diagramme de classe, vous pouvez le voir apparaitre à travers la classe GestionBaseDeDonnees).

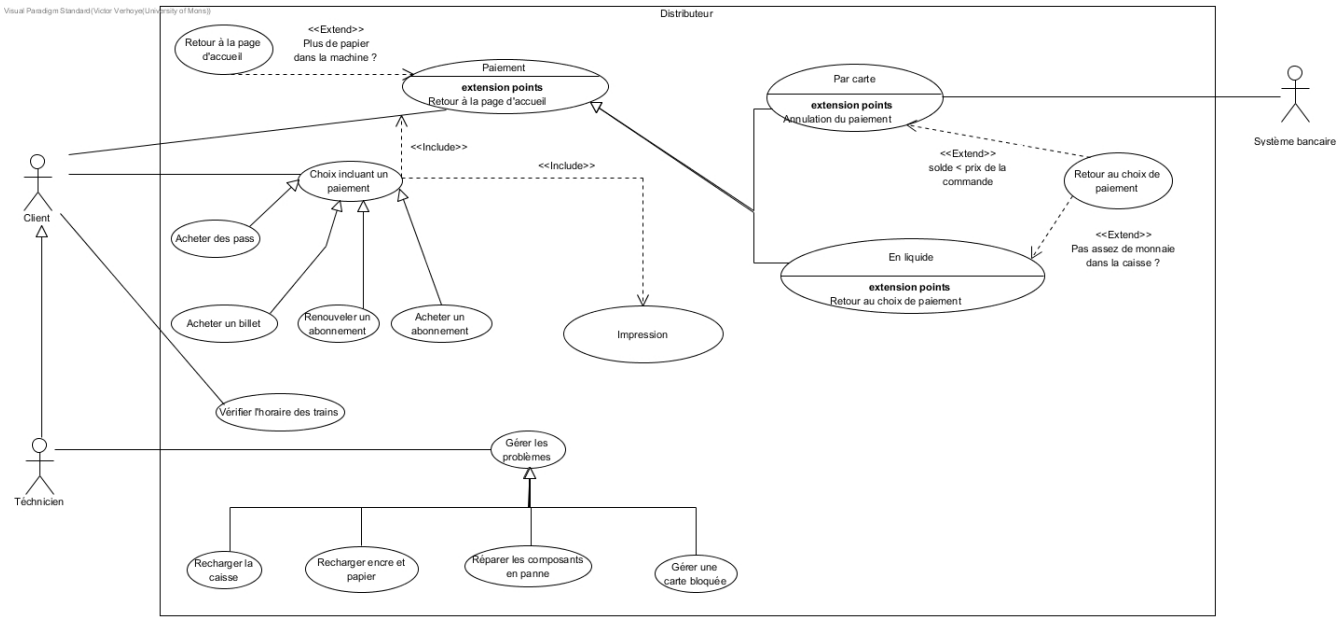


*Figure 1 - Diagramme de cas d'utilisation d'un distributeur de billet de train*

## Diagramme de classes :

Ce diagramme (voir Figure 2) représente les classes principales et leurs associations. Une partie de leur comportement sera explicité dans la suite de ce document à travers les diagrammes de séquences. Nous avons décidé de représenter chaque type de titre de transport à l'aide d'une classe différente (car chaque titre est un objet). La classe abstraite TitreDeTransport qui généralise Billet, Pass et Abonnement sert à éviter la redondance des attributs communs (le montant à payer, les dates de validité et d'expiration, …). Ces titres de transports sont rassemblés avec la classe Recu (la preuve de paiement d'un titre) dans le package Imprimable, car ceux sont tous les objets que l'on peut imprimer.

Un autre package InterfaceGraphique rassemble ce qui sera représenté (voir maquette de l'interface graphique) visuellement lors de l'implémentation. On peut y trouver Reception (l'endroit où on peut récupérer son argent, ainsi que tout ce qui aura été imprimé), LecteurCarte (le lecteur de carte de crédit), Ecran (ce qui affichera les messages, ainsi que les différents menus, l'utilisateur va principalement être en interaction avec celui-ci), et finalement FenetreConfiguration et FenetreSimulation (la fenêtre de configuration va permettre à l'utilisateur de choisir le type de distributeur qu'il souhaite, et la fenêtre de simulation va être le simulateur du distributeur). Il est important de préciser que toutes ces classes se trouvant dans le package InterfaceGraphique vont utiliser les classes de JavaFX (héritage, associations, …). Nous avons décidé de ne pas le représenter pour ne pas alourdir le diagramme.

Dans le dernier package Systeme, nous avons rassemblé toutes les classes qui touchent de près ou de loin à la logique du système. Il est composé de Controleur (comprend la logique principale du système), GestionBaseDeDonnees (qui permettra d'apporter des modifications à la base de données) et HoraireTrains (qui se chargera d'aller chercher les informations sur des horaires de train). Nous pouvons aussi y trouver PaiementLiquide (qui se chargera principalement de vérifier que l'utilisateur donne le bon montant lors d'un paiement liquide), Imprimante (qui se chargera, entre autres, de vérifier qu'il reste suffisamment d'encre et de papier pour imprimer un reçu ou un titre de transport) et Carte (cela représente les cartes de crédit, les données des cartes seront stockées dans la base de données). Finalement, la classe abstraite ComposantPanne est une généralisation des composants (du moins ceux qui sont représentés sous forme de classe) qui peuvent tomber en panne.

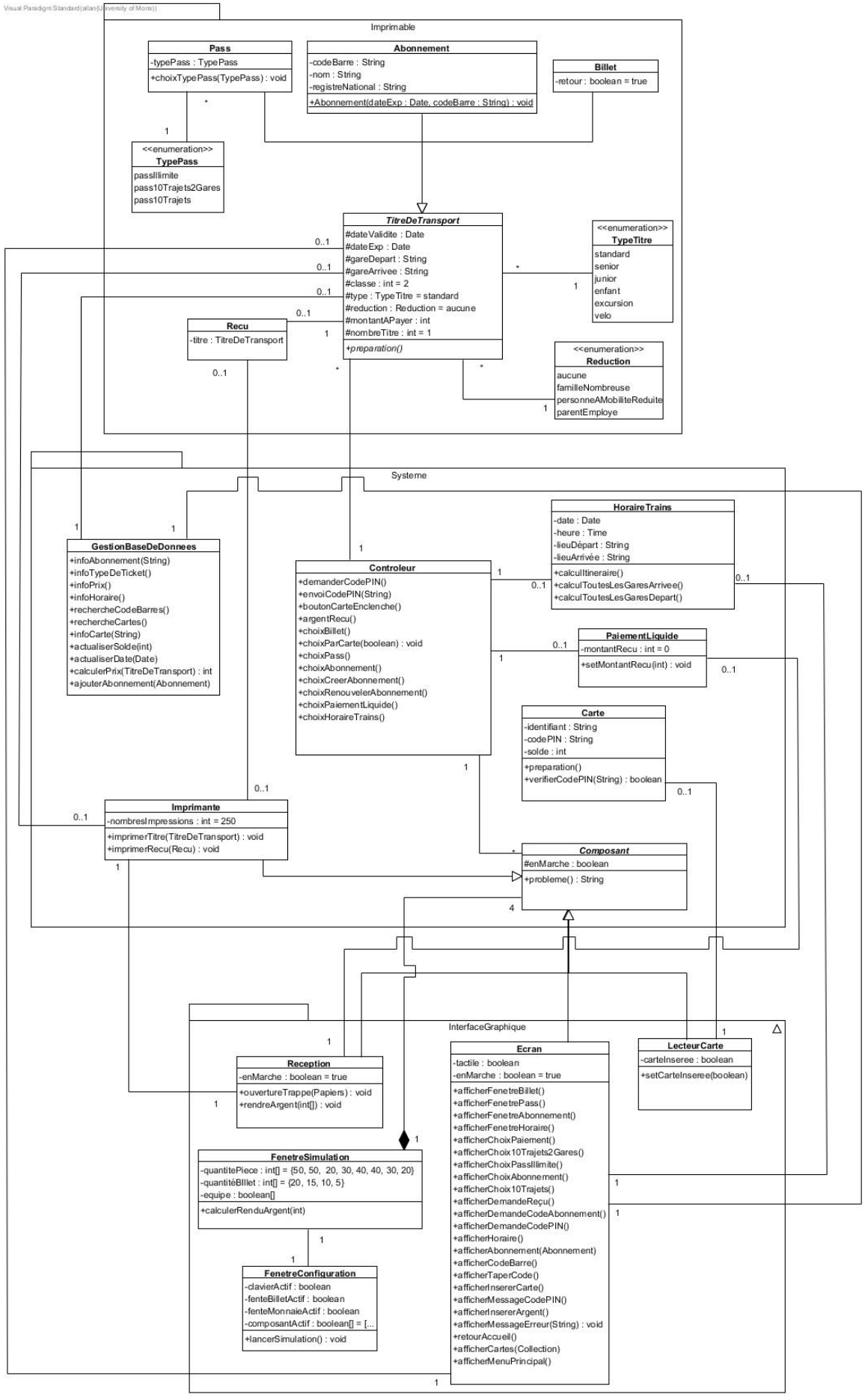


*Figure 2 - Diagramme de classes*

## Diagrammes de séquences :

1. Acheter billet :

Ce diagramme (voir Figure 3) représente le cas où, dans le menu principal, l'utilisateur appuie sur le bouton « Acheter un billet ». S'en suivra un appel d'une méthode qui permettra à Controleur de connaitre le choix de l'utilisateur, afin qu'il crée une instance de Billet. Ecran affichera un nouveau menu où l'utilisateur pourra taper toutes les informations sur le billet qu'il désire. *La suite est généralisée dans les diagrammes Paiement et Impression.*

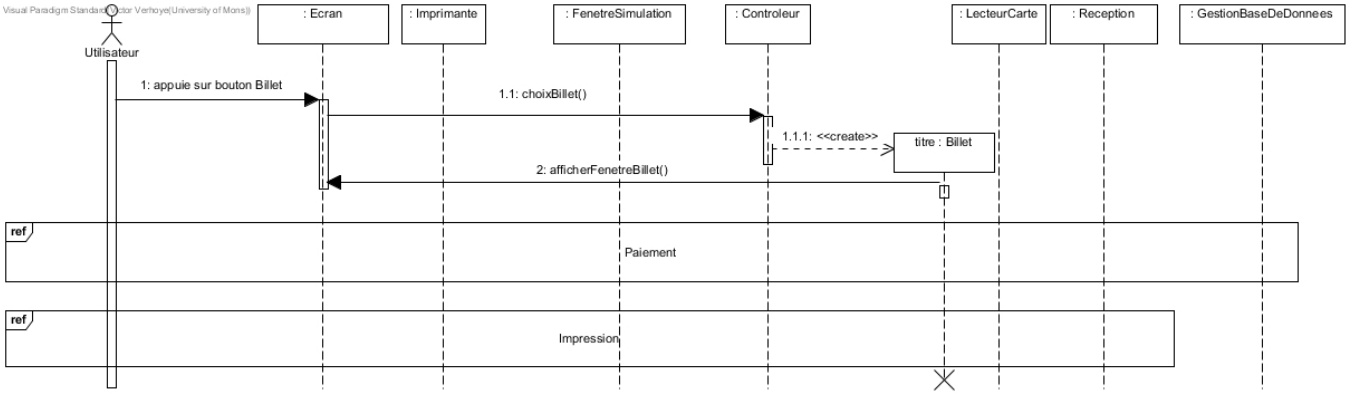


*Figure 3 - Acheter billet*

2. Acheter abonnement :

Ce diagramme (voir Figure 4) représente le cas où, dans le menu principal, l'utilisateur appuie sur le bouton « Acheter un abonnement ». S'en suivra un appel d'une méthode qui permettra à Controleur de connaitre le choix de l'utilisateur, afin qu'il crée une instance de Abonnement. Ecran affichera un nouveau menu où l'utilisateur pourra taper toutes les informations sur l'abonnement qu'il désire. *La suite est généralisée dans les diagrammes Paiement et Impression.* Lorsque toute la procédure est finie, ce nouvel abonnement sera ajouté à la base de données.

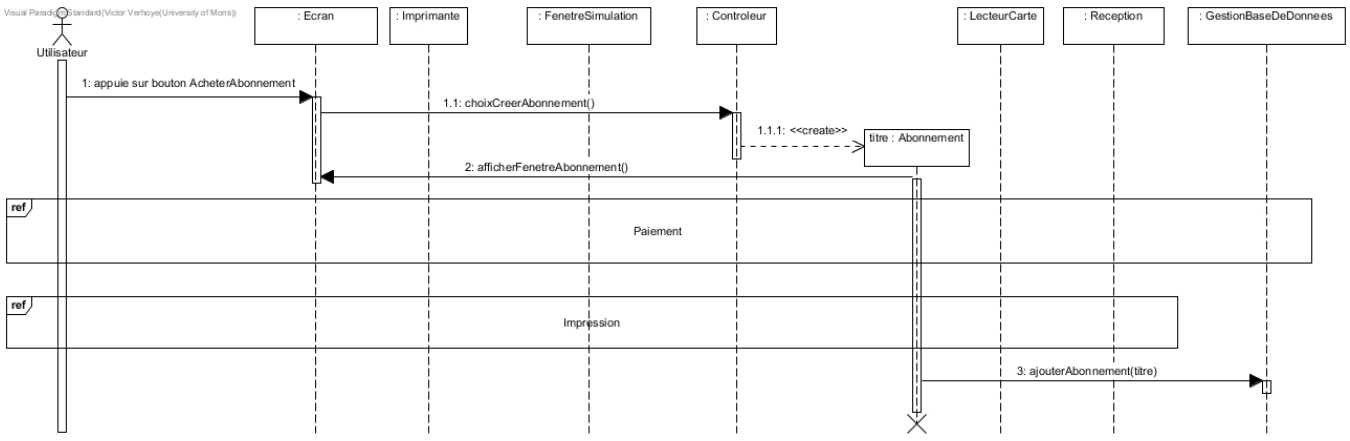


*Figure 4 - Acheter abonnement*

3. Renouveler abonnement :

Ce diagramme (voir Figure 5) représente le cas où, dans le menu principal, l'utilisateur appuie sur le bouton « Renouveler un abonnement ». S'en suivra un appel d'une méthode qui permettra à Controleur de connaitre le choix de l'utilisateur, afin qu'il crée une instance de Abonnement. Ecran affichera un nouveau menu où l'utilisateur choisira s'il veut taper ou scanner le code de l'abonnement. S'il choisit de scanner le code, GestionBaseDeDonnees ira chercher tous les codes des abonnements existants dans la base de données, et Ecran les affichera. L'utilisateur pourra donc choisir un code parmi ceux affichés. S'il a choisi de taper lui-même son code, Ecran affichera une fenêtre où l'utilisateur pourra taper son code. Qu'il ait tapé ou choisi son code, la suite est similaire. GestionBaseDeDonnees va aller chercher les informations de l'abonnement en fonction du code, et Ecran va afficher un nouveau menu qui affichera l'abonnement en question et où l'utilisateur pourra taper les derniers détails du renouvellement. *La suite est généralisée dans les diagrammes Paiement et Impression.* Lorsque toute la procédure est finie, la nouvelle date d'expiration de l'abonnement sera mise à jour dans la base de données.

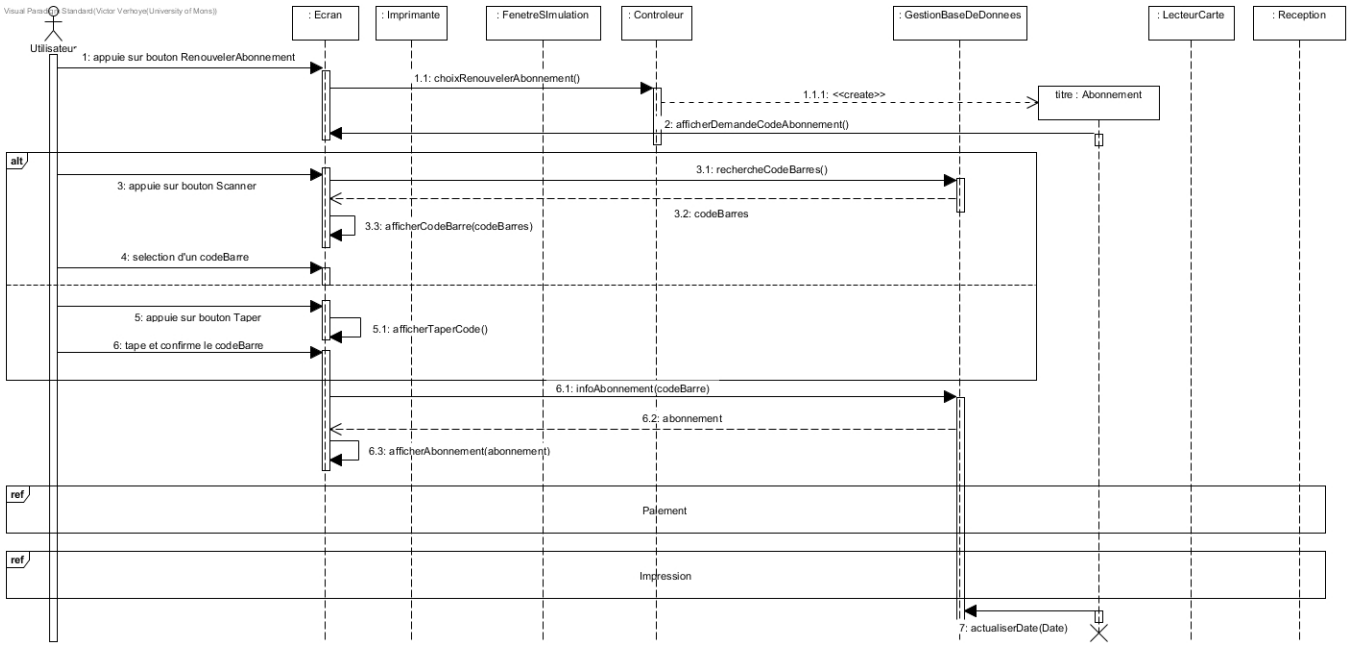


*Figure 5 - Renouveler abonnement*

4. Acheter pass :

Ce diagramme (voir Figure 6) représente le cas où, dans le menu principal, l'utilisateur appuie sur le bouton « Acheter un pass ». S'en suivra un appel d'une méthode qui permettra à Controleur de connaitre le choix de l'utilisateur, afin qu'il crée une instance de Pass. Ecran affichera un nouveau menu où l'utilisateur pourra choisir le type de pass qu'il souhaite (10 trajets, 10 trajets entre 2 gares prédéfinies ou pass illimité). Une méthode donnera l'information à Pass, qui demandera à Ecran d'afficher un nouveau menu, qui permettra à l'utilisateur de taper toutes les informations sur le pass qu'il désire. *La suite est généralisée dans les diagrammes Paiement et Impression.*

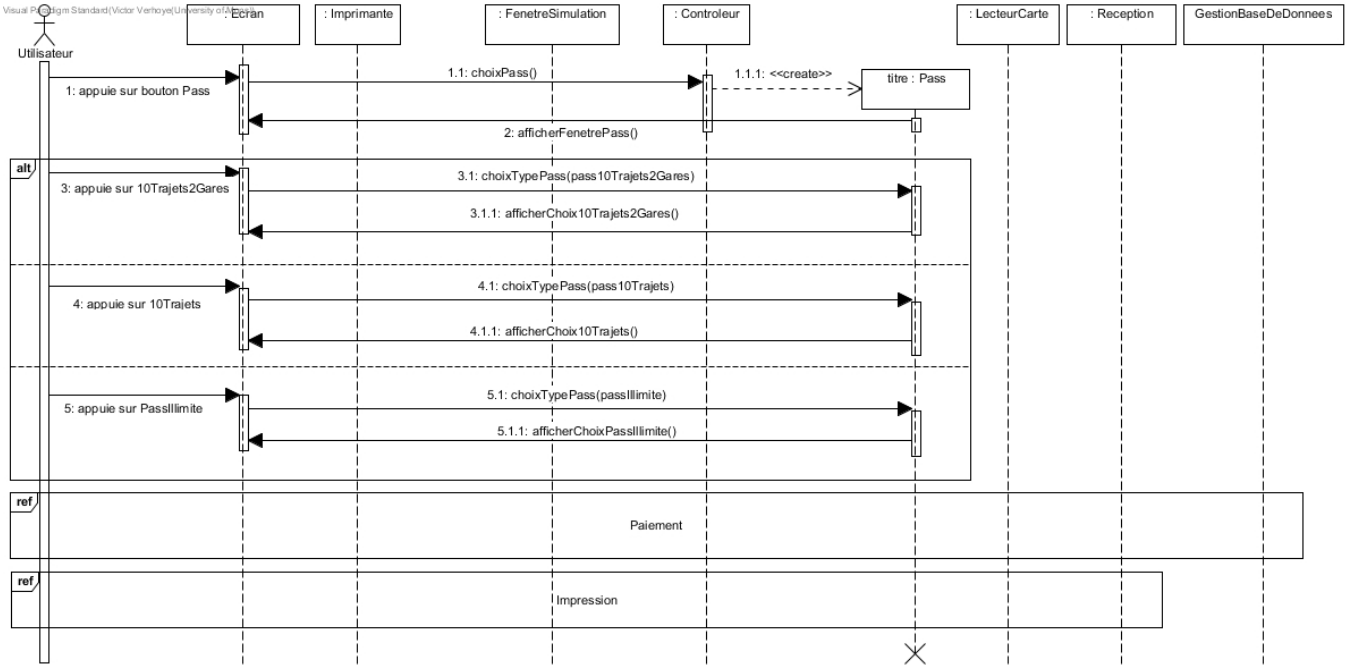


*Figure 6 - Acheter pass*

5. Paiement :

Ce diagramme (voir Figure 7) représente non seulement le paiement d'un titre de transport, mais aussi la fin de la création de ce titre (cette décision a été prise afin d'éviter une quelconque redondance). Il est important de savoir que le paiement ne se fait qu'à la condition que nombresImpressions soit supérieure ou égale à nombreTitre. Ce choix a été pris pour éviter que l'utilisateur paye sans pouvoir recevoir ses billets. Lorsque l'utilisateur aura tapé toutes les informations concernant son titre de transport, il va confirmer. A ce moment-là, la méthode preparation() va aller attribuer à chaque variable de titre de transport une des données tapées par l'utilisateur auparavant. GestionBaseDeDonnees va, lui, calculer le prix de ce titre de transport à l'aide de la base de données (qui stockera pareillement les réductions, les types de titres de transport, …), et ensuite attribuer cette valeur en tant que montantAPayer du titre. Ecran va alors afficher un nouveau menu où l'utilisateur pourra choisir son type de paiement. S'il choisit par carte, Controleur va créer une instance de Carte, et Ecran va demander à l'utilisateur d'insérer sa carte. Lorsque l'utilisateur appuie sur « Insérer carte », GestionBaseDeDonnees va faire une recherche dans la base de données de toutes les cartes stockées, et Ecran va les afficher. Lorsque l'utilisateur va choisir une carte, LecteurCarte va passer son attribut carteInseree à vrai, et les informations sur la carte (code PIN, …) vont être ajouté à l'instance de Carte qui a été créée auparavant. Ecran va alors demander à l'utilisateur son code PIN, et tant que ça ne sera pas le même que celui de l'instance de Carte, il devra recommencer. S'il tape le code correctement, ça modifiera le solde sur sa carte et dans la base de données. S'il décide de payer en liquide, une instance de PaiementLiquide sera créée, qui aura comme attribut le prix du titre de transport. L'utilisateur va alors pour insérer des pièces ou des billets comme bon lui semble. Quand le montant reçu excède le montant à payer, l'argent donnée en trop sera rendue à travers Reception. Nous portons votre attention sur notre choix de « créer une instance de la classe abstraite TitreDeTransport ». Nous sommes bien conscients que cela n'a pas de sens dans la programmation même, mais cette décision a été prise afin de pouvoir généraliser le comportement de Paiement pour un Abonnement, un Billet, ou un Pass.

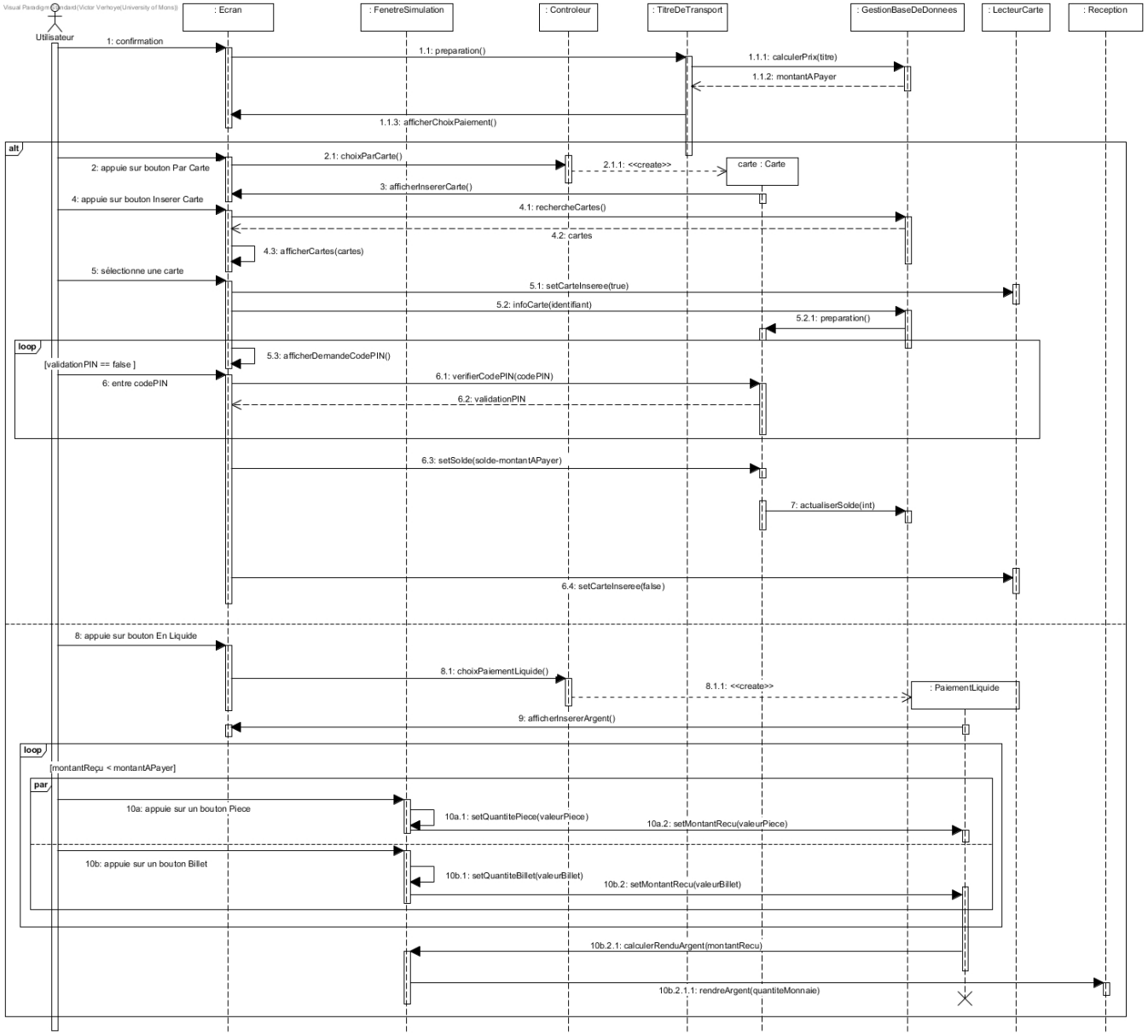


*Figure 7 - Paiement*

6. Impression :

Ce diagramme (voir Figure 8) représente le fait d'imprimer le titre de transport que l'utilisateur vient de payer. Ce dernier peut, s'il le désire, imprimer un reçu. A chaque impression, l'attribut nombresImpressions est décrémenté. Il est important de rappeler qu'il n'y a pas de soucis pour imprimer les titres car le nombre d'impressions a été vérifié au préalable (voir la description du diagramme de séquences Paiement). Cependant, pour le reçu, il se peut que nombresImpressions soit égal à 0, à ce moment-là un message d'erreur est affiché et on revient à la page d'accueil.

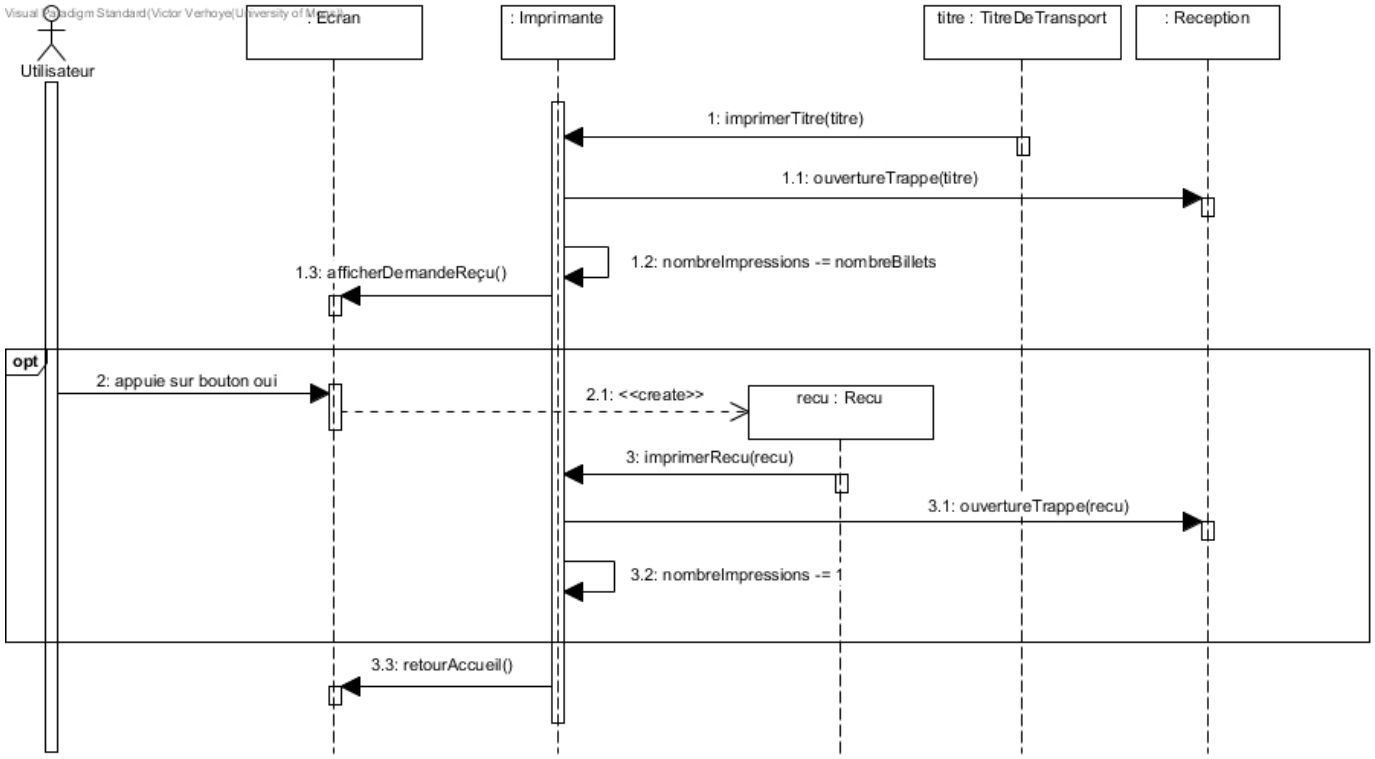


*Figure 8 – Impression*

7. Sortie de veille :

Ce diagramme (voir Figure 9) représente tout simplement le fait que lorsque l'utilisateur va appuyer sur le bouton « Démarrer », Ecran va afficher le menu principal (voir maquette de l'interface graphique).

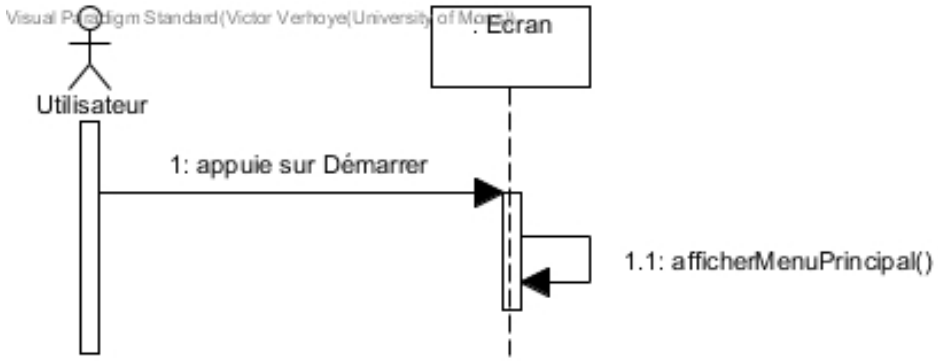


*Figure 9 - Sortie de veille*

8. Vérifier l'horaire d'un train :

Ce diagramme (voir figure 10) représente le choix de l'utilisateur de vérifier l'horaire d'un train. Ecran va afficher une fenêtre où l'utilisateur pourra taper les informations sur le trajet qu'il désire. En fonction des données entrées, HoraireTrains va calculer les trajets, et Ecran va les afficher.

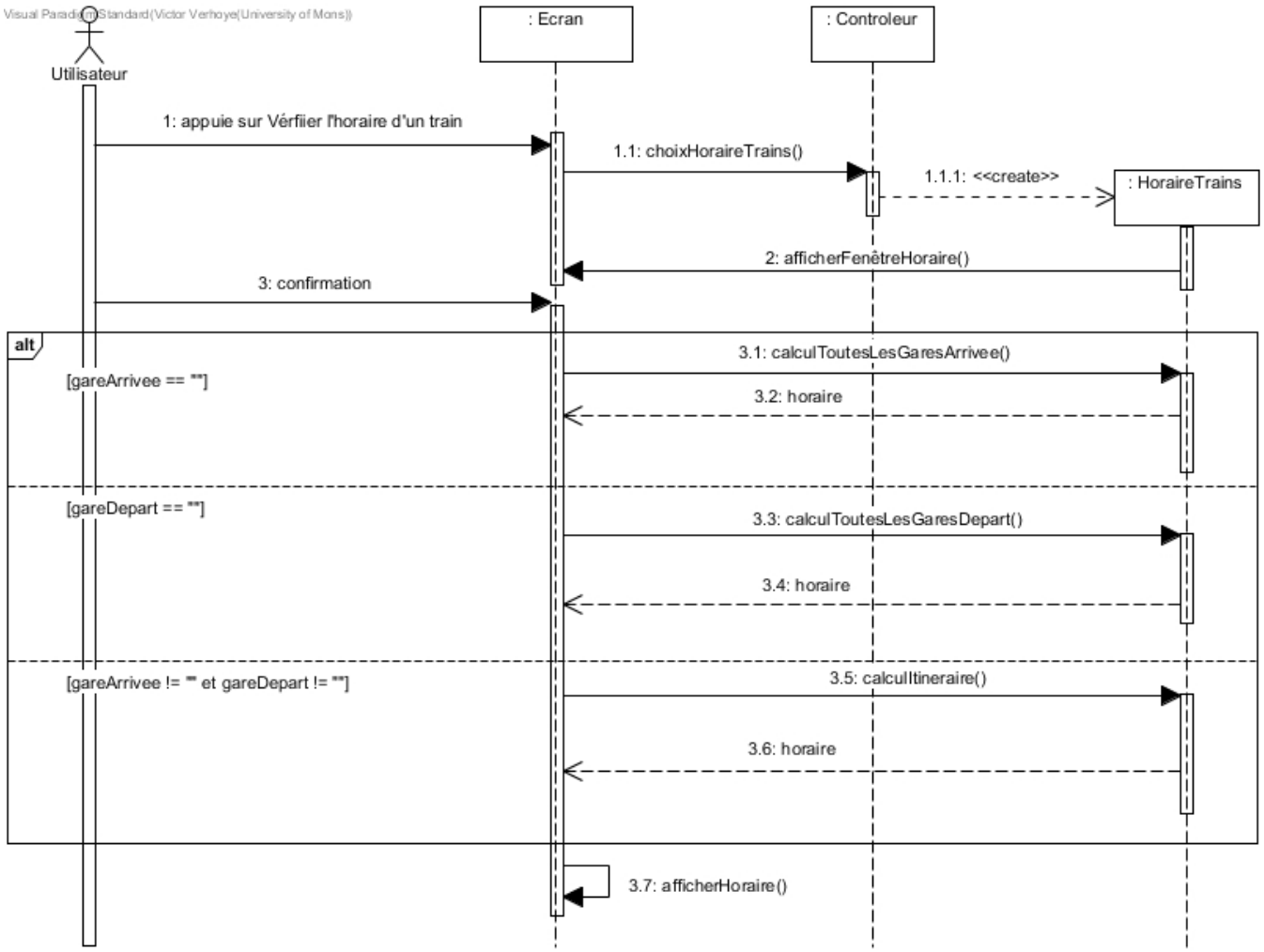


*Figure 10 - Vérifier l'horaire d'un train*

9. Créer/gérer une panne :

Ces diagrammes (voir figures 11 et 12) passent l'attribut enMarche d'un composant à false/true. Cet attribut est la représentation du fonctionnement ou du dysfonctionnement du composant en question. Nous ne représentons pas ici les mises en panne des composants qui ne sont pas représentés sous forme de classe.

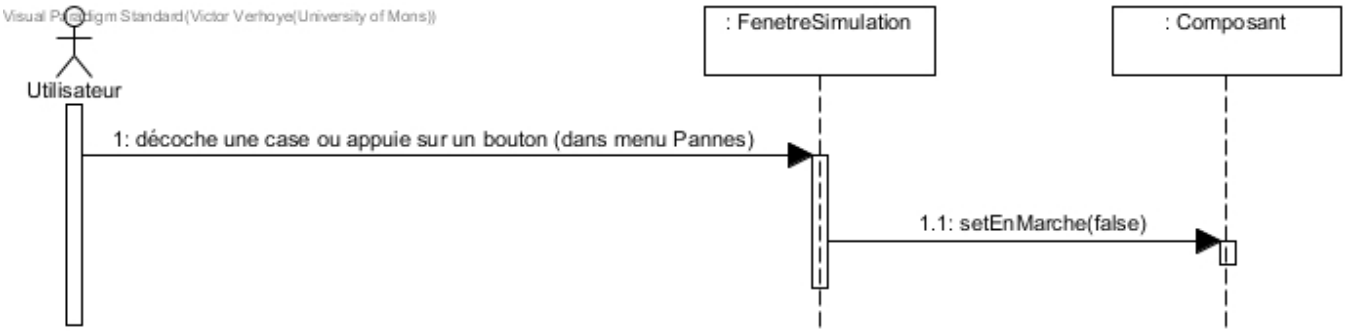


*Figure 11 - Créer une panne*

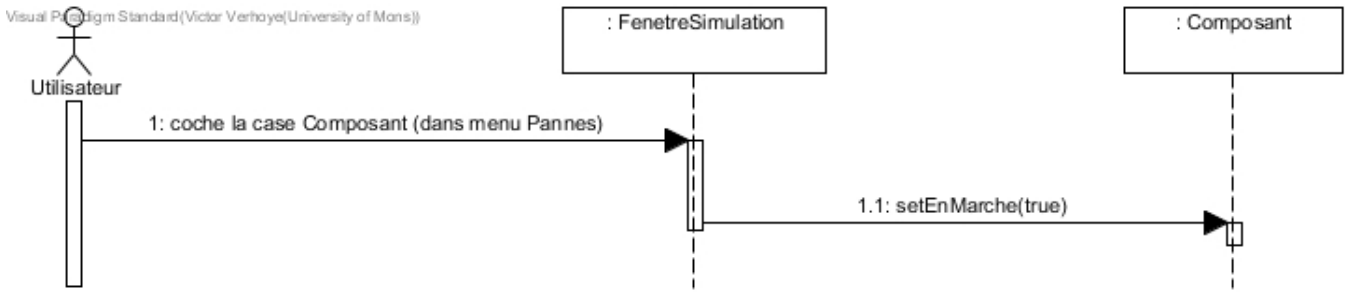


*Figure 12 - Gérer une panne*

10. Recharger/vider nombre d'impressions :

Ces diagrammes (voir figures 13 et 14) représentent le technicien qui remet de l'encre et du papier/qui vide l'encre et le papier. Ici l'attribut nombresImpressions correspond à la capacité du distributeur d'imprimer des tickets ou des reçus car un distributeur a une capacité limitée en encre et en papier. Ici, nombreImpressions est cette limite (vider nombre d'impressions est surtout là pour permettre à l'utilisateur de créer une panne).

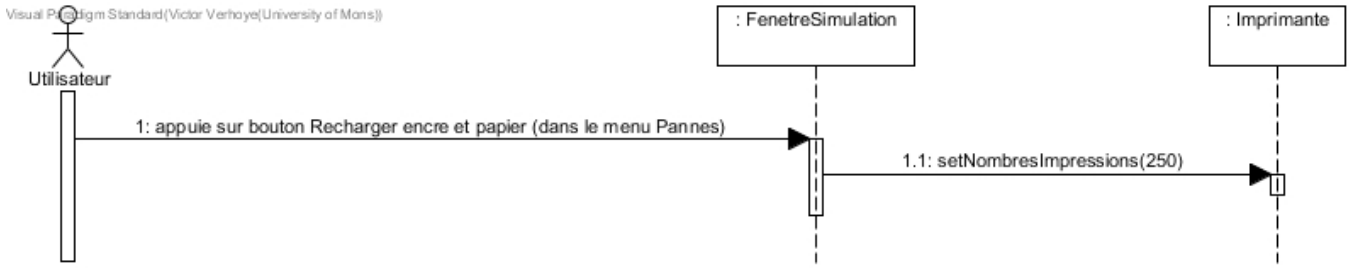


*Figure 13 - Recharger nombre d'impressions*

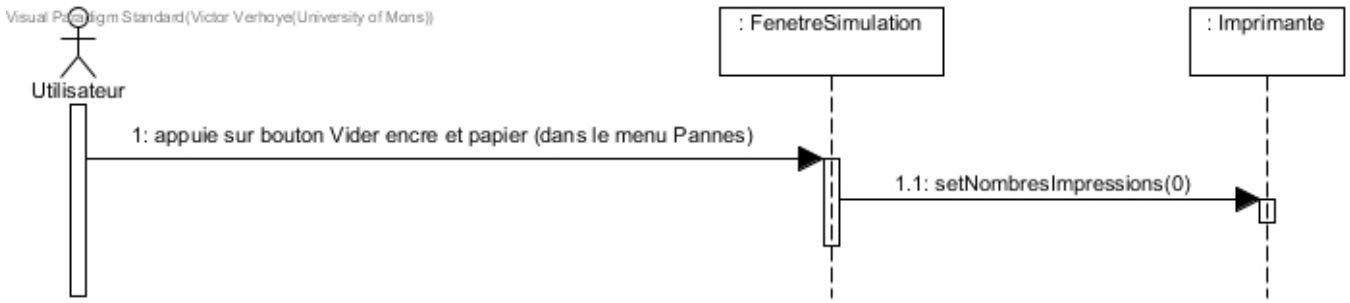


*Figure 14 - Vider nombre d'impressions*

11. Activer/désactiver composant optionnel :

Ces diagrammes (voir figures 15 et 16) représentent le choix de l'utilisateur de modifier FenetreSimulation quand il le désire. S'il coche/décoche une case dans le menu Composants optionnels, ça modifie l'attribut equipe dans FenetreSimulation, en passant à true/false à l'indice correspondant au composant en question.

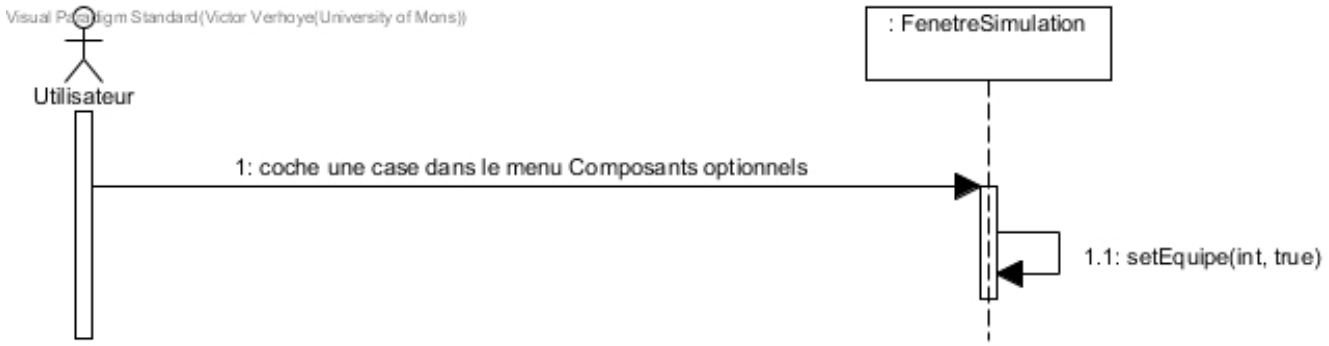


*Figure 15 - Activer composant optionnel*

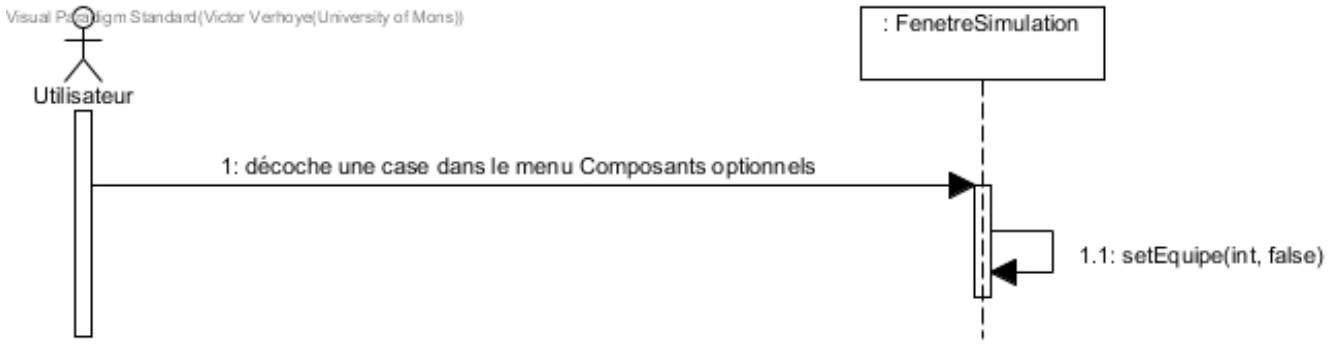


*Figure 16 - Désactiver composant optionnel*

12. Recharger/vider caisse :

Ces diagrammes (voir figures 17 et 18) représentent le technicien qui recharge/vide la caisse. Ici, les pièces et les billets sont représentés par deux tableaux où chaque élément du tableau représente la quantité restante d'une certaine pièce ou d'un certain billet. Ils sont triés par ordre croissant, donc le premier élément de pièce représente le nombre de pièces de 1 cent et le premier élément de monnaie représente le nombre de billets de 5 euros. Donc Recharger caisse consiste à actualiser le nombre de pièces et de billets que la machine possède, et Vider caisse représente quant à lui le fait de vider la caisse, on n'a donc plus de billets et de monnaies ; d'où le fait que tous les éléments du tableau passent à 0.

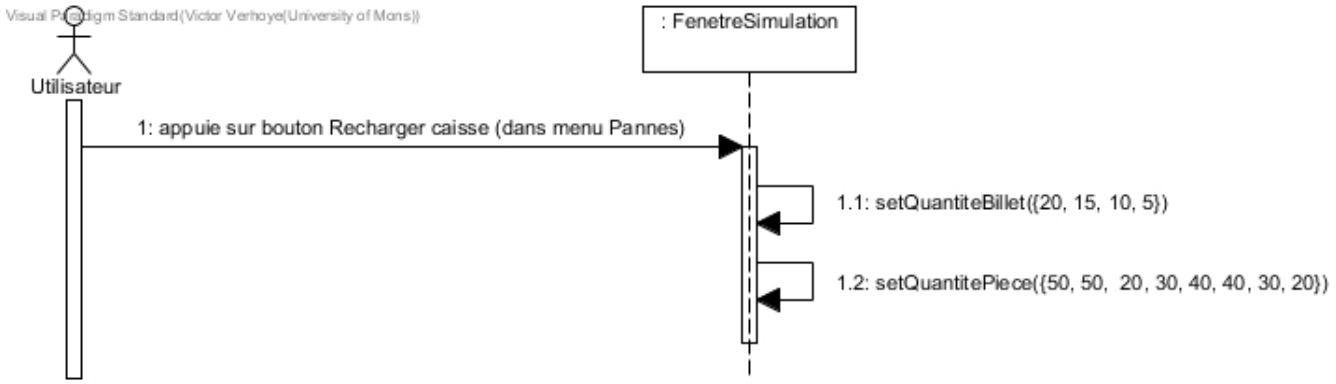


*Figure 17 - Recharger caisse*

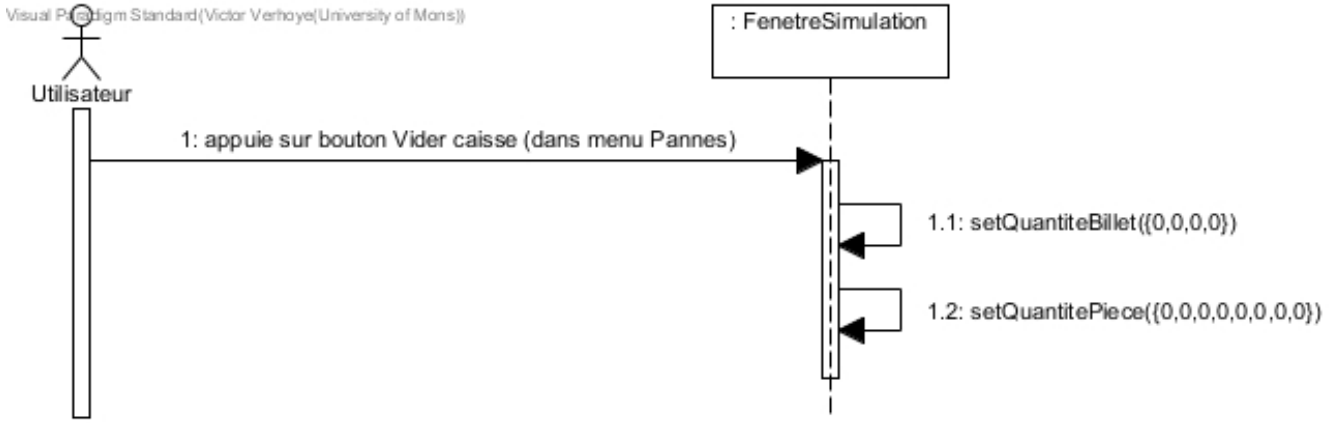


*Figure 18 - Vider caisse*

## Diagramme d'interaction globale (interaction overview diagram) :

Ce diagramme (voir figure 19) représente le fonctionnement typique du distributeur par un utilisateur. Il active la machine, choisit une fonctionnalité proposée par le distributeur, et une fois cette utilisation de l'appareil finie, le distributeur revient à la page d'accueil. Le diagramme comporte une répétition de paiement et d'impression car un achat comprend un paiement et une impression. Le but de cette répétition était d'insister sur l'importance que dans une utilisation normale, on a toujours un paiement et une impression.

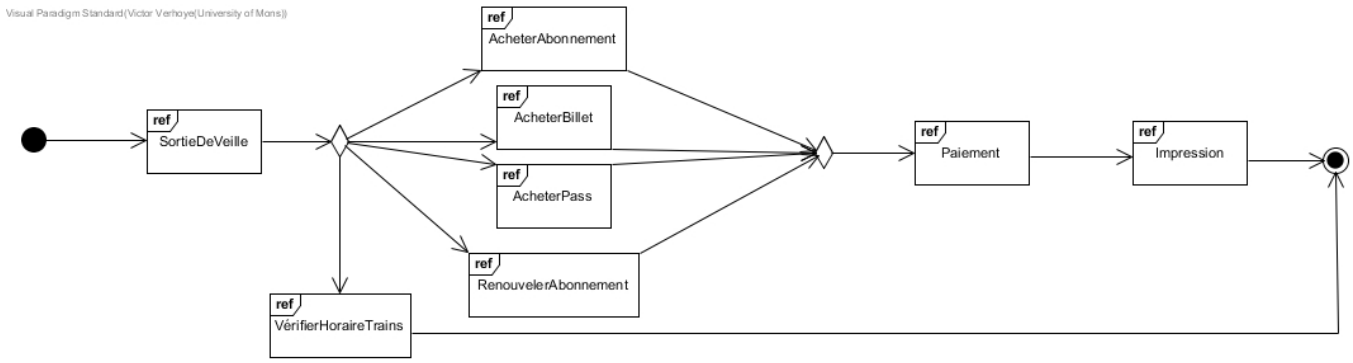


*Figure 19 - Diagramme d'interaction globale*