AP 3
***
La société immobilière HsH (Home Sweet Home)

Présentation

La société HsH est l'un des premiers réseaux immobilier français.

- Crée en 1984, HsH est aujourd'hui le troisième réseau d'agences immobilières en France.
- Plus de 1 000 points de ventes répartis sur tout le territoire, une implantation et des moyens qui favorisent les transactions grâce aux 5 500 collaborateurs.
- Le réseau HsH bénéficie d'un atout unique dans le secteur de l'immobilier : son fichier commun de plus de 100 000 biens.
- Un total de 5 500 collaborateurs et plus de 50 000 transactions par an.
- Plus de 60 GIE locaux (Groupement d'Intérêt Economique).
- Plus de 100 000 annonces sur www.HsH-immo.com.
- Un site central et des sites locaux.

La société HsH vous est présentée dans les documents du dossier CONTEXTE-HSH (cf. dossier commun sur le réseau).

Le service Direction des Systèmes d'Information de HsH

La majeure partie des développements, la supervision et maintenance des réseaux se fait par le biais d'une équipe interne. Cependant, M. Livio MIGLIOLI, Directeur des systèmes d'information, souhaite sous-traiter certaines prestations auprès des SSII partenaires, lorsque les équipes internes sont en surcharge et ne peuvent pas prendre en charge de nouveaux projets.

C'est le cas de SIVY qui se voit confier régulièrement des missions.

MISSION 2

Diagnostic et maintenance corrective pour un ensemble de jeux développés pour le Comité d'entreprise du siège social de HsH.

Introduction

Dans les entreprises d'au moins 50 salariés, l'employeur est tenu d'organiser la mise en place d'un comité d'entreprise composé :

- de représentants élus du personnel,
- de représentants syndicaux désignés par les organisations syndicales.

Ce comité assume d'une part, des attributions économiques et d'autre part, sociales et culturelles et dispose pour ce faire, des moyens matériels et financiers nécessaires. L'employeur (ou son représentant) assure les fonctions de président du CE.

Financement et moyens

Le CE de HsH dispose de deux subventions distinctes :

- la subvention de fonctionnement, versée tous les ans et correspondant à une pourcentage de la masse salariale brute;
- la contribution aux activités sociales et culturelles.

L'employeur met à la disposition du CE un local aménagé et le matériel nécessaire à son fonctionnement (téléphone, mobilier, photocopie…).

Attributions

Le Code du travail prévoit des CONSULTATIONS ANNUELLES (aménagement du temps de travail, bilan social, congés payés, égalité professionnelle, évolution des emplois et des qualifications, formation professionnelle, recherche) ainsi que des consultations ponctuelles (règlement intérieur, introduction de nouvelles technologies, organisation de l'entreprise…).

Le CE assure, organise et développe, en faveur des salariés de l'entreprise, des ACTIVITES SOCIALES ET CULTURELLES. Il peut s'agir notamment :

- de la prise en charge de tout ou partie d'une mutuelle de santé,
- de la prise en charge de tout ou partie d'une cantine,
- d'activités sportives ou de loisirs (colonies de vacance, séjours...),
- d'activités culturelles (bibliothèques, tarifs préférentiels pour des spectacles ou des musées...),
- etc…

Dans ce contexte, le Comité d'entreprise de HSH souhaite offrir un ESPACE DETENTE destiné au personnel de HsH et à leurs ENFANTS, permettant :

- l'accès Internet aux personnes qui ne possèdent pas d'ordinateur portable Wifi ou de Smartphones pour surfer sur le net avec le WIFI gratuit ;
- l'utilisation de JEUX INFORMATIQUES sur des ordinateurs sous Windows. Certains de ces jeux ont été développés par la Direction des Systèmes d'Information de HsH, autour de la plateforme VISUAL STUDIO, Framework .NET et langage C#.

Le besoin exprimé

Il est apparu que certains jeux ne marchent pas (ou mal) et présentent des fonctionnalités trop sommaires. La plupart doivent faire l'objet d'une maintenance corrective et/ou ne fonctionnent pas de manière satisfaisante.

Les informaticiens de HsH sont actuellement occupés sur des projets internes ; c'est la raison pour laquelle, M. HALTER, responsable du Département « Gestion des applications » fait appel à la SSII « SIVY » afin d'assurer la maintenance corrective et/ou évolutive d'un ensemble de jeux, parmi lesquels le « Morpion », le « Mastermind » et le « Poker ».

Eléments à prendre en compte

- Dans la page suivante, un CAHIER DE CHARGES récapitule le fonctionnement attendu de chacun des jeux, ainsi que des éléments techniques.

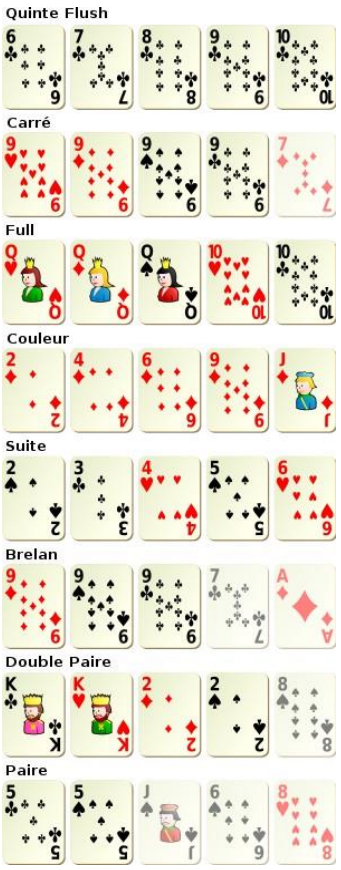
Echéances à respecter

1) Le POKER est à rendre au plus tard le MERCREDI 15/01/2025 à 15h30.
2) Vous ferez une documentions dans un README.md du fonctionnement du poker.

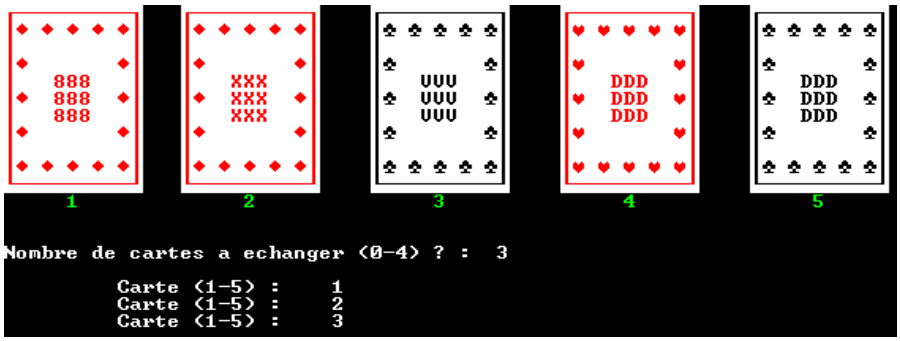
Le POKER

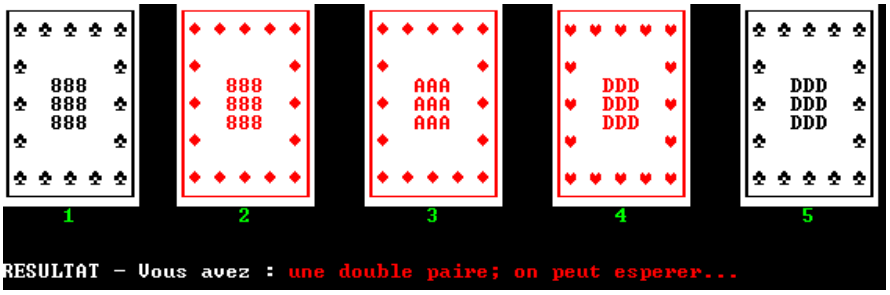
Le programme à développer s'inspire du jeu du Poker. Un tirage aléatoire de 5 cartes est effectué. Il faudra s'assurer, bien sûr, que chaque carte est unique, et ensuite, l'utilisateur aura la possibilité, soit de conserver son jeu, soit d'échanger aux plus quatre cartes, et obtenir ainsi une nouvelle main.

Le logiciel doit ensuite calculer et afficher le résultat parmi les combinaisons suivantes :



Le résultat à obtenir est le suivant :





Éléments techniques

```
// ----------------------------------------
// DECLARATION DES DONNEES
// ----------------------------------------

// Importation des DL (librairies de code) permettant de gérer les couleurs en mode console
[DllImport("kernel32.dll")]
public static extern bool SetConsoleTextAttribute(IntPtr hConsoleOutput, int wAttributes);
[DllImport("kernel32.dll")]
public static extern IntPtr GetStdHandle(uint nStdHandle);

static uint STD_OUTPUT_HANDLE = 0xfffffff5;
static IntPtr hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

// Pour utiliser la fonction C 'getchar()' : sasie d'un caractère
[DllImport("msvcrt")]
static extern int _getch();

// *** TYPES DE DONNEES

// 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};
```

Pour gérer les couleurs :

- Les DLL correspondantes ont été incluses.
- Des constantes correspondantes aux codes couleurs ont été déclarées.

L'instruction suivante permet d'afficher les différentes combinaisons de couleurs :

```
for (int k = 1; k < 255; k++)
{
        SetConsoleTextAttribute(hConsole, k);
        Console.WriteLine("{0:d3} I want to be nice today!",k);
}

SetConsoleTextAttribute(hConsole, 236);
```

```
// *** VARIABLES

// 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];
```

REMARQUE : le TYPE ENUMERE

Un type énuméré est un type de données (généralement défini par l'utilisateur) qui consiste en un ensemble de CONSTANTES appelées énumérateurs. Lorsque l'on crée un type énuméré on définit ainsi une « énumération ». Lorsqu'une variable est déclarée comme étant de type énuméré, cette variable peut recevoir n'importe quel énumérateur comme valeur.

```
// 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:
…
```

Fonctions à compléter

```
// Génère aléatoirement une carte : {valeur;famille}
// Retourne une expression de type "carte"
private static carte tirage()

// Indique si une carte est déjà présente dans le jeu
// Paramètres : une carte, le jeu 5 cartes, le numéro de la carte dans le jeu
// Retourne un booléen
private static bool carteUnique(carte uneCarte, carte[] unJeu, int numero)

// Affiche à l'écran une carte {valeur;famille}
```

```
private static void affichageCarte(carte uneCarte)

// Tirage d'un jeu de 5 cartes
// Paramètre : le tableau de 5 cartes à remplir
private static void tirageDuJeu(carte[] unJeu)

// Echange des cartes
// Paramètres : le tableau de 5 cartes et le tableau des numéros des cartes à échanger
private static void echangeDeCartes(carte[] unJeu, int[] e)

// Calcule et retourne la combinaison (paire, double-paire…) pour un jeu complet de 5 cartes
// La valeur retournée est un élement de l'énumération 'combinaison'
private static combinaison chercheCombinaison(carte[] unJeu)

// Calcul et affichage du résultat
// Paramètre : le tableau de 5 cartes
private static void afficheResultat(carte [] unJeu)

// Enregistrer le jeu dans un fichier
private static void enregistrerJeu(carte [] unJeu)

// Voir les scores depuis le fichier
private static void voirScores()

// Jouer au poker
public static void jouerAuPoker(carte [] leJeu)

// Affiche le menu du jeu et retourne l'option choisie
public static char afficherMenu()
```

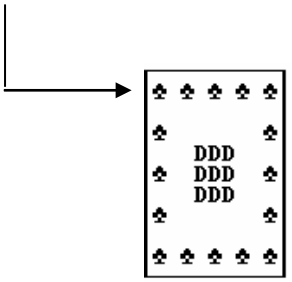
Programme principal

Les étapes suivantes sont nécessaires :

a) Tirage aléatoire de 5 cartes. Pour chaque carte :
   - Tirage de la carte n°i (le jeu doit être sans doublons).
   - Affichage de la carte.

b) Demande d'ECHANGE de cartes.
c) Calcul et affichage du RESULTAT du jeu.

Affichage de la carte depuis le haut :



Principe d'affichage d'une carte :

```
// Choix de la couleur
//     * couleur.ROUGE pour coeur et carreau
//     * couleur.NOIRE pour trèfle et pique
couleur lacouleur;
if (uneCarte.famille == 3 ||  uneCarte.famille == 4)
       lacouleur = couleur.ROUGESURBLANC;
else
       lacouleur = couleur.NOIRESURBLANC;

// Positionnement et affichage
// REMARQUE : dans la fenêtre exécution, choisir "Police Roster" pour afficher les symboles...
afficher_message
(
string.Format("{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}\n",'*','-','-','-','-','-','-','-','-','-','*'),
lacouleur,
new coordonnees {x= colonne * 15, y=1 }
);
afficher_message
(
string.Format("{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}\n",'|',
(char)uneCarte.famille,' ',
(char)uneCarte.famille,' ',
(char)uneCarte.famille,' ',
```

```
(char)uneCarte.famille,' ',
(char)uneCarte.famille, '|'), lacouleur, new coordonnees { x = colonne * 15, y = 2 }
);
```

Etc…

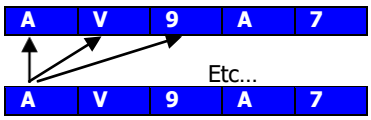
Principe de calcul du résultat

Il s'agit de calculer la COMBINAISON (paire, double paire..., quinte flush) pour un jeu complet de 5 cartes. La valeur retournée est un élément de l'énumération 'combinaison' (=constante). Deux tableaux peuvent être utilisés :

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

// 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'}
                   };
```

Soit le jeu suivant :



On compte, pour chaque carte, le nombre de fois où sa valeur apparaît dans le jeu.

Ce qui permet de remplir le TABLEAU « similaire ».

PAIRE : S'il existe un élément du tableau « similaire » égal à 2, on a une paire.

DOUBLE PAIRE : Chaque fois que l'on a une paire on incrémente un compteur.
Si ce compteur / 2 = 2 alors on a une double paire…

CARRE : Il existe un élément du tableau « similaire » égal à 4.

BRELAN : Il existe un élément du tableau « similaire » égal à 3.

QUINTE : Tous les éléments du tableau « similaire » doivent être 1.
Ensuite, il faut comparer chaque carte avec CHAQUE POSSIBILITE DE QUINTE (chaque élément du tableau « quintes ». Si égalité, on incrémente un compteur. Lorsque ce compteur est égal à 5, alors on a une quinte…

QUINTE FLUSH : Il faut avoir une quinte et toutes les cartes de la MEME FAMILLE…

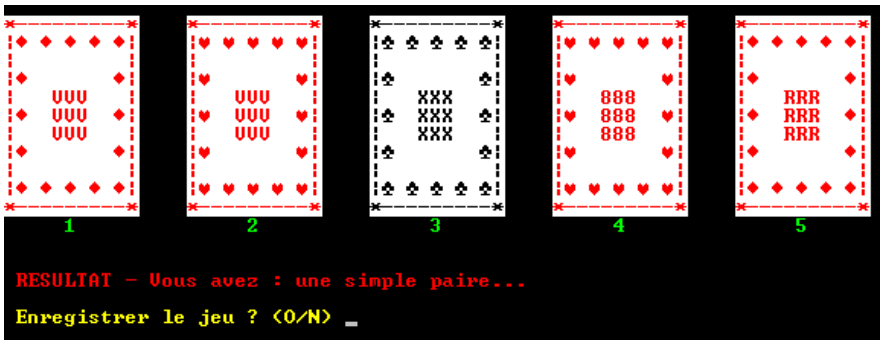
FULL : Il faut avoir à la fois un BRELAN et une PAIRE.

COULEUR : Les 5 cartes doivent être de la MEME FAILLE, mais sans constituer une QUNTE…

Évolution à prévoir : GESTION DES SCORES

Après avoir affiché le résultat du jeu, il est souhaitable de proposer à l'utilisateur d'ENREGISTER LE JEU OBTENU DANS UN FICHIER sur disque. Une fois le nom (ou pseudo) saisi, un article supplémentaire sera ajouté au fichier scores.

Exemple d'interface :



Un jeu doit être enregistré dans le fichier sous la forme d'un nouvel ENREGISTREMENT. Les valeurs du PSEUDO + JEU [= (famille carte + valeur carte) * 5] vont formater un objet chaîne « string » qui sera ensuite écrite dans le fichier.

Important : Vous allez devoir encrypter le fichier.

```
string nom, ligne;
BinaryWriter f;     // Variable FICHIER

// Ouverture du fichier en AJOUT
// Si le fichier EXISTE : ajout à la fin sinon création du fichier
f = new BinaryWriter(new FileStream("scores.txt", FileMode.Append, FileAccess.Write));
```

Etc…

Il faudra également offrir la possibilité de CONSULTER les SCORES enregistrés dans le FICHIER :

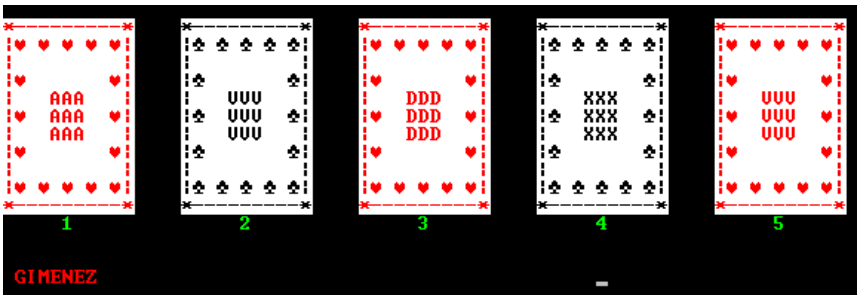
```
string article;         // Article à écrire dansle fichier
BinaryReader f;         // Variable FICHIER

// Caractères délimiteurs des champs de l'article
char [] délimiteurs = {';'};

// Une CARTE
carte uneCarte;

// Nom du joueur
string nom;

// Ouverture en LECTURE
f = new BinaryReader(new FileStream("scores.txt", FileMode.Open, FileAccess.Read));
```



…