|  |
| --- |
|  |
| Projet fil rouge |
| [Rendu n°2] |

|  |
| --- |
| TD N  12/04/2018 |

QUIBEL Florian & ROBIC Gaëtan

Table des matières

[I/ Introduction : 1](#_Toc511151731)

[II/ Présentation générale : 1](#_Toc511151732)

[III/ La classe Parc 1](#_Toc511151733)

[IV/ Le personnel 2](#_Toc511151734)

[V/ Les attractions 2](#_Toc511151735)

[VI/ L’affichage en console 3](#_Toc511151736)

[VII/ Conclusion 3](#_Toc511151737)

# 

# I/ Introduction :

Le parc d’attraction Zombillenium a besoin d’un système de gestion globale de son parc d’attraction. Il s’agit de pouvoir gérer les attractions ainsi que le personnel et de pouvoir les faire interagir entre eux. L’objectif est de répertorier l’ensemble des informations du parc et de pouvoir les modifier autant qu’on le désire grâce à un programme en C#.

# II/ Présentation générale :

La première chose à faire était de créer chaque classe avec les attributs imposés par le diagramme de classe.

Ensuite nous avez décidé de créer la classe Parc qui correspond à l’administration. Elle comporte les informations générales sur le parc, l’ensemble des attractions et des employés. Que ce soit pour la direction, pour le personnel ou pour l’ensemble des employés, nous avons décidé de les regrouper dans des listes afin de mettre en œuvre ce que nous avons appris durant ce semestre et ainsi pour utiliser toutes les fonctions qui sont propres aux listes.

Il y a donc 3 listes principales afin de répondre au mieux aux besoins du parc :

\_La direction : Mr Von Bloodt et son équipe qui ont accès à l’ensemble des informations.

\_Les attractions : Peu importe le type d’attraction que ce soit à sensation fortes, des spectacles ou même des boutiques.

\_Le personnel : Il correspond à l’ensemble des employés ainsi que leur responsable syndical.

# III/ La classe Parc

Si l’on s’intéresse à chaque classe une par une, on peut voir que la classe mère de la gestion est la classe Parc. C’est en effet à partir d’elle que l’on a les fonctions :

\_Ajouter ou supprimer des attractions, des membres de direction ou des employés.

\_Trier n’importe quelle liste par critère seulement ou par critère et par classe, en utilisant les propriétés génériques . Il suffit d’entrer en console la liste sur laquelle on veut appliquer le tri et le critère choisi.

\_Interagir avec des fichiers .CSV, que ce soit de le lire afin d’appliquer le Peuplement de nos listes ou d’écrire sur un nouveau .CSV, sur lequel on peut écrire l’ensemble des personnes d’une liste (par exemple du Personnel) avec toutes leurs informations.

\_Afficher toutes les personnes d’une liste, c’est-à-dire l’ensemble de leurs informations contenu dans la fonction ToString() de chaque classe.

# IV/ Le personnel

Comme on l’a dit précédemment, nous avons séparé le personnel de l’équipe de direction. On s’intéresse donc ici à un responsable syndical et tous les employés qui veillent au bon fonctionnement du parc. En d’autres termes, il regroupe les sorciers et les monstres (qui eux peuvent être vampires, démons, fantômes, zombies, loups-garous). Il faut savoir que l’on peut afficher la liste des personnes présentes dans le personnel. Pour le reste, il faut s’intéresser aux 2 classes qui en héritent.

D’une part les Sorciers, caractérisés par leur tatouage et leurs pouvoirs. Leur gestion est assez simple, on peut soit modifier leur tatouage, soit ajouter/supprimer des pouvoirs. D’autre part les Monstres, qui eux possède une cagnotte qui peut être modifiée par le directeur et qui peut leur affecter des bonus ou des malus en fonction de sa valeur. Mais par ailleurs, il faut toujours qu’il y ai assez de personnes affectés à l’attraction sans quoi l’employé ne peut pas quitter l’attraction. Il faut savoir que les monstres sont les employés qui travaillent sur les attractions ils y sont donc affectés mais cette affectation n’est pas fixe et peut être modifiée.

Il existe 5 sous classes qui héritent de Monstre. Ce sont les vampires, démons, fantômes, zombies, loups-garous. Chacun possède ses spécificités que l’on peut modifier : la luminosité pour les vampires, la force pour les démons, l’invisibilité pour les fantômes, la teinte et le degré de décomposition pour les zombies et enfin l’indice de cruauté pour les loups-garous. Ce sont des détails propres à chaque employé qu’il faut donc pouvoir modifier mais pour la gestion actuelle du parc, ces arguments ne sont pas liés à des fonctions spécifiques mais c’est quelque chose qui pourrait être facilement ajouté si besoin dans la gestion du parc.

# V/ Les attractions

Les attractions regroupent les manèges et les boutiques, ils ont un principe de fonctionnement équivalent. Tous on besoin d’un nombre minimal d’employés pour fonctionner et possède plein d’informations comme le nombre d’entrées ou bien si cette attraction est en fonctionnement ou maintenance (dans ce cas la raison de la maintenance est écrite). On peut alors modifier le personnel qui travail dans une des attractions, afficher les informations concernant une attraction et la mettre ou la retirer du fonctionnement. Pour la gestion du parc, il suffit de pouvoir modifier à la fois les attractions en elles même (quelque soit leur type) et modifier les personnes qui y travaillent.

# VI/ L’affichage en console

Pour montrer les différentes possibilités pour ce logiciel de gestion, nous avons décidé de vous en montrer certaines dans la console. Il en existe beaucoup d’autre que l’on peut tester qui sont présentes dans les différentes classes afin de gérer au mieux le parc. Nous avons commencé par instancier un parc, ensuite nous avons chargé le fichier .CSV Projet\_Zombi afin de remplir notre parc avec les employés et les attractions fournies. Deuxième étape, l’ajout, d’une attraction et d’un employé Démon pour montrer que l’on peut ajouter ce que l’on souhait au parc. Ensuite la recherche d’un employé par son nom (on pourrait le rechercher aussi par son matricule) et la modification de son degré de décomposition, on peut donc modifier les caractéristiques des personnes déjà présentes dans l’entreprise. Vient ensuite la mise en maintenance d’une attraction avec un motif qui est « révision » puis sa remise en service. On la remet ensuite de nouveau en maintenance pour le même motif afin d’effectuer un tri sur toutes les attractions afin d’obtenir celles qui sont en maintenance, puis les afficher. Ensuite un affichage complet des zombies qui travaillent dans le parc suivi du même affichage une fois les zombies triés par rapport à leur cagnotte dans un ordre croissant. La dernière chose est la création d’un fichier .CSV afin de pouvoir récupérer la liste des fantômes et de leurs informations de manière permanente.

# 

# VII/ Conclusion

Pour résumer, nous avons créé un logiciel de gestion pour le parc Zombillenium. Il peut être décomposé en 3 parties majeures qui sont l’administration, les employés, les attractions. Pour chacun d’entre eux, il y a des fonctions communes comme l’affichage de l’ensemble des personnes présentes ou d’un individu spécifique, l’ajout ou la suppression d’un individu, la lecture d’un fichier csv afin de peupler l’ensemble de ces classes, l’écriture d’un fichier csv avec les informations souhaitées (l’ensemble des zombies par exemple). Afin d’avoir une vision plus claire d’une liste (peut importe laquelle) on peut aussi la trier par classe et/ou par critère peut importe le critère choisi. On peut aussi modifier toutes les informations qui définissent une des instances (y compris la cagnotte). La gestion du parc est donc claire, l’administration peut maintenant agir sur tout ce qu’elle veut comme elle le souhaite, afficher les listes qu’elle veut afin d’avoir une meilleure vision du parc et vérifier toutes les informations que ce soit des employés ou des attractions afin de pouvoir optimiser son parc selon sa politique.