DOCUMENTATION Poker

Le projet Poker n'utilise pas de POO, mais des structures :

```
// Codes COULEUR
public enum couleur { VERT = 10, ROUGE = 12, JAUNE = 14, BLANC = 15, NOIRE = 0,
    ROUGESURBLANC = 252, NOIRESURBLANC = 240 };

// Coordonnées pour l'affichage
public struct coordonnees
{
    public int x;
    public int y;
}

// Une carte
public struct carte
{
    public char valeur;
    public int famille;
};
```

```
// Liste des combinaisons possibles
public enum combinaison { RIEN, PAIRE, DOUBLE_PAIRE, BRELAN, QUINTE, FULL, COULEUR, CARRE, QUINTE_FLUSH };
```

```
// Coordonnées de départ pour l'affichage
public static coordonnees depart;

// Fin du jeu
public static bool fin = false;

// Valeurs des cartes : As, Roi,...
public static char [] valeurs = {'A','R','D','V','X','9','8','7'};

// Codes ASCII (3 : coeur, 4 : carreau, 5 : trèfle, 6 : pique)
public static int [] familles = {3,4,5,6};

// Numéros des cartes à échanger
public static int [] echange = {0,0,0,0};

// Jeu de 5 cartes
public static carte[] MonJeu = new carte[5];
```

En étant un projet console, une gestion de l'affichage est codé :

```
private static void afficher_message(string message, couleur lacouleur, coordonnees c)
{
    // Se placer
    Console.SetCursorPosition(depart.x + c.x, depart.y + c.y);

    // Définir la couleur
    SetConsoleTextAttribute(hConsole, (int)lacouleur);

    // Afficher le message
    Console.Write(message);

    // Remise de la couleur par défaut
    SetConsoleTextAttribute(hConsole, (int)couleur.BLANC);
}
```

```
private static char question(string texte, coordonnees c, int deplacement_sur_x)
{
    afficher_message(texte, couleur.JAUNE, c);
    afficher_message("", couleur.BLANC, new coordonnees { x = c.x + deplacement_sur_x, y = c.y });
    char reponse = (char)_getch();
    afficher_message(new string(new char[] { reponse }), couleur.BLANC,
        new coordonnees { x = c.x + deplacement_sur_x, y = c.y });
    return reponse;
}
```

Le projet se sépare en plusieurs partie :

```
public static void jouerAuPoker(carte [] leJeu, int [] ech)
{
    char reponse;
    while (true)
    {
        Console.Clear();

        // TIRAGE D'UN JEU DE 5 CARTES
        tirageDuJeu(leJeu);

        // ECHANGE DE CARTES
        echangeDeCartes(leJeu, ech);

        // CALCUL ET AFFICHAGE DU RESULTAT DU JEU
        afficheResultat(leJeu);

        // NOUVEAU TIRAGE ?
        reponse = question("Une nouvelle main ? (O/N) ", new coordonnees { x = 1, y = 17 }, 27);
        if (reponse == 'n' || reponse == 'N')
            break;
    }

    // ENREGISTRER LE JEU ?
    reponse = question("Enregistrer le jeu ? (O/N) ", new coordonnees { x = 1, y = 17 }, 27);
    if (reponse == 'o' || reponse == 'O')
        enregistrerJeu(leJeu);
}
```

Le tirage de toutes les cartes et un fonction utilisant plusieurs fois le tirage d'une carte :

```
private static void tirageDuJeu(carte[] unJeu)
{
    for (int i = 0; i < unJeu.Length; i++)
    {
        unJeu[i] = tirage();
        for (; !carteUnique(unJeu[i], unJeu, i);)
        {
            unJeu[i] = tirage();
            if (carteUnique(unJeu[i], unJeu, i) == true)
            {
                affichageCarte(unJeu[i], i);
            }

            else
                unJeu[i] = tirage();
        }
        affichageCarte(unJeu[i], i);


    }
}
```

```
private static carte tirage()
{
    // La carte à générer
    carte uneCarte;

    Random rnd = new Random();
    int rndfam = rnd.Next(0, 8);
    uneCarte.valeur = valeurs[rndfam];

    Random rnd2 = new Random();
    int rndval = rnd2.Next(0, 4);
    uneCarte.famille = familles[rndval];

    return uneCarte;
}
```

Afin de prévenir le cas où une même carte soit tiré, une fonction vérifie l'intégrité :

```
private static bool carteUnique(carte uneCarte, carte[] unJeu, int numero)
{
    for (int i = 0; i < unJeu.Length; i++)
    {
        if (numero != i)
        {
            if (uneCarte.famille == unJeu[i].famille && uneCarte.valeur == unJeu[i].valeur)
            {
                return false;
            }
        }
    }

    return true;
}
```

Ensuite, on peut échanger les cartes, on ne fait que réutiliser la fonction tirage, et on prévoit aussi de ne pas retirer une ancienne carte, on efface ensuite toute la console puis on réaffiche les cartes :

```
private static void echangeDeCartes(carte[] unJeu, int[] e)...

// Fonction clear_console
// Pour clear la console et réafficher les cartes
private static void Clear_console(carte [] unJeu)...
```

Afficher le résultat :

```
private static void afficheResultat(carte [] unJeu)
{
    afficher_message("RESULTAT - Vous avez : ", couleur.ROUGE, new coordonnees { x = 1, y = 15 });

    // Test de la combinaison
    switch (cherche_combinaison(unJeu))
    {
        case combinaison.RIEN :
            afficher_message("rien du tout... desole!", couleur.ROUGE, new coordonnees { x = 24, y = 15 });
            break;

        case combinaison.PAIRE:
            afficher_message("la même que t'es baloche, une paire", couleur.ROUGE, new coordonnees { x = 24, y = 15
            break;

        case combinaison.DOUBLE_PAIRE:
            afficher_message("Félicitation, tu as deux paires", couleur.ROUGE, new coordonnees { x = 24, y = 15 });
            break;

        case combinaison.BRELAN:
```

Connaître le résultat des 5 cartes tiré :

```
private static combinaison cherche_combinaison(carte[] unJeu)
{
    int i, j, nbpaires = 0, nb;

    // Nombre de valeurs similaires dans le jeu pour chaque carte
    int[] similaire = { 0, 0, 0, 0, 0 };

    // Booléens : si paire ET brelan alors on a un FULL
    bool paire = false;
    bool brelan = false;

    // Possibilités de quinte. Tableau 4*5
    char[,] quintes = {    {'X','V','D','R','A'},
                           {'9','X','V','D','R'},
                           {'8','9','X','V','D'},
                           {'7','8','9','X','V'}
                      };

    // Résultat à renvoyer
    combinaison resultat;

    // Par défaut : aucun jeu
    resultat = combinaison.RIEN;
```

Enfin on peut enregistré le score dans un fichier .txt :

```
private static void enregistrerJeu(carte [] unJeu)
{
    string cheminrep = "Scores_Jimmy.txt";
    string UneLigne = null;
    StreamWriter LeFichier = null;
    if ( !File.Exists(cheminrep) )
        LeFichier = new StreamWriter(cheminrep);
    else
        LeFichier = File.AppendText(cheminrep);
    Clear_console(unJeu);
    afficher_message("Entrez votre pseudo : ", couleur.VERT, new coordonnees { x = 1, y = 15 });
    SetConsoleTextAttribute(hConsole, 2);
    UneLigne += Console.In.ReadLine();
    UneLigne += ("/" + cherche_combinaison(unJeu) + "\\");
    for (int i = 0; i < unJeu.Length; i++)
    {
        char valeur = unJeu[i].valeur;
        int famille = unJeu[i].famille;
        UneLigne += famille;
        UneLigne += valeur;
        if ( i != unJeu.Length - 1 )
            UneLigne += "-";
    }
    UneLigne += "|";
    SetConsoleTextAttribute(hConsole, 15);
```

```
UneLigne += "|";
SetConsoleTextAttribute(hConsole, 15);
try
{
    if (UneLigne != null)
    {
        LeFichier.Write(UneLigne);
    }
}
catch (Exception e)
{ System.Console.Error.WriteLine("Erreur suivante :" + e);}
finally
{
    try
    { LeFichier.Close(); }
    catch { }
}
```

Par la suite on peut lire ce fichier .txt :

```
public static void voirScores()
{
    string cheminrep = "Scores_Jimmy.txt";
    if (!File.Exists(cheminrep))
    {
        afficher_message("-----------------------------------", couleur.ROUGE, new coordonnees { x = 1, y = 4 });
        afficher_message("Le dossier de score n'existe pas /!\\", couleur.ROUGE, new coordonnees { x = 1, y = 5 });
        afficher_message("-----------------------------------", couleur.ROUGE, new coordonnees { x = 1, y = 6 });
    }
    else
    {
        StreamReader LeFichier = new StreamReader(cheminrep);
        string UneLigne = LeFichier.ReadLine();
        char[] score = UneLigne.ToCharArray();
        carte[] unJeu = new carte[5];
        int ns = 0;
        for (int i = 0; i < score.Length; i++)...

        string[] ts = new string[ns];                   // Tableau string , avec scores séparer
        for (int i = 0, j = 0; i < ts.Length; i++)...

        for (int i = 0; i < ns ; i++)...
    }
}
```