

# Rapport

Projet tutoré S2

Noe MERIC DE BELLEFON, Thomas LACAZE

# Sommaire

<b>Introduction</b>	<b>3</b>
<b>Conception générale</b>	<b>4</b>
Diagramme de cas d'utilisation	4
Diagramme de classes	5
Diagramme de séquence	7
<b>Conception détaillée</b>	<b>8</b>
<b>Dossier de test</b>	<b>10</b>
<b>Le manuel d'utilisation</b>	<b>11</b>
Menu	11
Ouvrir/Créer une frise	11
Création d'une frise	12
Ajout d'un événement	12
Affichage d'Événements	13
<b>Conclusion</b>	<b>14</b>

## Introduction

Dans le cadre de notre projet tutoré, nous avons, Noé Meric de Bellefon et Thomas Lacaze, conçu un programme permettant de créer plusieurs frises chronologiques et de placer des événements en fonction d'un poids (l'importance de l'événement entre 1 et 4) et de l'année. Une fois l'événement créé, il est possible de visualiser directement son contenu sur une frise. Les différentes tâches que nous allons effectuer tout au long de ce projet seront la conception générale et détaillée, le développement, les tests, et la rédaction du rapport en parallèle avec toutes ces étapes.

Nous avons choisi de concevoir globalement le programme et ensuite de commencer les prémices de la programmation avant la conception détaillée, afin de pouvoir évaluer avec précision les objets qui nous seront nécessaires à l'implémentation. A la suite du développement, nous allons revenir sur la conception détaillée afin d'affiner les objets et comportements.

La conception se fera via le logiciel StarUML afin de pouvoir produire des diagrammes de classes, objets, et séquence dans la phase de conception détaillée.

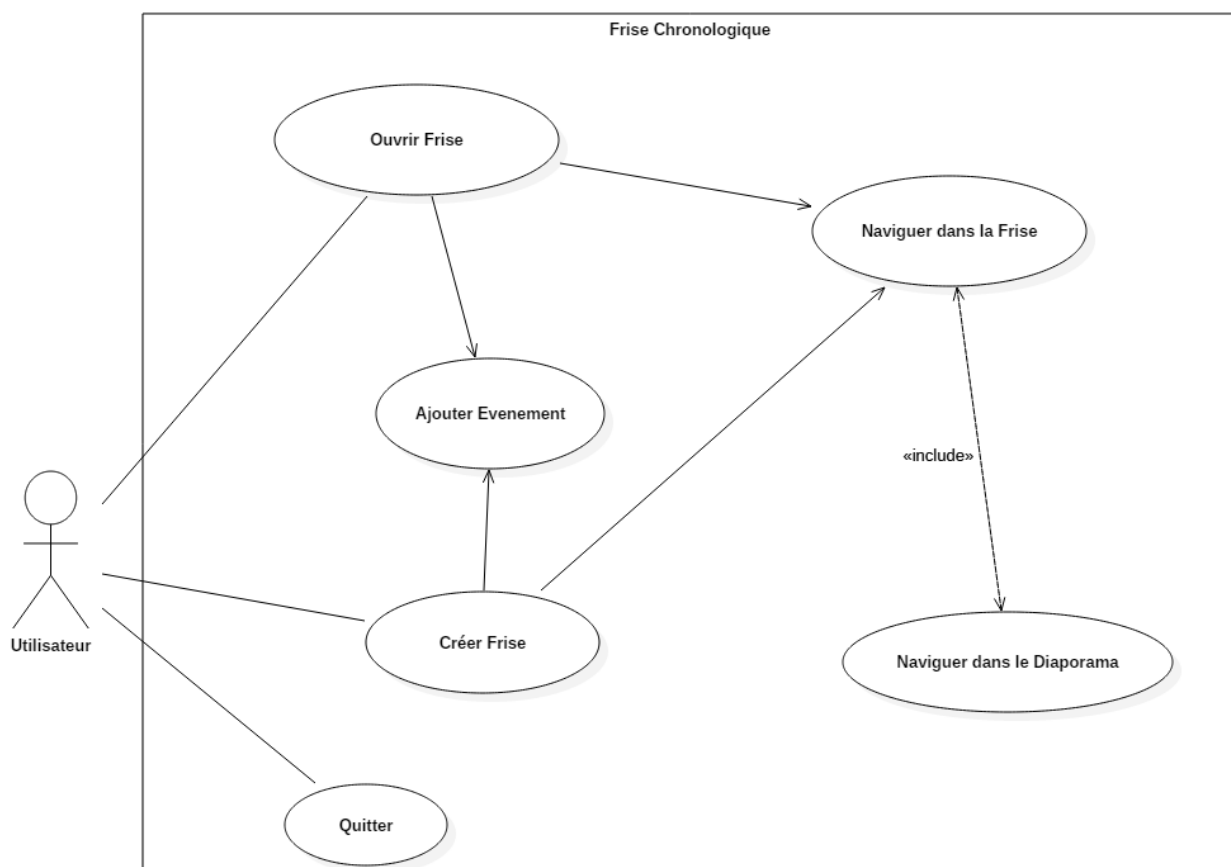
Nous effectuerons des tests avec JUnit tout au long du développement afin de réduire les problèmes majeurs en fin de programmation.

Enfin, nous allons utiliser un gestionnaire de version git afin de faciliter la collaboration et ainsi partager équitablement les différentes phases de l'implémentation grâce au site GitHub, qui nous semblait le meilleur jusqu'à son rachat par Microsoft 4 juin 2018.

# Conception générale

## Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet de visualiser toutes les actions possibles du programme, autrement dit, tout ce que peut faire l'utilisateur. C'est le diagramme qui résume le mieux et qui est compréhensible même pour les personnes ne pratiquant pas la programmation ou la conception orientée objet. C'est le diagramme qui doit être créé en premier pour avoir un aperçu global du projet.



L'utilisateur a 3 possibilités lorsqu'il ouvre le programme:

- Ouvrir une frise déjà existante.
- Créer une nouvelle frise avec un nouveau nom.
- Quitter directement le programme.

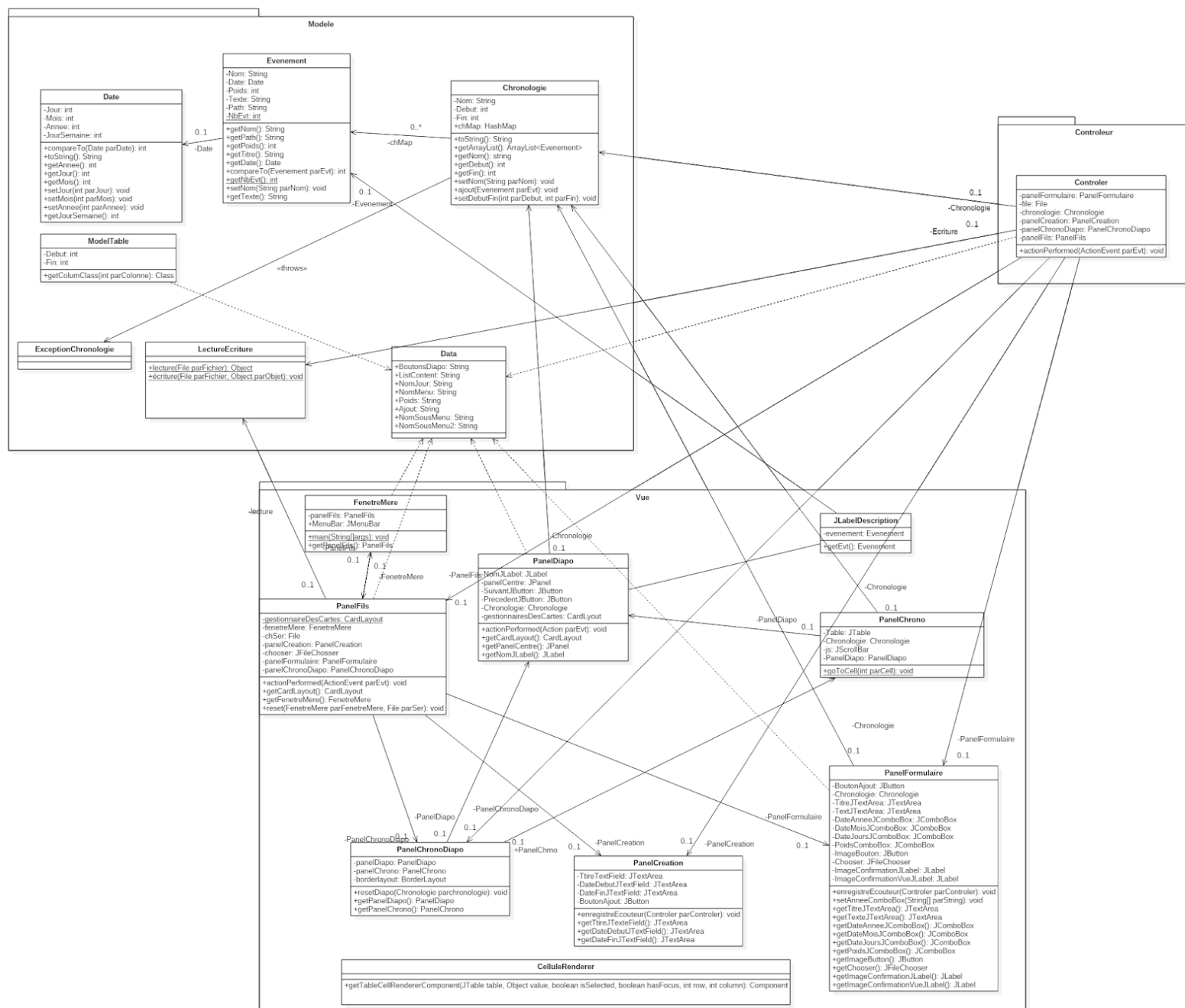
Ensuite il a la possibilité d'ajouter des événements que ce soit après avoir créé une nouvelle frise ou bien ouvert une frise déjà existante. Mais Il peut aussi voir les événements qu'il a lui même ajoutés en navigant dans le diaporama ou dans la frise chronologique.

## Diagramme de classes

Le diagramme de classes est plus conséquent que celui-ci dessus. Il présente toutes les classes avec tous les champs et méthodes qu'elles comprennent. Cela illustre en quelque sorte le squelette du programme, car en plus des champs et méthodes, il affiche aussi les liaisons entre les différentes classes (comprises dans le même package ou des packages différents) ainsi que les cardinalités et agrégations entre celles-ci.

Il nous a permis de commencer à réfléchir au contenu de chaque classe avant même de programmer donc de mieux gérer la répartition de chaque classe ainsi que leur utilité. Nous avons décidé de faire apparaître toutes les classes pour éviter d'en oublier et ainsi construire le projet sur de bonnes bases.

Tous les diagrammes présents sur le rapport sont directement disponibles sur le git dans le dossier conception.

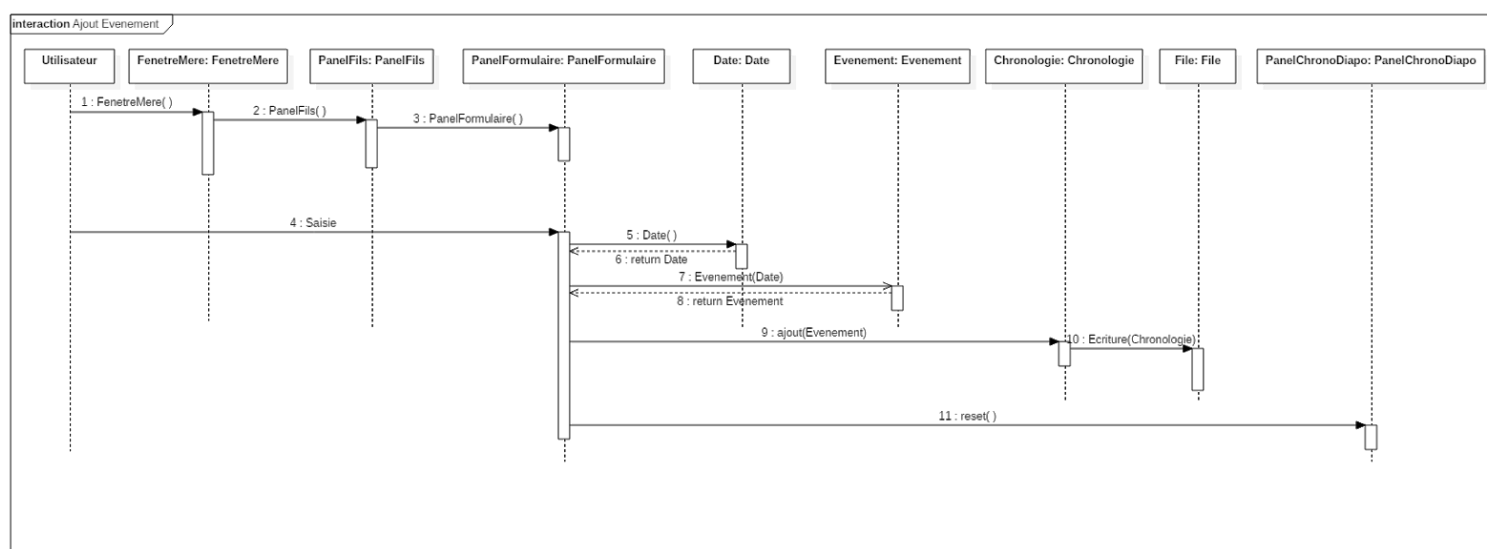


## Diagramme de séquence

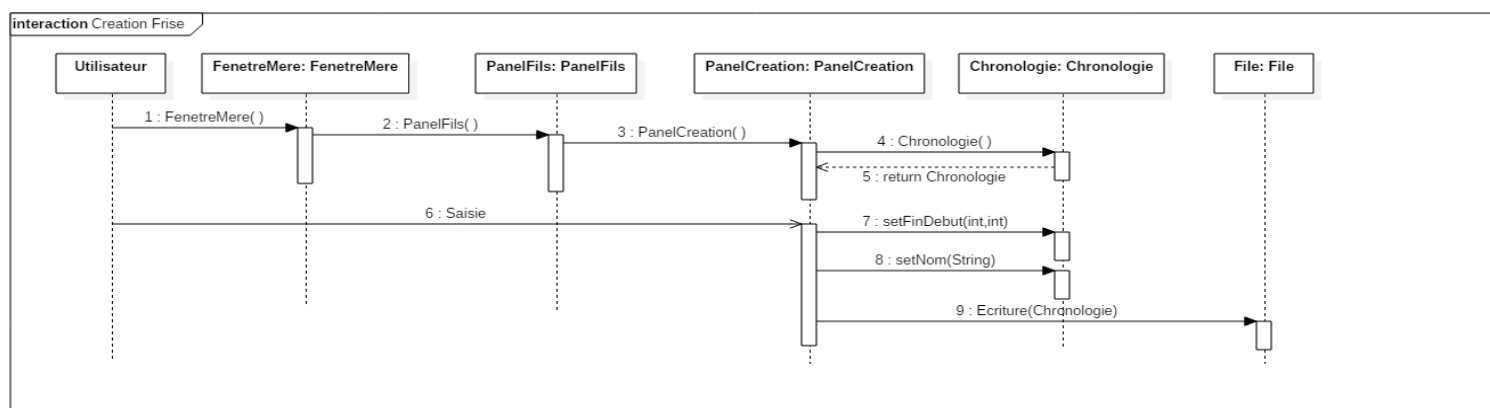
Il y a plusieurs diagrammes d'interactions, mais nous avons choisi celui de séquence.

Le diagramme de séquence a pour rôle lui, de décrire comment les éléments du programme interagissent. Il représente la séquence de messages entre les acteurs et les objets au cours d'une interaction en prenant en compte l'ensemble des participants sous forme de ligne de vie (verticale) et les messages qu'ils s'échangent au cours de l'interaction.

Nous avons pris la décision d'intégrer les getter, setter dans celui-ci.



Celui-ci représente les interactions lors de l'Ajout d'un événement. Une fois le formulaire ouvert, l'utilisateur saisit les données pour créer l'événement. On ajoute la date, l'événement à la chronologie pour le créer dans le fichier et donc ajouter l'événement à la frise(PanelChronoDiapo).



Le diagramme représente la création d'une frise. Il faut d'abord créer une chronologie grâce au PanelCreation, en assignant l'intervalle de la frise et son titre et créer la JTable correspondant au critère puis l'assigner à un fichier qui pourra être sauvegardé puis réouvert.

## Conception détaillée

Le programme est réparti en fonction de l'architecture MVC (Model View Controller):

- Package Model est composé des différentes classes comme la Chronologie, Date ou encore Événement.
- Package View est composé de la partie interface.
- Package Controller est composé des actions pour la partie diapo, frise chronologie, création d'une frise ou encore ajout d'un événement.

Si une classe possède un attribut alors il a forcément un préfixe "ch" de même pour les paramètres qui seront eux précédés d'un préfixe "par".

Tous les attributs sont "private" et disposent d'un getter et setter si nécessaire.

La classe Chronologie stocke les événements dans une HashMap permettant de les classer en fonction de leur date. La clé de la HashMap est l'année d'un événement. La majeure partie des constantes se situent dans l'interface Data, pour faciliter la traduction si nécessaire.

En nous focalisant sur un aspect très proche du code :

Pour le package Model:

- Classe Chronologie permettra de stocker des Événements avec une année de début, de fin, et un nom pour cette chronologie.
- Classe ExceptionChronologie permettra de relever une erreur lors de l'ajout d'un Événement qui comporte le même poids et la même année.
- Classe Événement permettra d'associer une Date à un nom d'événement, une image, un titre et une description et un poids entre 1 et 4.
- Classe Date permettra de mieux gérer le toString() lors de l'affichage d'un Événement, cette classe n'est pas nécessaire mais permet de mieux structurer le projet.



- Classe data comportera toutes les constantes utilisées par le programme pour l'IHM, ainsi cela pourra permettre une traduction du programme si nécessaire plus facilement après son développement.

Pour le package View:

- Classe Fenêtre mère servira de main à l'application. Elle héritera de JFrame,instanciera un PanelFils et configurera la taille ainsi que la position de la fenêtre. Elle instanciera également le PanelFils qui est un contentPane.
- Classe PanelFils héritera de JPanel et instanciera un objet JMenu et un CardLayout dans lequel on ajoutera l'interface de saisie, un diaporama et une frise chronologique, respectivement JPanel, CardLayout, DefaultTableModel avec DefaultTableCellRenderer
- Classe JLabelDescription servira à contenir l'image et la description d'un événement. Nous utiliserons l'héritage pour mettre en application tous les aspects vus en cours d'année. Cela permettra de mieux séparer les classes donc la maintenabilité et la compréhension du code

Pour le package Controller:

- Classe Controller permettra de retrouver les principales actions qui contribue à créer une frise ou même d'ajouter un événement à une frise déjà existante. Donc toutes les actions qui nécessitent l'utilisation du Package Model comme le veut l'architecture MVC. Cependant pour améliorer la maintenabilité du code en évitant de créer des getter et setter partout, nous avons opté pour laisser les actions comme les mouseListener ou encore l'interaction avec les boutons d'un diaporama directement dans leur classe d'instanciation pour éviter la non lisibilité du code.

Pour plus d'informations le code est commenté, et dispose d'une JavaDoc que vous trouverez donc directement sur le git.

## Dossier de test

Les tests représentent une grande partie du temps passé sur le projet. Ils avaient pour but de vérifier que les classes écrites ne comportaient pas de bug.

Nous avons effectué plusieurs tests boîtes noires pour la création d'une chronologie et l'ajout d'événements Manuel d'utilisation. Ils nous ont permis de nous assurer dès le début que les classes présentent dans le package Model ne comportaient pas de bugs et pouvaient être une bonne base pour la suite du développement du projet. Junit a beaucoup été utilisé en début de projet puis par la suite intégré au git en fin de projet.

Le premier test correspond à l'ajout de deux événements ayant la même année et le même poids. Ce cas n'est pas censé fonctionner car sur la frise les deux événements se chevauchent. Nous avons donc utilisé des Exceptions pour palier à ce problème.

Le second test correspond à l'utilisation de comparable d'une date directement dans un événement. Pour ainsi valider la méthode compareTo que nous avons écrit.

Le troisième test correspond à l'ajout de deux événements dans une frise chronologie est vérifier qu'ils sont bien présent avec la méthode getNb qui récupère la "size" de la hashMap.

Le dernier test nous a permis de vérifier si deux événements pouvaient être comparé entre eux grâce encore une fois au méthode compareTo.

Nous avons fait beaucoup de test mais pour éviter de rendre un rapport trop long nous avons opté pour choisir les principaux tests qui ont été effectué dès le début. Une aperçu des tests est disponible sur le git dans le Package test

## Le manuel d'utilisation

Il a pour but d'expliquer le fonctionnement du logiciel au néophyte. De plus le programme dispose de mnémonique pour faciliter la saisie et permettre au plus grand nombre d'utiliser l'application dans les meilleures conditions.

Le programme dispose de messages d'erreurs ce qui vous permet de mieux comprendre les problèmes et de les résoudre.

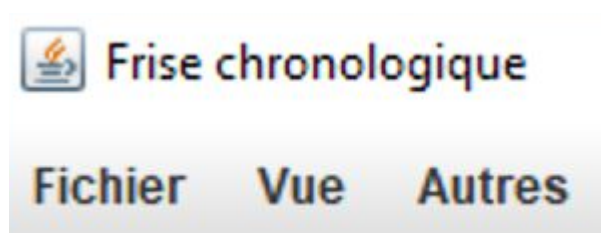
### Menu

Le programme est composé de trois menus (fichier, vue, autres).

Le premier permettant d'ouvrir une frise chronologique déjà existante ou simplement une nouvelle frise, en lui donnant un nouveau nom.

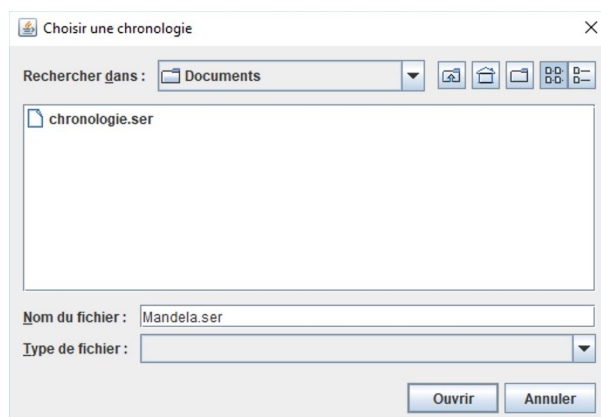
Le deuxième menu (Vue) permet d'afficher la frise chronologique et le diaporama des événements ou d'ajouter un nouvel événement.

Le dernier menu (Autres) permet de quitter le programme.



### Ouvrir/Créer une frise

Pour ouvrir une frise déjà créée au préalable il suffit d'aller dans le menu fichier et cliquer sur ouvrir puis sélectionner la frise que vous souhaitez. Si vous voulez créer une nouvelle frise il vous suffit de rentrer un nouveau nom puis de cliquer sur ouvrir et un menu apparaîtra.




## Création d'une frise

Le menu qui vous permet de créer une frise apparaît automatiquement lorsque le logiciel en a besoin. Il demandera un titre à votre chronologie, ainsi qu'une année de début et de fin.

## Ajout d'un événement

L'ajout d'un événement se fait par le menu "Vue". Il consiste à ajouter un élément sur la frise chronologique et sur le diaporama. Votre événement devra forcément contenir un titre, un texte, une date, un poids et une image que vous pouvez prévisualiser en bas une fois sélectionné.

## Affichage d'Événements






Robert Downey Junior

29 mars 1993

**Cérémonie des Oscars de 1993**

Il est nommé à l'Oscar du meilleur acteur pour son interprétation de Charlie Chaplin dans Chaplin: film de Richard Attenborough qui retrace la biographie de Charles Chaplin. Il est à ce moment reconnu comme un grand acteur de sa génération Il a alors 28 ans et son avenir est prometteur.

			1995					2000					2005			
																
																

Voici l’affichage que vous aurez lorsque vous allez ajouter quelques événements à votre frise. Il est composé de deux parties :

- La première se situe en haut et correspond à un diaporama qui contient tous vos événements ajoutés, classés par année, par ordre croissant. Il est possible de naviguer avec les flèches qui sont de part et d'autre du diaporama.
- La partie du bas est quant à elle composée d'une frise chronologique. Une colonne représente une année et une ligne représente un poids. Plus l'événement est important plus il est situé haut donc avec un poids faible. Une image de votre événement est présente directement dans la case où votre événement doit être situé. Vous avez directement la possibilité de cliquer sur un événement pour l'afficher dans le diaporama.

## Conclusion

La conception générale et détaillée de ce projet nous a mobilisé environ 15 heures. L'implémentation et la mise en place des tests a, quant à elle, nécessité 12 heures. La rédaction du rapport s'est faite tout au long du projet, du début de la conception, jusqu'à la fin des tests.

Ce projet nous a permis de mettre en pratique les méthodes de conception et de programmation vues en cours et en travaux pratiques. L'utilisation de git, nous a été bénéfique et nous a permis d'avoir une meilleurs gestions du temps et de la répartition du travail, afin de pouvoir rendre un programme et un compte rendu complet pour la dead-line.

Le programme nous a donc permis d'utiliser tous les aspects enseignés ce semestre concernant la programmation orientée objet et de nous donner un avant-goût du projet tutoré de l'année prochaine.