

Annee UNIVERSITAIRE: 2023-2024

**Rapport de projet**

**3ème année**

**Ingénierie Informatique et Réseaux**

**Encadré par :**

Tuteur de l’école : Pr. Mariame Amine

Gestion d’un parc

**Réalisé par :**

Ilyass Baba

**Sommaire**

1 - Introduction …………………………………………………………. 2

2 - Objectif ………………………………………………………………… 3

3 - Realisation ……………………………………………………………. 4

3.1 – Presentation de class …………………………..……... 5

3.2 – Presentation du Main(){} …………………………... 8

**Introduction**

Ce projet vise à concevoir un système de gestion avancé pour un parc de jeux en utilisant le langage C++ et en mettant en œuvre les principes d'héritage et de polymorphisme. L'objectif principal de ce projet est de développer une plateforme robuste et conviviale pour la gestion efficace des différents jeux disponibles dans le parc, en prenant en compte leurs spécificités et leurs caractéristiques uniques.

Le système de gestion du parc de jeux sera conçu de manière à offrir une interface utilisateur intuitive et conviviale, permettant aux responsables du parc de gérer facilement les jeux, de suivre leur disponibilité, de gérer les réservations, et d'accéder à des informations détaillées sur chaque jeu. Grâce à l'utilisation de l'héritage et du polymorphisme, le système sera capable de traiter différents types de jeux, tels que les jeux d'extérieur et les jeux d'intérieur, en fournissant des fonctionnalités spécifiques à chaque type.

En utilisant les concepts avancés de la programmation orientée objet, ce projet vise à fournir une solution flexible et évolutive pour la gestion des parcs de jeux. Le système sera conçu pour gérer efficacement la capacité maximale du parc, en limitant les réservations en fonction de cette capacité. De plus, il permettra de générer des rapports et des statistiques pour évaluer l'utilisation des jeux, les préférences des visiteurs et faciliter la prise de décision en matière de gestion et d'optimisation des ressources.

En résumé, ce projet ambitieux vise à développer un système de gestion complet et professionnel pour les parcs de jeux, en utilisant les principes d'héritage et de polymorphisme du langage C++. Il offrira une plateforme conviviale et robuste pour la gestion des jeux, en tenant compte de leurs spécificités et de leurs caractéristiques uniques, tout en permettant une gestion efficace de la capacité du parc et en fournissant des fonctionnalités avancées telles que la génération de rapports et de statistiques.

**Objectif**

L'objectif principal de ce projet ambitieux est de développer un système complet de gestion pour un parc de jeux, en mettant en œuvre les principes d'héritage et de polymorphisme du langage C++. Ce système offrira une plateforme avancée et conviviale pour l'ajout, la gestion et l'affichage des différents jeux disponibles dans le parc.

Grâce à l'utilisation de l'héritage, le système sera capable de traiter différents types de jeux, tels que les jeux d'extérieur et les jeux d'intérieur, en fournissant des fonctionnalités spécifiques à chaque type. Par exemple, il pourra gérer les jeux d'extérieur en prenant en compte des caractéristiques telles que la surface nécessaire et les conditions météorologiques appropriées. De même, pour les jeux d'intérieur, le système pourra gérer des aspects tels que la capacité maximale et les restrictions d'âge.

Le polymorphisme permettra au système de gérer efficacement les jeux en utilisant des interfaces communes, ce qui facilitera l'ajout de nouveaux jeux et améliorera la flexibilité du système. Par exemple, une classe de base "Jeu" pourra être étendue pour créer des sous-classes spécifiques telles que "JeuExterieur" et "JeuInterieur", qui auront leurs propres fonctionnalités uniques. Le polymorphisme permettra ainsi au système de traiter tous les types de jeux de manière transparente, en utilisant des méthodes communes telles que "ajouterJeu" et "afficherJeux".

En plus de la gestion des jeux, le système offrira des fonctionnalités avancées telles que la gestion de la capacité maximale du parc, la gestion des réservations, la génération de rapports et de statistiques, et une interface utilisateur conviviale pour faciliter la navigation et l'utilisation du système. Par exemple, les responsables du parc pourront facilement ajouter de nouveaux jeux, vérifier la disponibilité des jeux, effectuer des réservations pour les visiteurs et générer des rapports détaillés sur l'utilisation des jeux.

En résumé, ce projet vise à développer un système complet et professionnel de gestion pour un parc de jeux, en utilisant l'héritage et le polymorphisme du langage C++. Ce système offrira des fonctionnalités avancées pour l'ajout, la gestion et l'affichage des jeux, tout en garantissant une gestion efficace, une flexibilité accrue et une expérience utilisateur optimale. Il permettra aux responsables du parc de gérer facilement les jeux, d'optimiser l'utilisation des ressources et de fournir une expérience divertissante et sécurisée pour les visiteurs du parc de jeux.

**Realisation**

**Class JeuInterieur{} :**

class JeuInterieur : public Jeu{

    private :

        int capaciteMax;

    public :

        JeuInterieur(string nom , int ageMinimum , int capaciteMax) : Jeu( nom ,  ageMinimum){

            this -> capaciteMax = capaciteMax;

        }

        void afficher() override {

            cout << "--------------------------" << endl;

            cout << "Nom du Jeu : " << nom << endl;

            cout << "Age minimum : " << ageMinimum << endl;

            cout << "capaciteMax : " << capaciteMax << endl;

            cout << "--------------------------" << endl;

        }

};

Ce code définit une autre classe dérivée appelée "JeuInterieur" qui hérite également de la classe de base "Jeu". La classe "JeuInterieur" ajoute un membre de données privé appelé "capaciteMax" de type entier.

La classe "JeuInterieur" possède un constructeur qui prend en paramètres une chaîne de caractères "nom", un entier "ageMinimum" et un entier "capaciteMax". Le constructeur utilise le constructeur de la classe de base "Jeu" pour initialiser les membres de données correspondants, puis initialise le membre de données "capaciteMax" avec la valeur fournie.

La classe "JeuInterieur" redéfinit également la méthode "afficher" héritée de la classe de base "Jeu". La méthode redéfinie affiche le nom du jeu, l'âge minimum requis, la capacité maximale du jeu en termes de nombre de participants, et une ligne de séparation.

En résumé, la classe "JeuInterieur" étend la classe de base "Jeu" en ajoutant un membre de données "capaciteMax" et en redéfinissant la méthode "afficher". Cela permet de spécifier et d'afficher des informations spécifiques aux jeux d'intérieur lorsqu'on utilise des objets de la classe "JeuInterieur".

**Class ParcJeux{} :**

class ParcJeux{

    private :

        vector<Jeu\*> jeux;

        int capaciteMax;

        // pas de capacite max car j'utilise un vecteur

    public :

        ParcJeux(int capaciteMax ){

            this -> jeux = jeux;

            this -> capaciteMax = capaciteMax;

        }

        ParcJeux(int capaciteMax , vector<Jeu\*> jeux){

            if(jeux.size() < capaciteMax){

                this -> jeux = jeux;

                this -> capaciteMax = capaciteMax;

            }

            else{

                cout << "CAPACITE MAX !!!!!! " << endl;

            }

        }

        void AjouterJeux(Jeu\* Jeu){

            if(jeux.size() < capaciteMax){

                jeux.push\_back(Jeu);

            }

            else{

                cout << "CAPACITE MAX !!!!!! " << endl;

            }

        }

        void afficherJeux(){

            for(Jeu\* jeu : jeux){

                jeu -> afficher();

            }

        }

};

Ce code définit une classe appelée "ParcJeux" qui représente un parc de jeux. La classe possède les membres de données privés suivants : un vecteur de pointeurs vers des objets de type "Jeu" appelé "jeux", et un entier "capaciteMax" qui représente la capacité maximale du parc en termes de nombre de jeux.

La classe "ParcJeux" a deux constructeurs. Le premier constructeur prend en paramètre un entier "capaciteMax" pour initialiser la capacité maximale du parc. Le deuxième constructeur prend en paramètres un entier "capaciteMax" et un vecteur de pointeurs vers des objets de type "Jeu" appelé "jeux". Ce deuxième constructeur vérifie si la taille du vecteur "jeux" est inférieure à la capacité maximale spécifiée. S'il l'est, il initialise les membres de données correspondants. Sinon, il affiche un message indiquant que la capacité maximale est atteinte.

La classe "ParcJeux" a une méthode nommée "AjouterJeux" qui prend en paramètre un pointeur vers un objet de type "Jeu". Cette méthode vérifie si la taille du vecteur "jeux" est inférieure à la capacité maximale du parc. Si c'est le cas, elle ajoute le jeu au vecteur. Sinon, elle affiche un message indiquant que la capacité maximale est atteinte.

La classe "ParcJeux" a également une méthode nommée "afficherJeux" qui parcourt le vecteur "jeux" et appelle la méthode "afficher" pour chaque jeu contenu dans le vecteur.

En résumé, la classe "ParcJeux" représente un parc de jeux avec une capacité maximale spécifiée. Elle permet d'ajouter des jeux au parc, en respectant la capacité maximale, et d'afficher les jeux disponibles dans le parc.

**Int Main(){} :**

int main(){

    Jeu TICTACTOE("TICTACTOE" , 3);

    JeuInterieur Scrable("Scrable" , 5 , 2);

    JeuInterieur Monopoly("Monopoly" , 5 , 5);

    ParcJeux Parc1(2);

    Parc1.AjouterJeux(&TICTACTOE);

    Parc1.AjouterJeux(&Scrable);

    Parc1.AjouterJeux(&Monopoly);

    Parc1.afficherJeux();

    return 0;

};

Ce code illustre l'utilisation des classes "Jeu", "JeuInterieur" et "ParcJeux".

Il crée quelques objets de jeux tels que "TICTACTOE", "Scrable" et "Monopoly" avec leurs caractéristiques respectives. Ensuite, un objet "Parc1" de la classe "ParcJeux" est créé avec une capacité maximale de 2.

Les jeux sont ajoutés au parc à l'aide de la méthode `AjouterJeux()`, et la méthode `afficherJeux()` est appelée pour afficher les informations des jeux présents dans le parc. Le code illustre ainsi comment créer un parc de jeux, ajouter des jeux à celui-ci et afficher les jeux disponibles.