|  |  |
| --- | --- |
|  | Rapport Projet POO 2017-2018 |
|  |  |
| 08/01/2018 | DEVECIOGLU Serhat et MOUNSI Arezki GR.I2 |
|  | |

Rapport Projet POO 2017-2018

DEVECIOGLU Serhat et MOUNSI Arezki GR.I2

Pour commencer nous avons suivi les instructions donner à la lettre sur le pdf, nous avons essayé de nous répartir au maximum les différentes tâches tout en essayant de rester clair et d’améliorer par la suite le code et éventuellement traquer les différents bugs.

Arezki a commencer avec la classe Spectacle, dans cette classe il a fait des getters et setters pour nous faciliter nos requêtes concernant les interprètes et les titres pour la suite, ensuite il a fait un equals en passant en paramètre un objet pour confirmer l’égalité de plusieurs spectacles qui peuvent avoir les même interprètes et titres et dans ce cas retourne un vrai ou faux et un hashcode dont les deux sont complémentaires et permettent de comparer des instances. Pour finir il a redéfini la classe toString pour afficher le titre et les éventuelles interprètes.

Ensuite il a fait la classe Film qui hérite de la classe Spectacle, ici il appelle dans le constructeur Film un super pour hériter du titre et interprètes déjà coder dans la classe Spectacle et initialise le réalisateur et la durée. Il a par la suite initialisé des getters et setters pour réalisateur et durée puis il a redéfini la classe toString pour afficher le réalisateur et la durée d’un film.

Pour la classe Heure il s’est aider de la classe que nous avions vu en TP, il a rajouté pour améliorer le code les fonctions hashCode et equals qui ont la même fonction comme dit précédemment ci-dessus et a redéfini la fonction toString pour pouvoir afficher les heures.

Enfin pour la classe PieceTheatre, il a procédé comme la classe Film en héritant des fonctions de la classe Spectacle, en initialisant les deux variables metteurEnScene et nbEntractes avec des getters et setters et pour finir il a redéfini la classe toString pour afficher le nom du metteur en scène et le nombre d’entractes de la pièce.

Pour cette première partie, Arezki n’a pas eu beaucoup de soucis à faire les classes en s’aidant notamment de ce que nous avons vu en TP/TD et de ces connaissances personnelles.

Pour la suite Serhat à continuer avec ce que Arezki à fait, donc il a créé la classe Salle avec les instructions fournies, puis initialiser chaque variable de la classe en appelant le constructeur. Il a ensuite redéfini la classe toString pour afficher la capacité, le prix standard, le prix réduit avec une réduction de 60% et le nombre de places d’une Salle passer en paramètre. Il a aussi initialisé des getters et setters pour les variables de la classe Salle.

Ensuite pour la classe SalleTheatre il a hériter la classe Salle pour ne pas avoir à récrire ce qu’il a coder précédemment, il a initialisé les variables nbFauteuils et prixFauteuil, il a redéfini la classe toString de sorte qu’il nous affiche le prix du fauteuil et le nombre de fauteuils. Il a initialisé les variables dans des getters et setters pour les variables ci-dessus. Cette classe étant assez courte et compréhensible il n’a pas eu beaucoup de mal à le coder.

Il a continué avec la classe abstraite Seance, il a créé les variables et il les a initialisées comme demander en appelant le constructeur de la classe, ensuite il a écrit dans cette classe les méthodes demander dans le pdf et il a aussi ajouter un compareTo par la suite pour comparer deux séances pour éviter un conflit au niveau des horaires. Il a ici aussi fait des getters setters pour les variables de la classe.

Par la suite il a fait la classe SeanceCinema qui hérite de la classe Seance, il a initialisé les variables salle et nbPlacesVenduesTR et fait des getters setters pour ceux-là. Ensuite il a fait des méthodes qui ont été redéfinies à partir de la classe mère comme le nombre de places disponible pour une séance de cinéma, le taux de remplissage de la séance et le total des places vendues. Il a aussi fait un sous-programme pour appeler la fonction vendre une place à tarif réduit.

Enfin pour la classe SeanceTheatre il n’a pas eu vraiment de mal à le faire car c’était à peu près la même chose que la classe SeanceCinema donc cela a été rapide.

Serhat n’a pas eu de mal dans cette partie du projet car pour lui c’était assez compréhensible avec les instructions données.

Pour continuer il fallait bien associer à chaque film des séances, donc on a décidé tous les deux de créer une nouvelle classe :

La classe ProgrammationFilm qui a pour but de créer des listes de séance car un film peut avoir plusieurs séances dans un et même jour.

Ensuite on a ajouté les différentes méthodes comme ajout, remove….

On a aussi ajouté un sort par date tout en utilisant une classe interne, une méthode rechercher qui recherche les seancefilm du jour donné et de l’heure donnée, une méthode qui retourne le taux de remplissage, une méthode pour le chiffre d’affaire puis implémenter les toStrings que l’on avait redéfini au préalable.

La classe ProgramationTheatre étant presque la même chose nous n’avons pas eu du mal et de perte de temps pour le faire.

Enfin pour associer aux films les séances on a créer la classe ProgramationSemaine et donc :

On a utilisé un hashmap pour ne pas avoir besoins de trier les films et pour pouvoir comparer les programation semaine on a redéfini les méthodes hashcode et equals.

Enfin pour la classe GestionSpectacles qui contient le main nous avons fait comme suit :

Arezki a commencé par faire les méthodes qui sont chargementsalleCinema et chargementsalletheatre.

Pendant que Serhat travaillait un peu sur les méthodes programmations semaines et actions sur les programmations d’une semaine.

Nous avons remarqué que c’était difficile de travailler sur les autres méthodes chacun de son côté alors on a décidé de se voir et les faire ensemble.

Donc pour que notre méthode programmations d’une semaine marche bien il a fallu que nous mettions en place une méthode infos spectacles qui lui aller être en lien avec quatre autres méthode, une pour l’informations sur les films, une pour celui des pièces, ensuite une pour l’informations sur les séances de d’un film et pour finir une sur les informations des séances de théâtre.

Après que nous avions les infos pour les programmations, il fallait donc une fonction pour pouvoir vendre des places de ciné et des places de théâtre donc notre fonction vendre programmation de la semaine avait recours à deux autres fonctions, l’une est celle pour vendre des places de cinéma et l’autre permet la vente des places de théâtre.

Il nous fallait aussi la possibilité de pouvoir ajouter ou supprimer une programmation dans la semaine, pour cela nous avons fait une fonction modification programmation semaine qui va nous permettre de choisir entre l’ajout d’une séance film ou bien la suppression d’une séance film et l’ajout d’une séance de théâtre ou la suppression d’une séance de théâtre et tout cela pour une semaine donnée.

Pour finir il nous fallait créer la programmation d’une semaine et pour cela on devait proposer deux choix, soit créer un film soit créer une pièce de théâtre, du coup nous avons fait une fonction création programmation semaine qui aller donc avoir besoin de création programmation semaine théâtre ou bien création programmation semaine film.

Avec tout cela nous avons testé notre programme et il marche parfaitement avec toute les combinaisons possibles, à défaut d’un bug que nous n’aurions pas pu constater au moment de ces tests.

En conclusion, on peut dire que ce projet nous a pris du temps malgré d’autre projet qui était en cours mais nous avons beaucoup apprit à travers celui-ci et nous pensons avoir réussi le projet dans le temps imparti, nous avons appris que programmé c’est avant tout communiqué avec son binôme, avoir son point de vue et s’entraider, gérée son temps surtout que nous avions 2 semaines de vacances pour travailler ce projet et celui concernant PHP/MVC.