Minivilles

-La Mini Présentation :)

Antoine

Malaak

Enki

Elliot

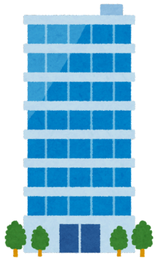
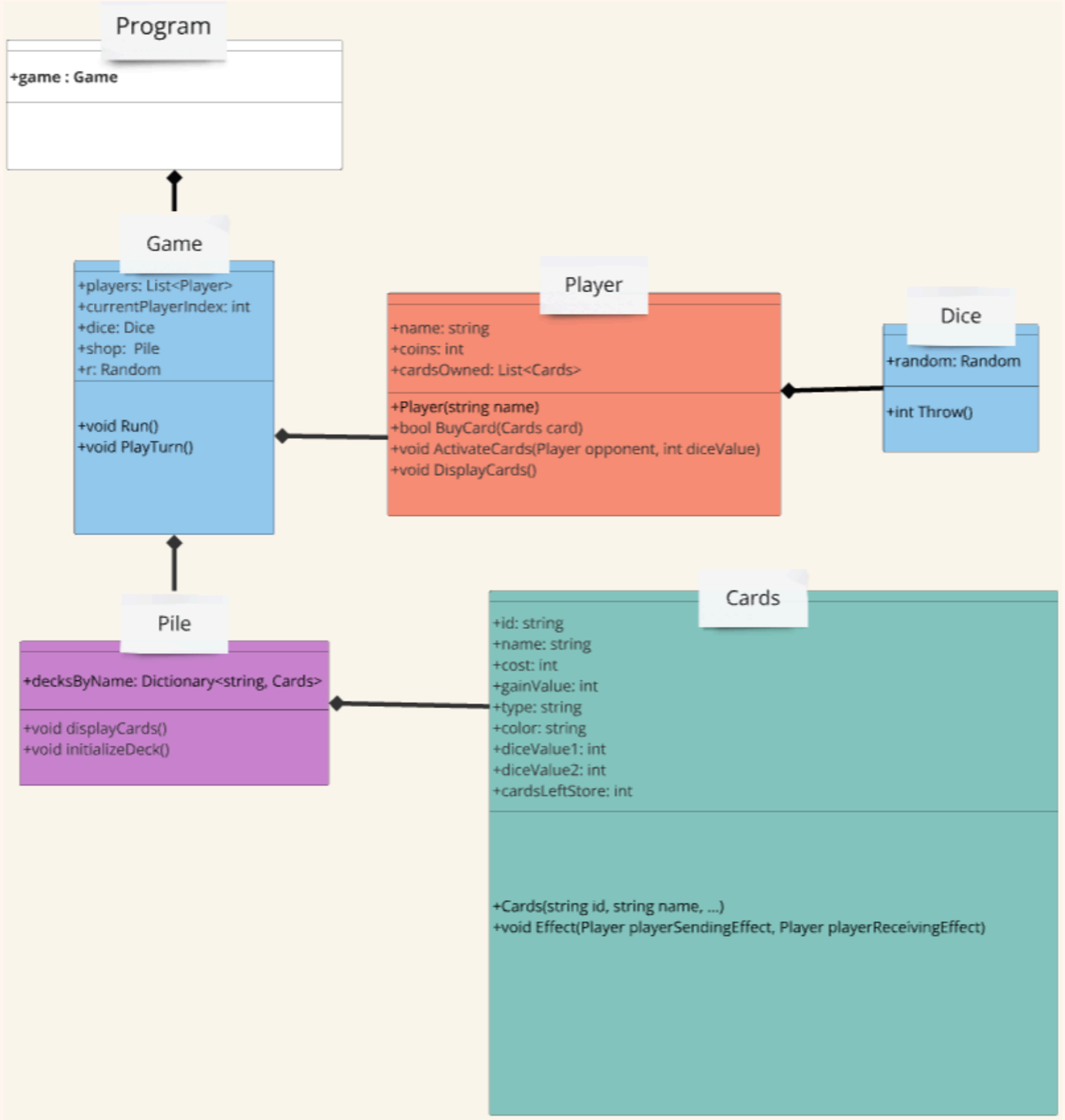
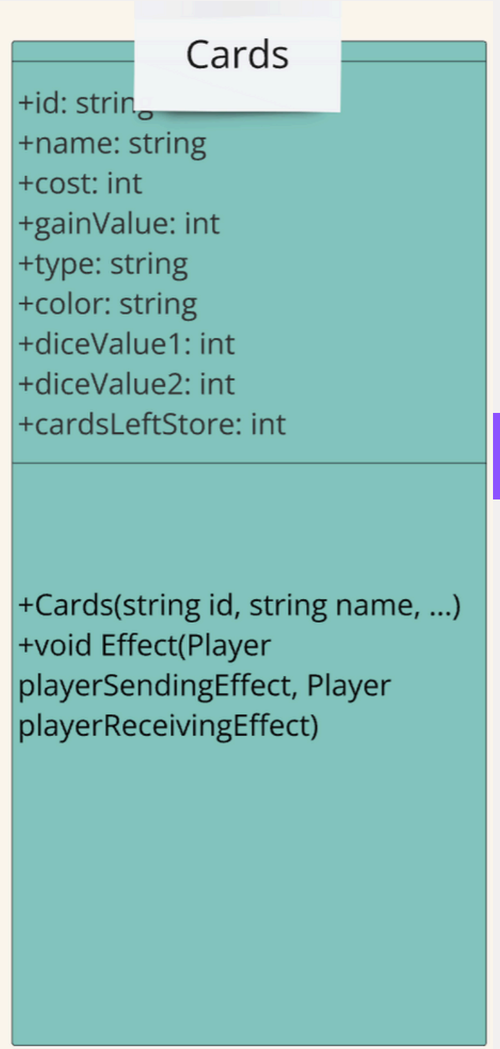


Diagramme de classe du jeu

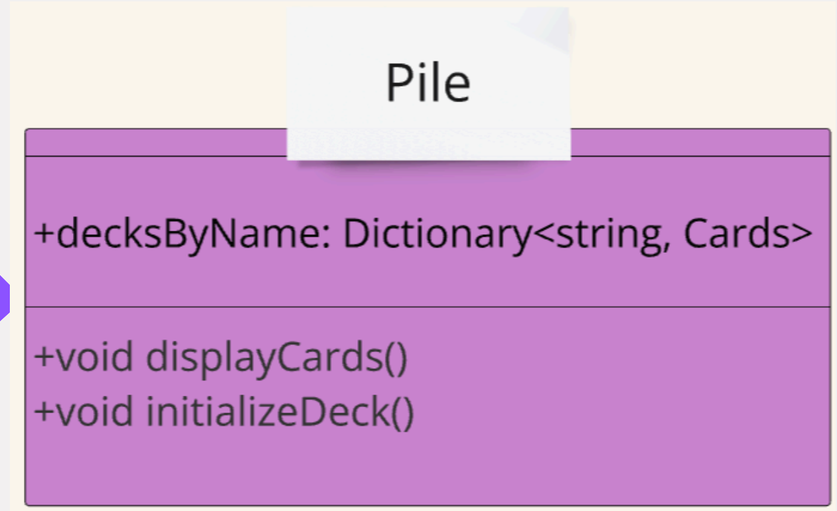


```
 fonction qui gère les effets des cartes hors monuments
ence
ic void Effect(Player playerSendingEffect, Player otherPlayer) //playerSendingEffect correspond au joueur qui lance les dés et playerReceivingEffect tous les autres

//initialisation de l'effet des cartes en fonction de leur couleur : on ajoute des pièces au joueur actuel parfois au détriment d'autres joueurs
if (this.Color == "blue")
{

    playerSendingEffect.Coins += this.GainValue;
    Console.ForegroundColor = ConsoleColor.Blue;
    Console.WriteLine($"\n {playerSendingEffect.Name} reçoit {this.GainValue} pièce(s) de {this.Name}.");
    Console.ForegroundColor = ConsoleColor.White;

}
if (this.Color == "red")
{
    playerSendingEffect.Coins += this.GainValue;
    otherPlayer.Coins -= this.GainValue;
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine($"\n {playerSendingEffect.Name} reçoit {this.GainValue} pièce(s) de {otherPlayer.Name} par le biais de {this.Name}.");
    Console.ForegroundColor = ConsoleColor.White;

}
if (this.Color == "green")
{
    if (this.Type == "shop")
    {
        playerSendingEffect.Coins += this.GainValue;
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine($"\n {playerSendingEffect.Name} reçoit {this.GainValue} pièce(s) de {this.Name}.");
        Console.ForegroundColor = ConsoleColor.White;
    }
    else //cas spécial des cartes vertes qui ajoutent un nombre de pièces spécifiques pour certains types de cartes
    {
        int gain = 0;
        string researched = "";
        switch (this.Id) //condition sur le nom des cartes vertes spéciales
        {
            case "FR":
                researched = "breeding";
                break;
            case "FM":
                researched = "natural";
                break;
            case "MA":
                researched = "harvest";
                break;
        }
        foreach (var card in playerSendingEffect.CardsOwned) //on parcourt la main du joueur qui vient de lancer les dés
        {
            if (card.Type == researched) //si le joueur a des cartes au type affecté, on ajoute pour chacune d'entre elle le montant du gain de la carte verte spécial
            {
                gain += this.GainValue;
            }
        }
        playerSendingEffect.Coins += gain;
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine($"\n {playerSendingEffect.Name} reçoit {gain} pièce(s) de {this.Name}.");
        Console.ForegroundColor = ConsoleColor.White;
    }
```

Classe Cards



Classe Pile




```
public void DisplayCards()
{
    Console.ForegroundColor = ConsoleColor. Magenta;
    Console.WriteLine("Achetez une carte en saisissant la commande associée:");
    foreach (var cardType :string  in DecksByName.Keys)
    {
        Cards card = DecksByName[cardType];
        Console.Write($"-'{cardType}': Achète un.e {card.Name} pour {card.Cost} pièce(s) [{(card.Color == "green"? "Durant votre tour uniquement" : "Peu importe le tour")}");
        Console.Write($", {(card.Color == "red"? "volez" : "gagnez")} {card.GainValue} pièce(s)");
        switch (card.Id)
        {
            case "FR":
                Console.Write(" pour chaque ferme que vous possédez");
                break;
            case "FM":
                Console.Write(" pour chaque forêt et mine que vous possédez");
                break;
            case "MA":
                Console.Write(" pour chaque champs de blé et verger que vous possédez");
                break;
        }
        Console.WriteLine($" sur un {card.DiceValue1}{(card.DiceValue2 != 0? $" ou un {card.DiceValue2}" : "")}] ({card.CardsLeftStore} restants)");
    }
    Console.WriteLine("-'/': Passer la phase d'achat");
    Console.ForegroundColor= ConsoleColor. White;
}
```

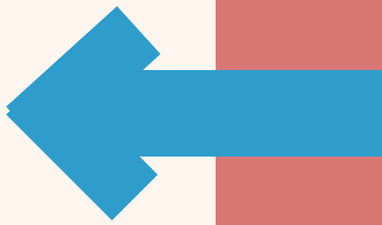
```
private void InitializeDeck()
{
    List<Cards> allCards = new List<Cards>
    {
        new Cards("CB","champs de blé", 1, 1, "harvest", "blue", 1, 0),
        new Cards("FO","forêt", 3, 1, "natural", "blue", 5, 0),
        new Cards("MI","mine", 6, 5, "natural", "blue", 9, 0),
        new Cards("FE","ferme", 2, 1, "breeding", "blue", 2, 0),
        new Cards("VE","verger", 3, 3, "harvest", "blue", 10, 0),
        new Cards("BO","boulangerie", 1, 1, "shop", "green", 2, 3),
        new Cards("SU","supérette", 2, 3, "shop", "green", 4, 0),
        new Cards("FR","fromagerie", 5, 3, "factory", "green", 7, 0),
        new Cards("FM","fabrique de meubles", 3, 3, "factory", "green", 8, 0),
        new Cards("MA","marché", 5, 2, "grocery", "green", 11, 12),
        new Cards("CA","café", 2, 1, "restaurant", "red", 3, 0),
        new Cards("RE","restaurant", 3, 2, "restaurant", "red", 9, 10)
        /*new Cards("BC","business center", 8, 0, "establishment", "purple", 6, 0, 6), //EXCEPTION : échar
        new Cards("TE","television", 7, 5, "establishment", "purple", 6, 0, 6),
        new Cards("ST","stadium", 6, 2, "establishment", "purple", 6, 0,6),*/
    };

    // Regrouper les cartes par couleur
    foreach (Cards card in allCards)
        DecksByName[card.Id] = card;
}
```

Classe Player

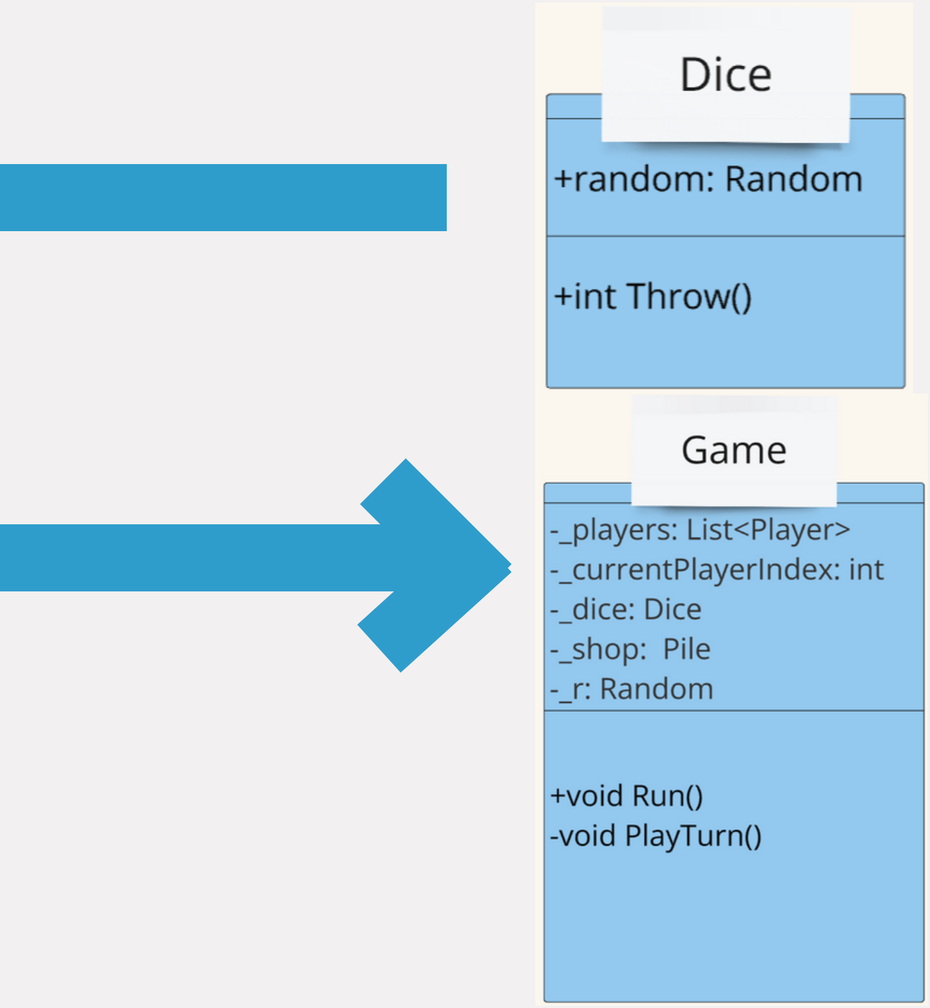

```
public class Player
{
    //Attributs de la classe player
    public readonly string Name;                //Nom du joueur
    public int Coins;                   //Nombre de pieces que le joueur possède
    public bool CanRollTwoDice;         //Indique si le joueur peut lancer 2 dés
    public bool CanReroll;              //Indique si le joueur peut relancer ses dés
    public readonly List<Cards> CardsOwned;      //Liste des cartes possédées par le joueur

    2 références
    public Player(string name)
    {
        this.Name = name;   //Nom du joueur
        Coins = 3;          //La partie commence avec 3 pièces
        //Liste des cartes avec 2 cartes attribuées en début de partie
        CardsOwned =
        [
            new Cards("CB", "champs de blé", 1, 1, "harvest", "blue", 1, 0),
            new Cards("BO", "boulangerie", 1, 1, "shop", "green", 2, 3)
        ];
    }
```

```
public bool BuyCard(Cards card)
{
    if (Coins >= card.Cost && card.CardsLeftStore > 0) //Vérifie si le joueur peut acheter sa carte
    {
        Coins -= card.Cost;          //Retire le coût en pieces de la carte du porte monnaie du joueur
        card.CardsLeftStore--;       //Réduit le stock de la carte
        CardsOwned.Add(card);        //Ajoute la carte pour le joueur

        Console.WriteLine($"{Name} a acheté {card.Name}. Il en reste {card.CardsLeftStore}.");
        return true;                 //Achat réussi
    }
    else
    {
        //On gère les cas où l'achat n'est pas possible
        if (Coins < card.Cost)
            Console.WriteLine($"{Name} n'a pas assez de pièces pour acheter {card.Name}.");
        else if (card.CardsLeftStore <= 0)
            Console.WriteLine($"{card.Name} est en rupture de stock.");

        return false;
    }
}
```

2 références
```
public void ActivateCards(Player opponent, int diceValue, bool hasItsTurn)
{
    foreach (var card in CardsOwned)    //Parcours les cartes du joueur
    {
        //Vérifie si la carte est activées par le dé
        if (card.DiceValue1 == diceValue || card.DiceValue2 == diceValue)
        {
            //Si la carte est bleue ou verte son effet se déclenche
            if (card.Color == "blue" || (card.Color == "green" && hasItsTurn))
                card.Effect(this, opponent);
        }
    }
}
```



1 référence
```
public void DisplayCards()
{
    //Montre au joueur toutes les cartes qu'il possède
    foreach (var card in CardsOwned)
    {
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine($"- {card.Name} ({card.Color}) : {card.GainValue} gain(s) [Activation : {card.DiceValue1}, {card.DiceValue2}]");
        Console.ForegroundColor = ConsoleColor.White;
    }
}
```




```
internal class Dice
{
    private readonly Random _random = new();

    1 référence
    public int Throw()
    {
        return _random.Next(1,7);
    }
}
```

```
nternal class Game

    private readonly List<Player> _players = new();
    private int _currentPlayerIndex;
    private readonly Dice _dice = new Dice();
    private readonly Random _r = new Random();
    private readonly Pile _shop = new Pile();
```

```
public void Run()
{
    //initialize all players
    _players.Add(new Player("Player"));
    _players.Add(new Player("AI"));

    //game loop
    while (_players[0].Coins < 20 && _players[1].Coins < 20)
    {
        Console.Clear();
        Console.WriteLine("___  ____       _       _ _ _                ");
        Console.WriteLine(@"|  \\/ (_)     (_)     (_) | |               ");
        Console.WriteLine("| .  . |_ _ __  ___   ___| | | ___  ___ ");
        Console.WriteLine(@"| |\/| | | '_ \| \ \ / / | | |/ _ \/ __|");
        Console.WriteLine(@"| |  | | | | | | |\ V /| | | |  __/\__ \");
        Console.WriteLine(@"\_|  |_/_|_| |_|_| \_/ |_|_|_|\___||___/");
        Console.WriteLine();
        PlayTurn(_players[_currentPlayerIndex]);
        _currentPlayerIndex = (_currentPlayerIndex == 1) ? 0 : 1;
    }
    //display the end of the game
    foreach (var player in _players)
    {
        if(player.Coins >= 20)
        {
            Console.WriteLine($"Joueur {player.Name} a gagné en atteignant 20 pièces!");
        }
    }
}
```

```
private void PlayTurn(Player ActivePlayer)
{
    Player activePlayer = ActivePlayer;
    Player otherPlayer = (_currentPlayerIndex == 1) ? _players[0] : _players[1];

    //display whose playing and which cards he's handling
    Console.WriteLine($"C'est au tour de {activePlayer.Name}. Il détient {activePlayer.Coins} pièce(s) et ses cartes sont: ");
    Thread.Sleep(1000);
    activePlayer.DisplayCards();
    Thread.Sleep(1000);

    int diceRoll = 0;
    int n;
    //AI decides if she wants to roll 1 or 2 dices
    if (activePlayer.Name == "AI")
    {
        n = _r.Next(1,3);
    }
    //player decides if he wants to roll 2 dices
    else
    {
        Console.WriteLine("Voulez-vous lancer 1 ou 2 dés? (1/2)");
        while(!int.TryParse(Console.ReadLine(), out n))
        {
            Console.WriteLine("Saisie invalide. Veuillez réessayer.");
        }
    }

    //dices being rolled
    for (int i = 1; i <= n; i++)
    {
        diceRoll += _dice.Throw();
    }
    Console.WriteLine($"{activePlayer.Name} a obtenu un {diceRoll}");
    Thread.Sleep(500);

    //activate the card's effects based on the dices
    otherPlayer.ActivateCards(activePlayer, diceRoll, false);
    activePlayer.ActivateCards(otherPlayer, diceRoll, true);

    Console.WriteLine("Il est temps d'acheter une carte!");
    Thread.Sleep(100);

    //display shop and ask the player to buy
    _shop.DisplayCards();
    string choice = PurchaseChoice(activePlayer);
    if (choice != "/")
    {
        while (!activePlayer.BuyCard(_shop.DecksByName[choice]))
        {
            choice = PurchaseChoice(activePlayer);
            if (choice == "/")
            {
                break;
            }
        }
    }

    // wait for the player to read
    Console.WriteLine("Appuyez sur Entrée pour continuer.");
    Console.ReadLine();
}
```

```
private string PurchaseChoice(Player activePlayer)
{
    string choice;
    if (activePlayer.Name == "AI")
    {
        choice = _shop.DecksByName.ElementAt(_r.Next(0, _shop.DecksByName.Count - 1)).Key;
        if (activePlayer.Coins == 0)
        {
            choice = "/";
        }
    }
    else
    {
        choice = Console.ReadLine();
        while (!_shop.DecksByName.Keys.Contains(choice) && choice != "/")
        {
            Console.WriteLine("Mauvaise saisie. Veuillez réessayer");
            choice = Console.ReadLine();
        }
    }
    return choice;
}
```

Difficultés?
Evolution et choix de la structure du jeu?