

# **Projet LAGAN**

Entreprise Dechou & Co

## **Développement d'un logiciel de pilotage d'une batterie d'ascenseurs**

**Dossier de Conception**

**Date de dernière mise à jour : 09/01/2011**

**Version du document : 2.0**

**Version de l'application : 1.0**

**Objet du document :**

**Ce document rassemble les critères et les orientations ayant servi à la conception de l'application. Sa lecture est importante pour tout nouveau développeur, il fournit le cadre conceptuel indispensable à toute modification de l'application.**

	Nom	Fonction	Date	Visa
Auteurs	BOUQUIN Nicolas	Responsable fonctionnel	06/01/2010	NB
	BRENET Victor	Chef de projet	06/01/2010	VB
Vérificateurs	GUILLAUMOND Bertrand	Responsable technique	08/01/2011	BG
	SARRA Fabrice	Responsable qualité	08/01/2011	FS
Approbateurs	BRENET Victor	Chef de projet	09/01/2011	VB

**Destinataires :**

<b>Nom</b>	<b>Société</b>	<b>Date et signature</b>
HEURTEL Philippe	CPE Lyon	
THEVENON Jérôme	CPE Lyon	
TROUILLOT Xavier	CPE Lyon	

**Table des mises à jour du document :**

<b>Version de l'application</b>	<b>version du document</b>	<b>Date</b>	<b>Objet de la mise à jour</b>
1.0	1.0	17/11/10	Création du document
1.0	1.1	05/12/10	Mise à jour du document
1.0	1.2	03/01/2011	Ajout rappel spécifications, ajouts multiples
1.0	1.3	05/01/2011	Améliorations use case modèle
1.0	2.0	09/01/2011	Refonte des diagrammes de séquence, du schéma des use cases, description du diagramme de classes, clarification des schémas, relecture.

# I. SOMMAIRE

<b>II. INTRODUCTION .....</b>	<b>5</b>
1. OBJET DU DOCUMENT .....	5
2. DIFFUSION .....	5
<b>III. RAPPEL DES SPECIFICATIONS FONCTIONNELLES .....</b>	<b>6</b>
1. OBJECTIFS .....	6
2. BESOINS DE FONCTIONNALITES DU LOGICIEL .....	6
a) <i>Environnement logiciel et matériel nécessaire</i> .....	6
b) <i>Fonctionnalités mises en place</i> .....	6
<b>IV. INTERFACES UTILISATEUR .....</b>	<b>9</b>
1. CHOIX DU MODE DE FONCTIONNEMENT .....	9
2. DEMARRAGE DU MODE ALEATOIRE .....	10
3. DEMARRAGE DU MODE SAISIE D'ÉVÉNEMENTS .....	11
4. MODULE DE STATISTIQUES .....	12
5. PANNEAU DE SUIVI DES ASCENSEURS (MODE ALEATOIRE ET SAISIE) .....	13
<b>V. USE CASE MODEL .....</b>	<b>14</b>
<b>VI. USE CASES DÉTAILLÉS .....</b>	<b>15</b>
1. USE CASES ID .....	15
2. USE CASES DÉTAILLÉS .....	16
<b>VII. DIAGRAMME DE RESPONSABILITE .....</b>	<b>19</b>
<b>VIII. DIAGRAMME DE CLASSE .....</b>	<b>20</b>
1. DIAGRAMME DE CLASSE PRELIMINAIRE .....	20
2. DIAGRAMME DE CLASSE DÉTAILLÉ .....	21
3. DESCRIPTION DES PRINCIPAUX ÉLÉMENTS DU DIAGRAMME DE CLASSE .....	22
<b>IX. DIAGRAMMES DE SEQUENCE .....</b>	<b>25</b>
1. DEMARRAGE DE LA SIMULATION .....	25
2. RAFFRAICHISSEMENT GLOBAL DE L'APPLICATION .....	25
3. APPEL D'UN ASCENSEUR .....	26
4. ENVOI D'UN ASCENSEUR A UN ÉTAGE .....	27
5. DEBARQUEMENT D'UN UTILISATEUR .....	28
6. ENVOI D'UN ASCENSEUR A UN ÉTAGE .....	28
7. CHANGEMENT DU MODE DE FONCTIONNEMENT .....	29

## **II. INTRODUCTION**

### **1. Objet du document**

Ce document rassemble les critères et les orientations ayant servi à la conception de l'application. Sa lecture est importante pour tout nouveau développeur, il fournit le cadre conceptuel indispensable à toute modification de l'application. Le but est d'y articuler très clairement la structure générale du code source dans un premier temps, puis le cœur de l'application permettant la mise en œuvre. Nous ne décrivons pas ici la partie interface graphique.

### **2. Diffusion**

Ce document a été approuvé par le chef de projet. Il est donc à disposition des membres de la maîtrise d'œuvre et de la hiérarchie pour l'entreprise Dechou et sera mis à disposition de la maîtrise d'ouvrage dès le 11 Janvier 2011.

### **III. RAPPEL DES SPECIFICATIONS FONCTIONNELLES**

Les spécifications sont détaillées dans le document "Cahier de spécifications fonctionnelles et techniques détaillés". En voici un bref rappel :

#### **1. Objectifs**

Ce logiciel est destiné au pilotage d'une batterie d'ascenseurs. Les buts de cette application sont les suivants :

- Gérer les déplacements des ascenseurs dans le but de desservir tous les étages et de satisfaire les demandes de tous les clients.
- Optimiser ces déplacements suivant deux critères : le temps de traitement d'une demande (qui comprend le temps d'attente devant l'ascenseur et le temps de trajet jusqu'à la destination) et la consommation électrique (de chaque ascenseur et globale).
- Analyser les différents paramètres pour pouvoir démontrer la pertinence de l'utilisation
- Interfacer le logiciel avec une batterie d'ascenseurs réelle via un boîtier d'interfaçage qui permettra de retranscrire à l'application les appels des utilisateurs.

#### **2. Besoins de fonctionnalités du logiciel**

##### **a) Environnement logiciel et matériel nécessaire**

Le programme utilisera une plateforme Java, ce qui inclut pour les besoins de fonctionnement, un ordinateur ayant une machine virtuelle Java installée (JVM version 1.6.0.18 minimum).

La configuration matérielle requise est celle que l'on peut trouver sur les ordinateurs de vos locaux, CPE Lyon (Intel Core 2 duo, 1Go de mémoire vive, 40 Go d'espace disque, prise USB).

La bibliothèque 'awt' est également nécessaire au bon fonctionnement du programme.

##### **b) Fonctionnalités mises en place**

Le logiciel sera capable d'ordonnancer des ascenseurs selon plusieurs mode de fonctionnement, ainsi on retrouve le mode "jour", "nuit et weekend", "Economie d'énergie" et "Economie de déplacement".

Ces différents mode se traduisent par des d'algorithmes de fonctionnement.

Les différents algorithmes représentant un mode de fonctionnement sont explicables de la sorte :

**i. Algorithme de sélection**

Si un ascenseur est à l'arrêt à l'étage de l'appelant, il est sélectionné.

Sinon,

L'ascenseur le plus proche à l'arrêt est sélectionné.

Sinon

L'ascenseur en déplacement dans le même sens que le trajet de l'appelant, et le plus proche de l'étage est sélectionné s'il n'est pas plein.

Si deux ascenseurs à la même distance (nombre d'étage \* énergie) de l'étage appelant, celui qui a le moins d'arrêts déjà prévus dans sa course est sélectionné.

Sinon

L'ascenseur ayant la destination la plus proche de l'étage appelant et le nombre d'arrêts prévu le plus faible est sélectionné, et son sens de déplacement pourra être changé lorsqu'il aura amené tous les utilisateurs à destination.

**ii. Algorithme de repositionnement standard (rez-de-chaussée)**

Si la liste des destinations et la liste des appelants de l'ascenseur sont vides toutes les deux  
Alors l'ascenseur est repositionné au rez-de-chaussée

**iii. Algorithme de repositionnement par zones**

On définit au départ trois zones : les sociétés indépendantes, la multinationale et l'hôtel. A chaque zone sont affectés deux ascenseurs.

Si la liste des destinations et la liste des appelants de l'ascenseur sont vides toutes les deux  
Alors on vérifie à quelle zone appartient l'ascenseur  
On repositionne l'ascenseur à l'étage correspondant au début de la zone auquel il appartient

#### iv. Algorithme de repositionnement optimisé

En utilisant les statistiques d'utilisations qui nous ont été fournies, on peut déterminer la position adéquate des ascenseurs pour traiter les demandes en un minimum de temps. La position de retour des ascenseurs après avoir exécuté cet algorithme permet de traiter les demandes le plus rapidement possible.

En outre, on définit un ordre de priorité entre ces différentes positions. Ainsi, si un seul ascenseur est libre, il se positionnera à la position ayant la priorité la plus haute.

Si la liste des destinations et la liste des appelants de l'ascenseur sont vides toutes les deux  
 Alors on vérifie combien d'ascenseurs sont libres  
 Alors on se positionne à l'étage ayant la priorité suivante

Les priorités définies sont les suivantes :

Priorité	Jour	Nuit et Week-end
1	rez-de-chaussée (étage 0)	rez-de-chaussée (étage 0)
2	rez-de-chaussée (étage 0)	hôtel (étage 35)
3	sociétés indépendantes (étage 12)	sous-sol (étage -2)
4	hôtel (étage 35)	hôtel (étage 37)
5	sociétés indépendantes (étage 6)	rez-de-chaussée (étage 0)
6	multinationale (étage 19)	hôtel (étage 0)

Ces valeurs ont été définies via les statistiques du cahier des charges. Pour la journée, nous avons cherché à repositionner des ascenseurs dans tout le building en priorisant le rez-de-chaussée. En revanche, la nuit et le week-end, les ascenseurs sont répartis entre l'hôtel et le rez-de-chaussée.



## IV. INTERFACES UTILISATEUR

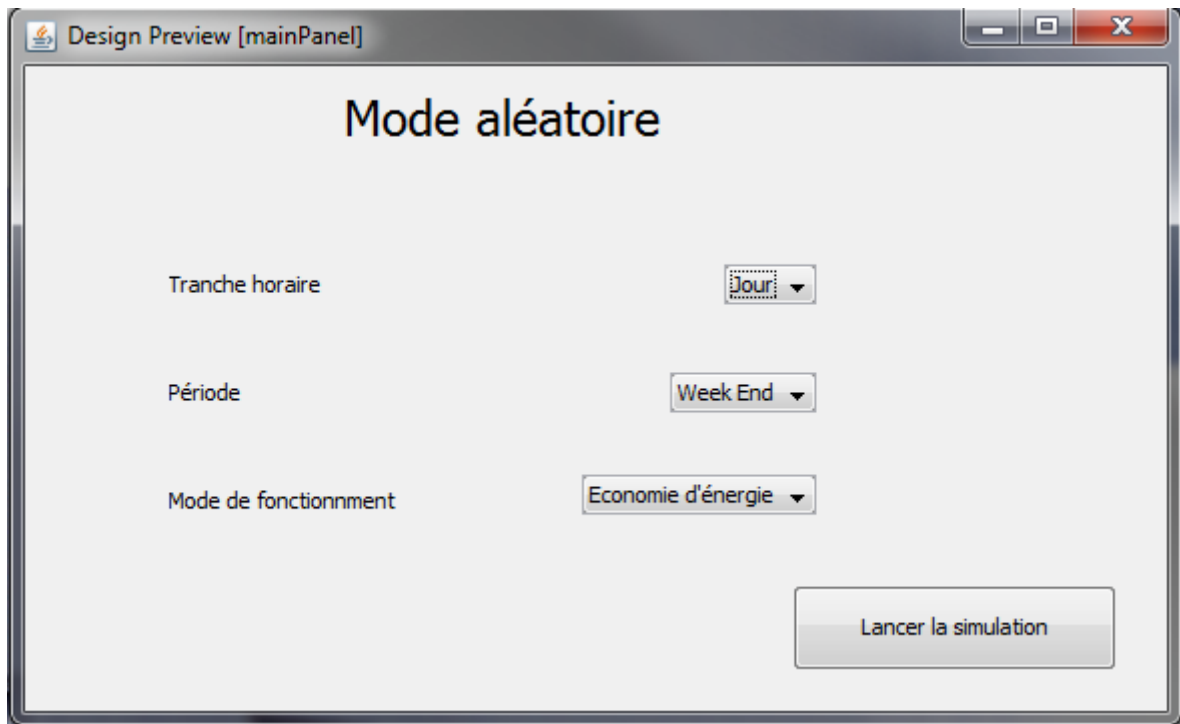
L'interface utilisateur est très simple et complète, l'utilisation du programme se fait par le biais d'affichage d'une fenêtre présentant les différents choix de mode de fonctionnement. On accède ensuite à la fenêtre de paramétrage du mode de fonctionnement choisi. On pourra également retrouver une vue représentant l'immeuble et ses ascenseurs. Enfin, on disposera d'une fenêtre récapitulant les diverses statistiques de l'application.

### 1. Choix du mode de fonctionnement



Cette interface est la première à laquelle on accède lorsque l'on démarre le programme, grâce à celle-ci, on peut choisir de lancer une simulation en mode « aléatoire » ou en mode « saisie d'évènement ».

## 2. Démarrage du mode aléatoire



Si c'est le mode de fonctionnement « aléatoire » qui a été choisi, cette interface sera alors affichée à l'écran. Celle-ci permet d'entrer les trois données clés nécessaires au démarrage de la simulation puisque celle-ci ne se déroulera pas de la même façon suivant la tranche horaire, la période ou encore le mode de fonctionnement sélectionné.

### 3. Démarrage du mode saisie d'évènements

Design Preview [DesktopApplication1View]

File Help

## Mode saisie d'évènements

Modes de fonctionnement

Semaine / Weekend Jour / Nuit Economie d'énergie / Economie de déplacement

Repositionnement optimisé ok

Appel d'un ascenseur

Niveau d'appel: Entrez le niveau Appeler

Destination : Entrez le niveau

Déplacement d'un ascenseur

Ascenseur Ascenseur 1 Etagage Envoyer

Repositionnement ascenseurs

Ascenseur Ascenseur 1 Tous ? Repositionner

Si c'est le mode de fonctionnement « saisie d'évènements » qui a été choisi, cette interface sera alors affichée à l'écran. A la différence de l'interface précédente, lorsque l'on choisit ce mode de fonctionnement, la simulation est démarrée directement, et on peut alors voir instantanément quelle sont les effets de l'appel d'un ascenseur dans les conditions sélectionnées.

Sur cette interface, il n'est pas seulement possible de simuler un appel, mais l'on peut aussi tout simplement replacer un ascenseur à l'étage voulu ou encore repositionner automatiquement tous les ascenseurs de façon optimisée.

#### 4. Module de statistiques

**Statistiques**

Réinitialiser

---

**Consommation électrique**

Ascenseur 1 xxx KWh	Ascenseur 3 xxx KWh	Ascenseur 5 xxx KWh
Ascenseur 2 xxx KWh	Ascenseur 4 xxx KWh	Ascenseur 6 xxx KWh
Globale :   xxx KWh		

---

**Temps de trajet**

Ascenseur 1 xxx min	Ascenseur 3 xxx min	Ascenseur 5 xxx min
Ascenseur 2 xxx min	Ascenseur 4 xxx min	Ascenseur 6 xxx min
Moyen   xxx min		

---

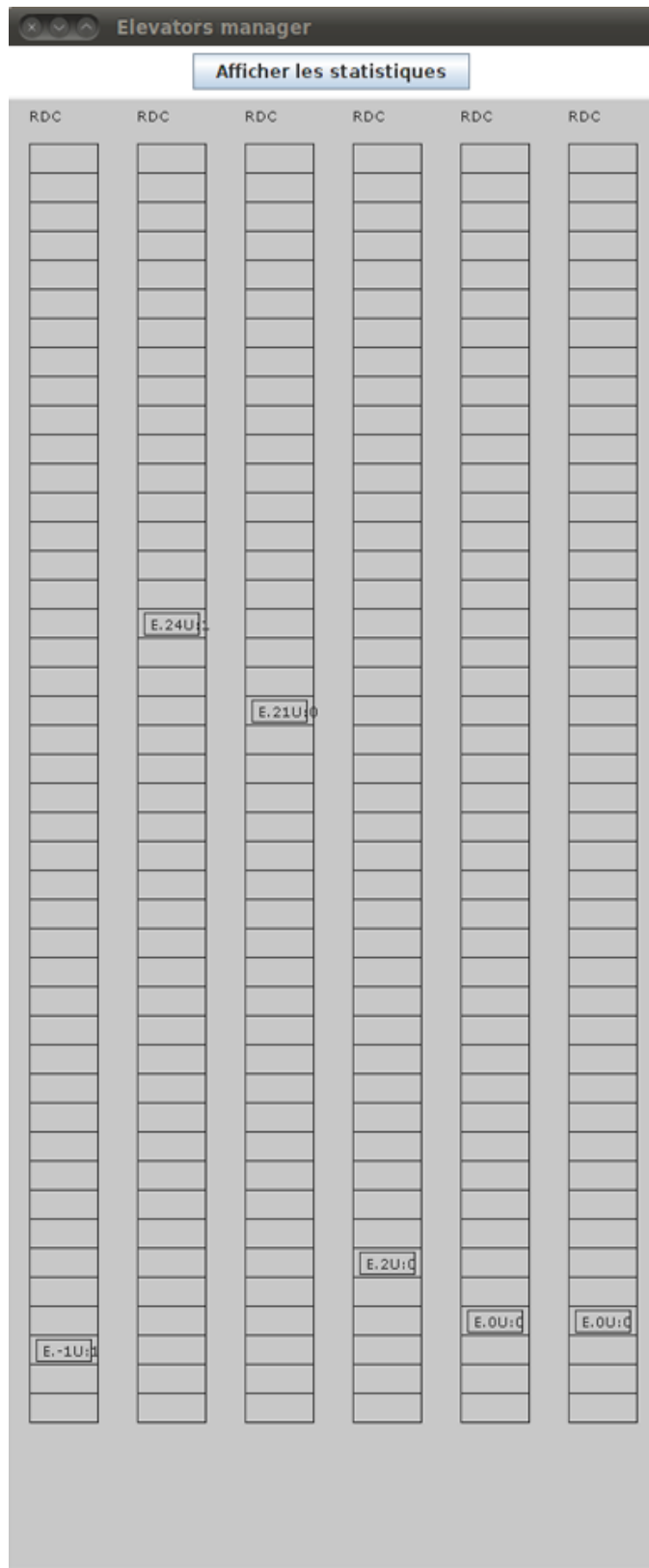
**Temps d'attente**

Ascenseur 1 xxx sec	Ascenseur 3 xxx min	Ascenseur 5 xxx sec
Ascenseur 2 xxx sec	Ascenseur 4 xxx sec	Ascenseur 6 xxx sec
Moyen   xxx sec		

Quel que soit le mode de fonctionnement choisi précédemment, il est possible d'ouvrir cette fenêtre à partir de l'interface représentant les ascenseurs (voir page suivante). Cette interface permet quand à elle d'afficher trois types de statistiques concernant les ascenseurs : la consommation électrique, la durée du trajet et le temps d'attente des clients.

Pour chacun de ces types de mesure, une moyenne sur tous les ascenseurs est calculée. De même, il est possible de remettre à zéro toutes ces valeurs en cliquant sur le bouton « Réinitialiser ».

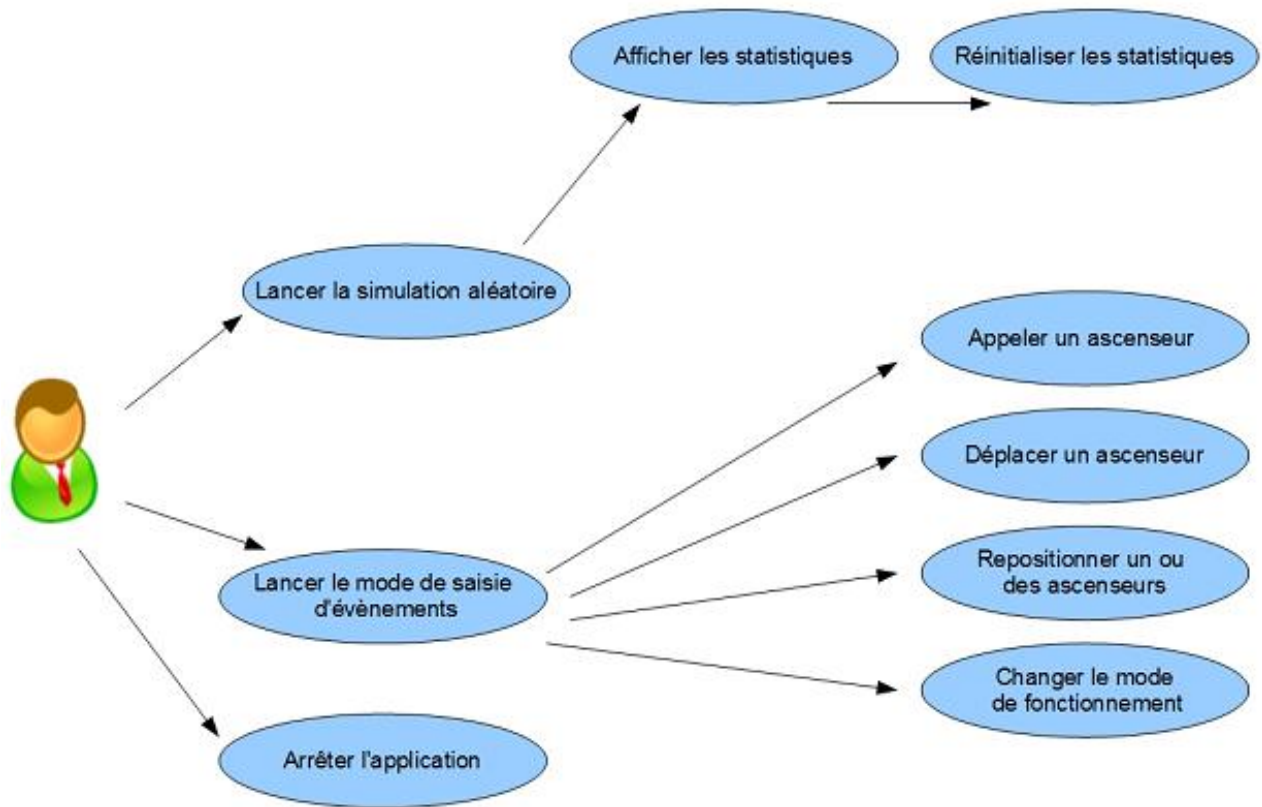
## 5. Panneau de suivi des ascenseurs (Mode aléatoire et saisie)



Cette interface est celle qui sera la plus « parlante » puisqu'elle représente graphiquement les ascenseurs et leurs positions, de plus le déplacement des ascenseurs s'effectue en direct. C'est aussi grâce à cette interface que l'on aura accès au module statistique vu précédemment.

## V. USE CASE MODEL

Le *use case model* montre très brièvement les actions possibles pour l'utilisateur et les différentes étapes de fonctionnement.



## VI. USE CASES DETAILLÉS

Un *use case* permet de mettre en évidence les relations fonctionnelles entre les acteurs et le système étudié. Nous concernant, les use case commençant par « S » correspondent aux use case liés au système, ceux commençant par « L » sont liés au logiciel.

### 1. Use Cases ID

Voici la liste des Uses Case qui vont être détaillés :

Use Case ID	Description
S1	Exécution
S2	Arrêt
L1	Mode simulation
L2	Mode Aléatoire
L3	Mode saisie d'évènements
L4	Configuration des évènements
L5	Changer de mode de fonctionnement
L6	Simulation d'un appel
L7	Positionner un ascenseur
L8	Repositionner les ascenseurs
L9	Affichage des performances

## 2. Use Cases Détaillés

Use case :	Exécution
Use case ID :	S1
Acteurs :	Administrateur
Préconditions :	Le logiciel est sur le post-computer
Evénement(s) :	
	Lance la compilation du programme
Post condition(s) :	
	Le programme a été lancé

Use case :	Arrêt
Use case ID :	S2
Acteurs :	Administrateur
Préconditions :	Le logiciel est exécuté
Evénement(s) :	
	Arrête le logiciel
Post condition(s) :	
	Le logiciel a bien été arrêté

Use case :	Module Simulation
Use case ID :	L1
Acteurs :	Administrateur
Préconditions :	Le logiciel est exécuté
Evénement(s) :	
	L'utilisateur clique sur « Simulation »
Post condition(s) :	
	Le module de simulation est lancé

Use case :	Mode aléatoire
Use case ID :	L2
Acteurs :	Administrateur
Préconditions :	Le module simulation est ouvert
Evénement(s) :	
	L'utilisateur clique sur « mode aléatoire »
Post condition(s) :	
	La simulation en mode aléatoire est lancée



Use case :	Mode saisie d'évènements
Use case ID :	L3
Acteurs :	Administrateur
Préconditions :	Le module simulation est ouvert
Événement(s) :	
	L'utilisateur clique sur « mode saisie d'évènements »
Post condition(s) :	
	La simulation en mode saisie d'évènements est lancée

Use case :	Configuration des événements
Use case ID :	L4
Acteurs :	Administrateur
Préconditions :	Le mode saisie d'évènements est ouvert
Événement(s) :	
	L'utilisateur saisit les événements qu'il souhaite insérer dans la simulation
Post condition(s) :	
	Les événements sont injectés dans la simulation

Use case :	Changer de mode de fonctionnement
Use case ID :	L5
Acteurs :	Administrateur
Préconditions :	Le mode saisie d'évènements est ouvert
Description :	
	Plusieurs modes de fonctionnements sont à disposition. Par défaut,
	les modes sont à : semaine / jour / économie d'énergie
Événement(s) :	
	Changer de période : L'utilisateur clic sur le bouton "jour/nuite"
	Changer de mode : L'utilisateur clic sur le bouton "semaine/week-end"
	Changer le mode énergétique : L'utilisateur clic sur le bouton "Economie d'énergie /Economie de déplacement"
Post condition(s) :	
	La simulation en mode saisie d'évènements est lancée

Use case :	Simulation d'un appel
Use case ID :	L6
Acteurs :	Administrateur
Préconditions :	Le mode saisie d'évènements est ouvert
Événement(s) :	
	L'utilisateur doit saisir un étage d'appel
	L'utilisateur doit saisir un étage de destination
	L'utilisateur doit cliquer sur le bouton "appeler"
Post condition(s) :	
	Les ascenseurs prennent en compte l'appel

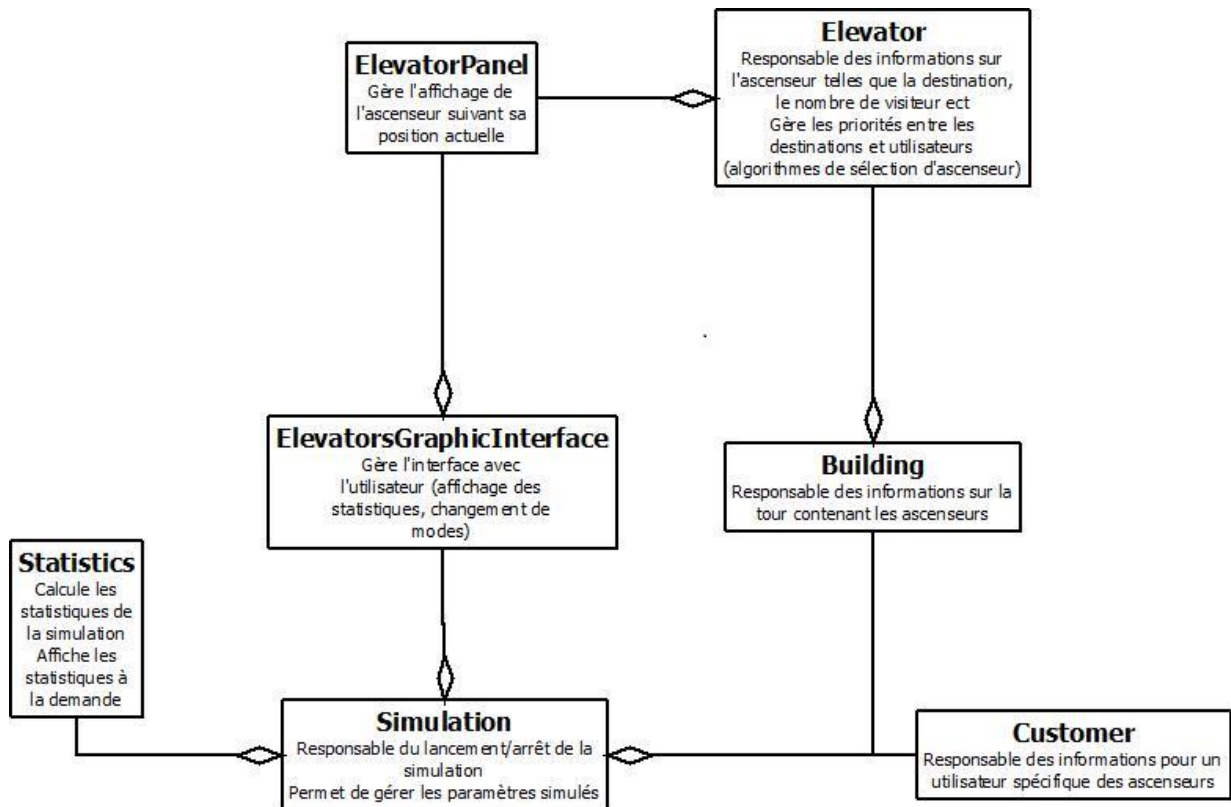
Use case :	Positionner un ascenseur
Use case ID :	L7
Acteurs :	Administrateur
Préconditions :	Le mode saisie d'évènements est ouvert
Événement(s) :	
	L'utilisateur doit choisir un ascenseur
	L'utilisateur doit saisir un étage
	L'utilisateur doit cliquer sur le bouton "envoyer"
Post condition(s) :	
	L'ascenseur saisi par l'utilisateur va se rendre à l'étage saisi

Use case :	Repositionner les ascenseurs
Use case ID :	L8
Acteurs :	Administrateur
Préconditions :	Le mode saisie d'évènements est ouvert
Événement(s) :	
	L'utilisateur choisit l'ascenseur dans la liste déroulante OU
	l'utilisateur coche la case "Tous"
	L'utilisateur doit cliquer sur le bouton "Repositionner"
Post condition(s) :	
	L'ascenseur saisi par l'utilisateur va se rendre à l'étage correspondant au résultat de l'algorithme de repositionnement

Use case :	Affichage des performances
Use case ID :	L9
Acteurs :	Administrateur
Préconditions :	La simulation est démarrée
	La fenêtre de visualisation des déplacements des ascenseurs est ouverte
Événement(s) :	
	L'utilisateur doit cliquer sur le bouton "Statistiques"
Post condition(s) :	
	La fenêtre des statistiques s'ouvre et les statistiques s'affichent

## VII. DIAGRAMME DE RESPONSABILITE

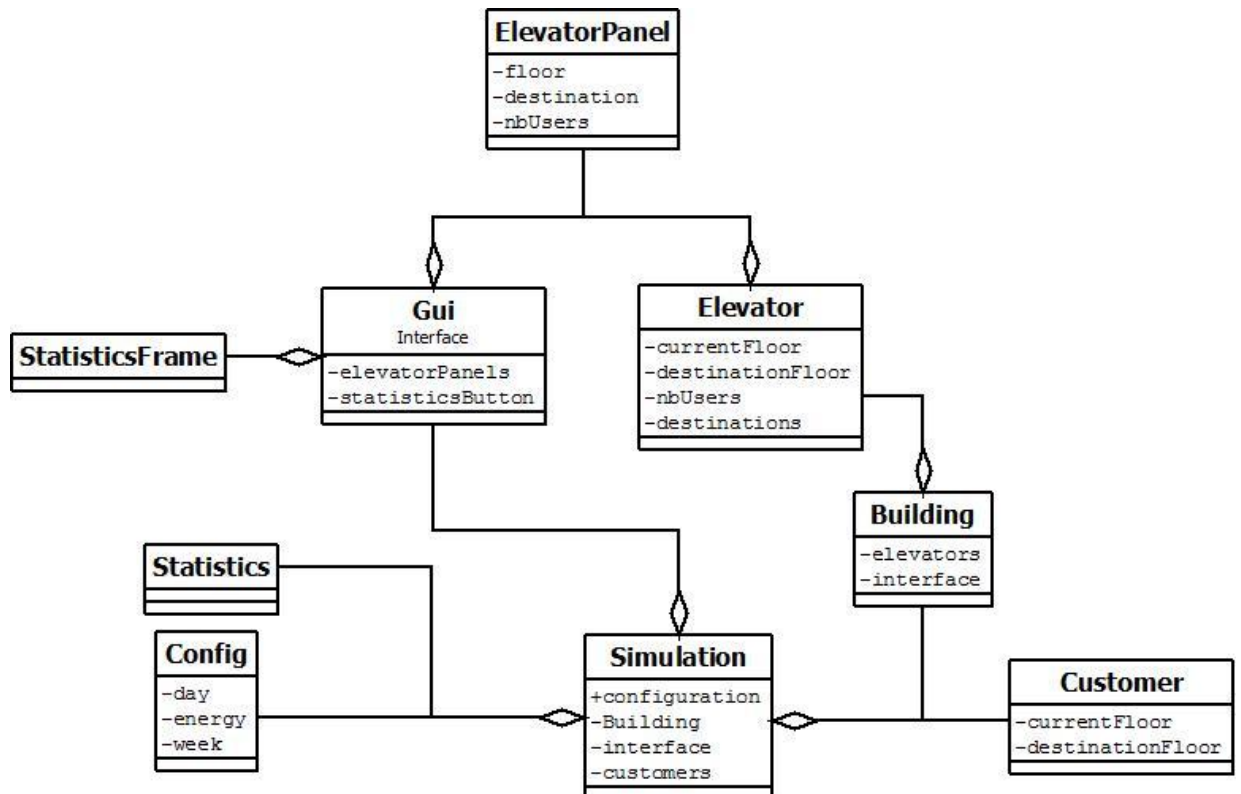
Le diagramme de responsabilité indique l'utilité de chaque classe, afin de spécifier son champ d'action au sein du logiciel. Ces précisions sont nécessaires à la bonne compréhension des classes, et ainsi au logiciel lui-même.



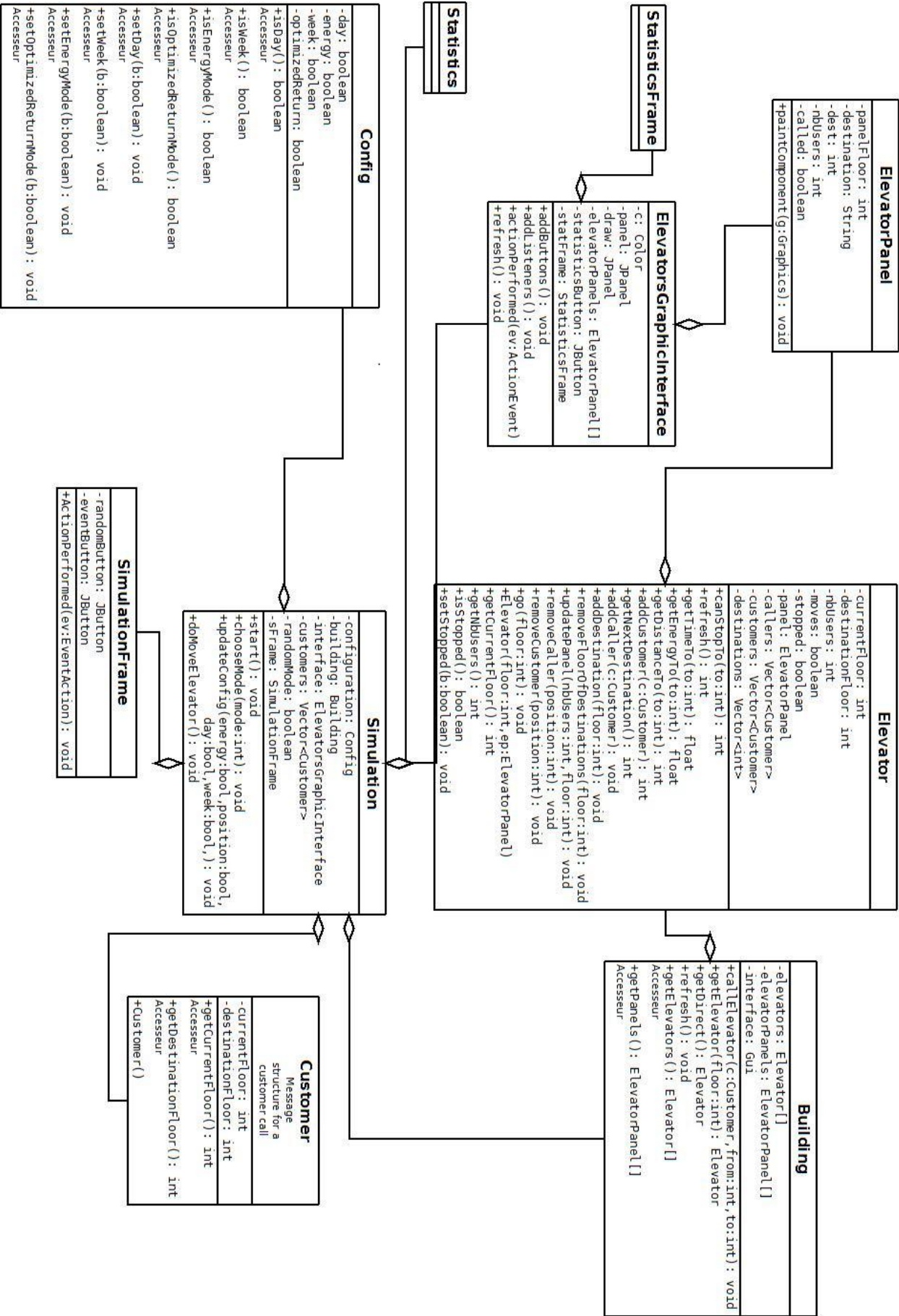
## VIII. DIAGRAMME DE CLASSE

### 1. Diagramme de classe préliminaire

Le diagramme de classe préliminaire est la transposée des idées de développement en langage objet. Il est préliminaire car non abouti. Le diagramme de classe complet est ci-après



2. Diagramme de classe détaillé



### 3. Description des principaux éléments du diagramme de classe

#### a) **Classe Elevator**

**Rôle** : permet la gestion d'un ascenseur, qui est presque autonome puisque son statut évolue en fonction de l'évolution de son environnement (rafraichissement du building) et des arrivées d'utilisateurs.

#### **Attributs particuliers :**

Callers : vecteur d'utilisateurs ayant appelé l'ascenseur, usagers qui n'ont pas encore embarqué et que l'ascenseur doit aller chercher.

Customers : vecteur d'utilisateurs embarqués dans l'ascenseur, qu'il doit déposer à leur étage de destination.

Stopped : booléen qui vaut vrai quand l'ascenseur est forcément immobile (durant l'embarquement ou le débarquement d'usagers). Pour lui le temps « s'arrête » tant que les conditions permettant de le débloquent ne sont pas réunies.

Destinations : vecteur d'entiers, triés par une méthode spécifique détaillée plus bas, qui matérialise la liste des destinations de l'ascenseur. C'est une liste d'étages auxquels l'ascenseur doit s'arrêter pour aller chercher un usager ou pour l'amener à destination.

#### **Méthodes particulières :**

- Refresh : permet le déplacement des utilisateurs, matérialise la répercussion de l'horloge de l'application sur l'ascenseur, (à travers les appels successifs des méthodes de rafraichissement des autres éléments, Building, Simulation ...)

- addCustomer : ajoute un usager à la liste des utilisateurs embarqués. Cette méthode est appelée dans la méthode refresh, dans le cas, où, à un instant T, l'étage courant de l'ascenseur est présent dans la liste des étages auxquels se trouvent les utilisateurs ayant appelé l'ascenseur. Un « caller » devient donc un « customer » à ce moment là. Elle appelle la fonction addDestination.

addDestination : on utilise cette méthode pour ajouter à la liste des destinations de l'ascenseur la destination de l'utilisateur embarqué.

- removeCaller : retire un utilisateur de la liste des callers, méthode appelée dans refresh, à la suite de addCustomer, afin de finir de transformer un caller en customer.

- addCaller : ajoute une entrée dans la liste d'utilisateurs ayant appelé l'ascenseur (ascenseur qui a été choisi pour eux par l'algorithme de sélection d'un ascenseur). Elle appelle addDestination, qui ajoute l'étage de l'appelant à la liste des destinations de l'ascenseur.

- getNextDestination : lorsqu'un ascenseur a terminé d'embarquer ou de débarquer ses usagers à l'étage auquel il est arrêté, il doit choisir la destination suivante, c'est cette méthode qui trie la liste en fonction du sens de déplacement actuel de l'ascenseur, en sens croissant ou décroissant pour éviter les retours arrière.

- removeFloorOfDestinations : cette méthode retire l'étage qui lui est passé en paramètre de la liste des destinations de l'ascenseur, après avoir embarqué ou débarqué des utilisateurs.

- canStopTo : cette méthode permet de savoir s'il est possible de s'arrêter à un étage dans un trajet donné, en prenant en compte le sens de déplacement.

- updatePanel : l'ascenseur a connaissance du panel d'affichage qui lui est associé, mais ne le redessine pas lui-même, il se contente de mettre à jour les attributs du panel en question et laisse la boucle générale de refresh de l'application se charger de la mise à jour des panels des ascenseurs.

## **b) Classe Building**

**Rôle** : permet la gestion de la batterie d'ascenseurs présente dans un bâtiment, cette classe a la connaissance de la liste des ascenseurs qui la composent et des panneaux d'affichage associés. Elle possède également la connaissance de la fenêtre qui regroupe tous ces panneaux. Pour finir, elle a la responsabilité de la sélection d'un ascenseur au moment d'un appel.

### **Méthodes particulières :**

callElevator : symbolise l'appel d'un utilisateur à un étage donné, et déclenche la sélection d'un ascenseur. L'appel débute dans cette classe puisque la classe Customer n'a de raison d'exister que pour donner une structure aux données relatives aux utilisateurs et pour modéliser le message de leurs appels d'ascenseurs. L'appel devrait donc émaner d'un utilisateur mais en réalité un Customer est passé au Building lors de la génération d'évènements par la simulation.

getElevator : permet la sélection d'un ascenseur parmi la batterie présente, en fonction de l'algorithme de sélection présenté plus haut.

getDirect : avant de lancer le calcul de l'ascenseur le plus proche, on vérifie qu'il n'y a pas d'ascenseur arrêté au niveau duquel l'utilisateur émet son appel.

## **c) Classe Simulation**

**Rôle** : la classe a deux rôles en fonction du mode de fonctionnement choisi pour l'application. Soit elle sert de générateur aléatoire d'évènements dans le cas où le mode de simulation aléatoire est choisi, soit elle sert de connexion à la batterie d'ascenseurs dans le cas où le mode de génération d'évènements est choisi et permet d'interfacer l'application avec le réel. Les saisies d'évènements dans le panneau de saisie présenté en page 11 seront à remplacer par de véritables appels d'utilisateurs, via le boîtier d'interfaçage avec la batterie d'ascenseurs.

Son comportement est défini par la classe de configuration qui regroupe les différents paramètres nécessaires au fonctionnement de l'application.

**Méthodes particulières :**

-start : démarre l'application dans le mode choisi.

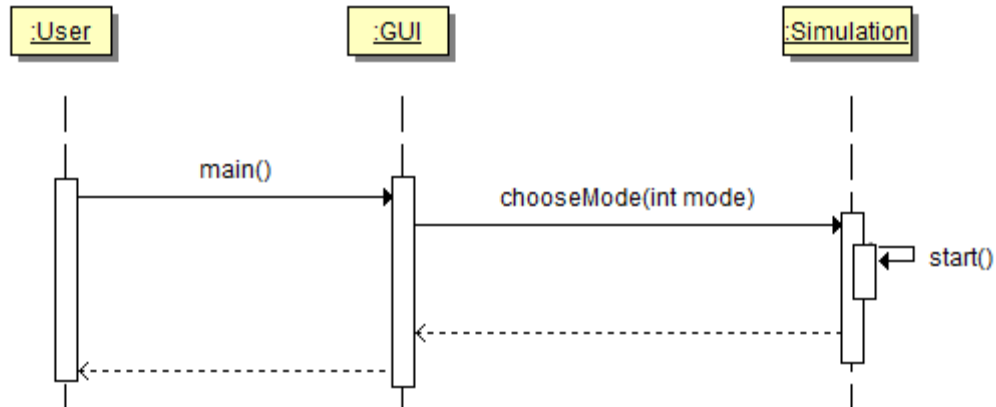
-chooseMode : permet de choisir entre le mode aléatoire et le mode saisie d'évènements.

-UpdateConfig : met à jour la configuration de l'application en fonction des choix de l'utilisateur



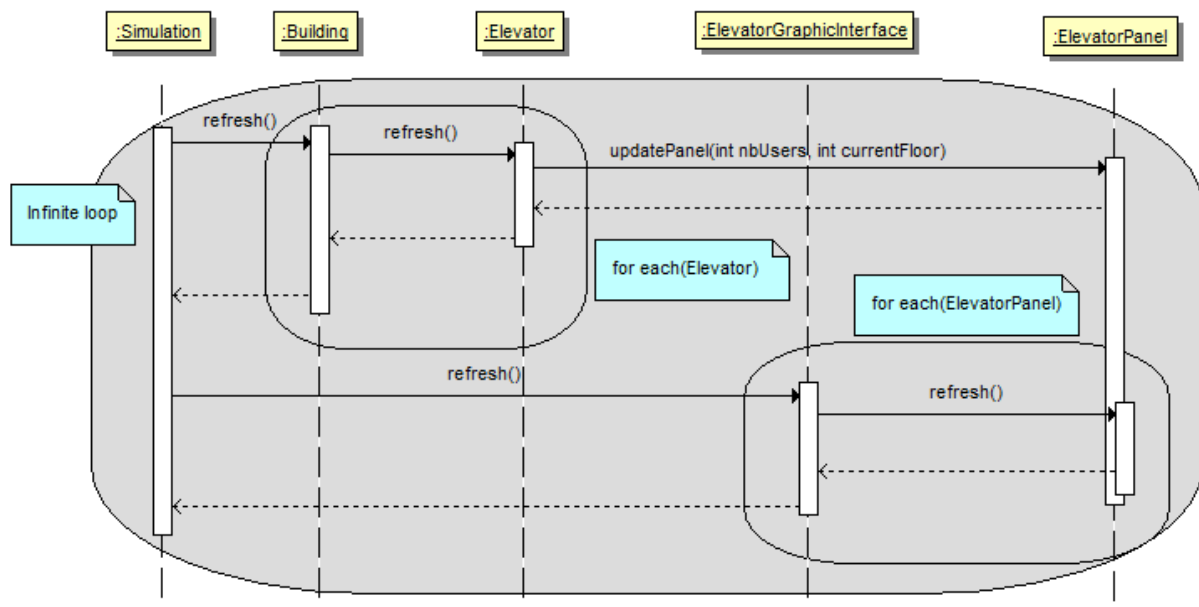
## IX. DIAGRAMMES DE SEQUENCE

### 1. Démarrage de la simulation



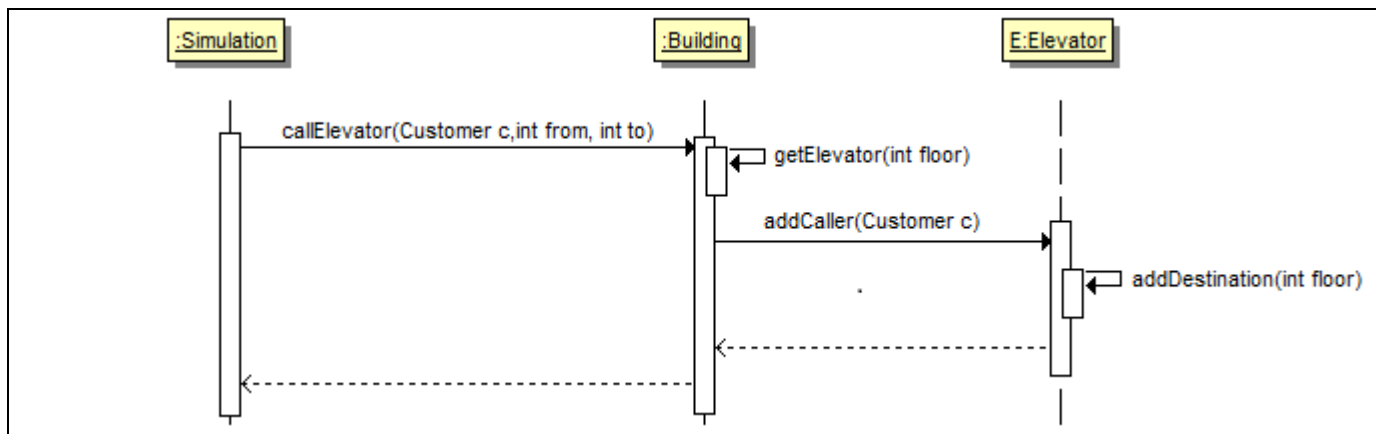
Au lancement de l'application, l'utilisateur choisit le mode de fonctionnement de l'application (Aléatoire ou Saisie d'évènements) ce qui a pour effet de lancer le départ de la simulation.

### 2. Rafraichissement global de l'application



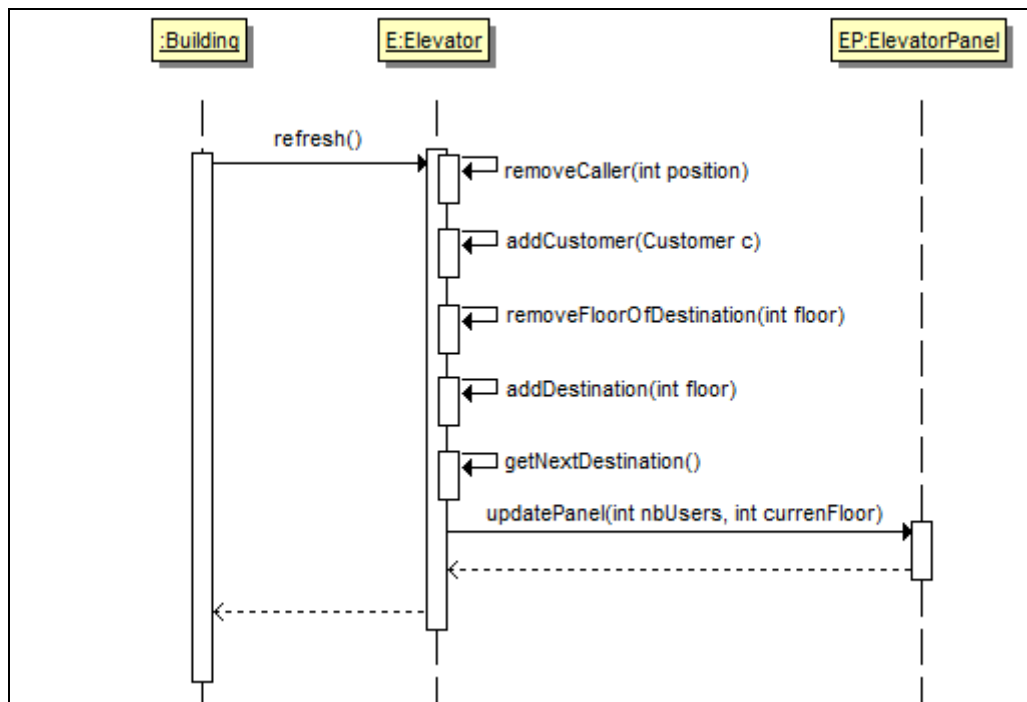
Ce diagramme de séquences symbolise le rafraichissement général de l'application (dans le cas présent, il ne se passe rien d'autre). Ce diagramme se décompose de deux parties distinctes dans lesquelles on rafraichit séparément chaque objet Elevator et chaque objet ElevatorPanel via respectivement les objets Building et ElevatorGraphicInterface.

### 3. Appel d'un ascenseur



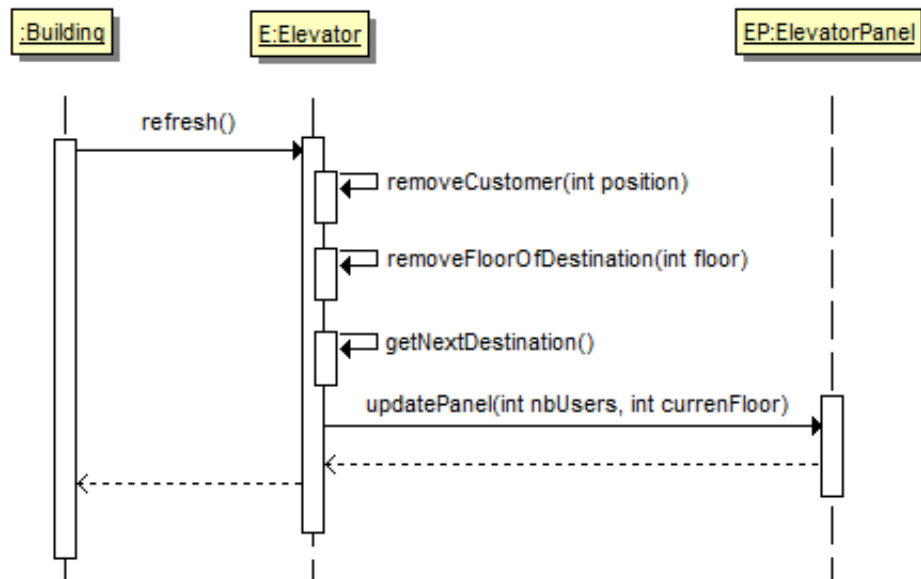
A l'appel d'un ascenseur (par la méthode `callElevator`), c'est l'objet `Building` qui, à travers la méthode `getElevator`, choisit le meilleur ascenseur via l'algorithme de sélection. Ensuite, on ajoute au tableau des appelants de l'objet `Elevator` sélectionné l'appelant en question. On peut ensuite, pour l'`Elevator`, ajouter la destination (qui sera l'étage où se trouve l'appelant et non pas sa destination) dans la liste des destinations.

#### 4. Envoi d'un ascenseur à un étage



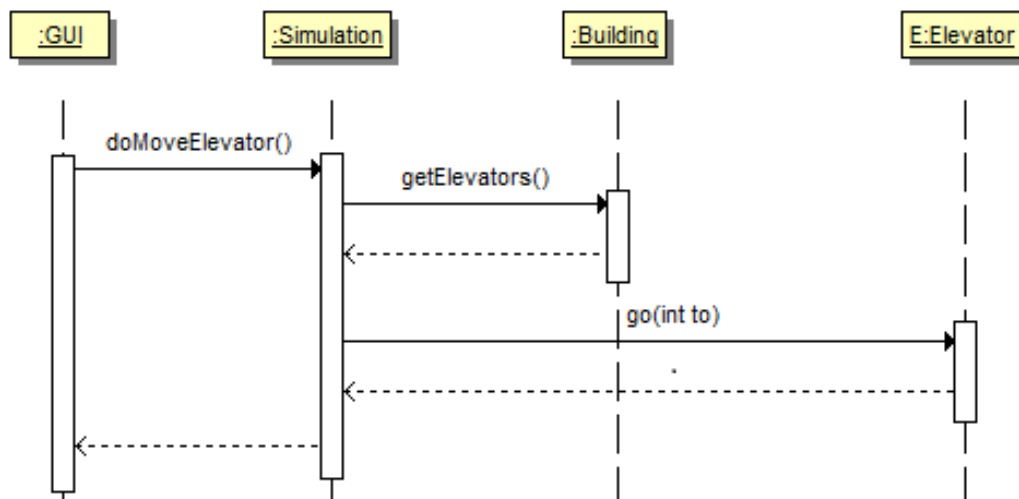
Au moment d'embarquer l'utilisateur, on le supprime d'abord du tableau des appelants de l'ascenseur ensuite on l'ajoute au tableau des clients de cet Elevator. On supprime alors la destination de l'ascenseur dans le tableau des destinations (cet étage) puis on ajoute l'étage du client à cette liste. Enfin, on récupère la prochaine destination (l'étage le plus proche sans retour arrière). Pour finir, on met à jour l'objet ElevatorPanel.

## 5. Débarquement d'un utilisateur



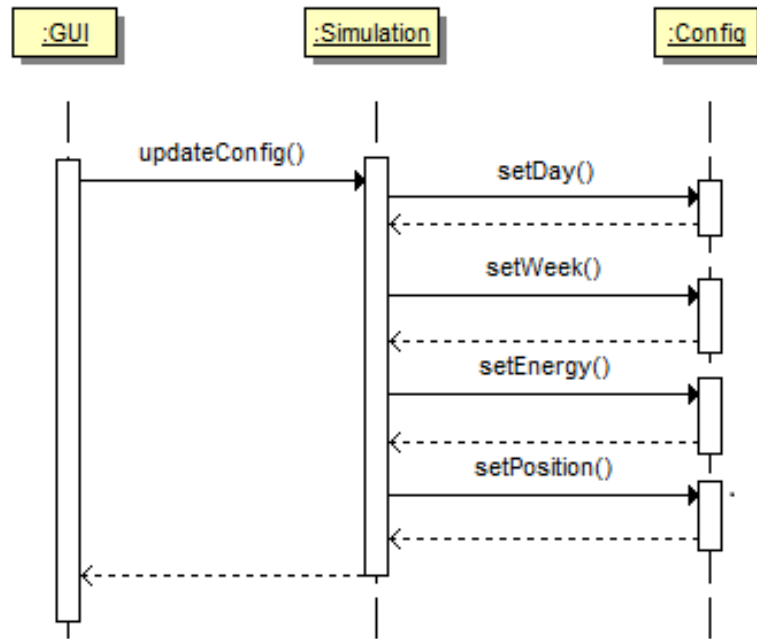
Au débarquement d'un utilisateur, on supprime le client du tableau des clients de cet Elevator. Comme pour le cas précédent, on supprime la destination de l'ascenseur dans le tableau des destinations (qui est ici l'étage courant) puis on va chercher dans ce tableau la prochaine destination. Enfin, on met à jour l'objet ElevatorPanel.

## 6. Envoi d'un ascenseur à un étage



Via l'interface graphique, on force un ascenseur à se déplacer à un étage précis. Pour cela, on récupère un tableau de tous les objets de type Elevator. Ensuite, il n'y a plus qu'à appeler la méthode `go` qui se décomposera en deux fonctions `goUp` et `goDown` pour permettre la montée et la descente.

## 7. Changement du mode de fonctionnement



Au choix de l'action de changement de configuration, on appelle la méthode `updateConfig` qui nous permettra de lancer les mutateurs de l'objet `Config`