***Группа ИТ-2707. Леженко С.А.***

1. Модифицировать консольную игру "21" (создание игры было заданием одной из предыдущих домашних работ) таким образом, чтобы в программе была возможность вести одновременно 10 игр (предположим, что это небольшое казино на 10 столов). В программе необходимо реализовать класс "Игра" (или "Стол"). В программе должен быть создан всего лишь один объект "Игра" на все 10 одновременно ведущихся игр. Текущее состояние каждой игры необходимо сохранять с помощью сериализации в обьекты MemoryStream. Процесс переключения между играми состоит в сериализации текущей игры в память, и десериализации игры, к которой происходит переключение, из памяти в объект "Игра". Переключение между играми осуществлять с помощью клавиш Alt+Fномер\_игры.

***Card.cs***

using System;

namespace BJ

{

public enum suit { spades = 6, hearts = 3, diamonds = 4, clubs = 5 };

[Serializable]

public class Card

{

private suit s;

private int value;

public Card(int v, suit s)

{

this.s = s;

this.value = v;

}

public int GetValue() { return value; }

public suit GetSuit() { return s; }

public override string ToString()

{

string tmp = " ";

switch (value)

{

case 10: { tmp += "0"; break; }

case 11: { tmp += "J"; break; }

case 12: { tmp += "Q"; break; }

case 13: { tmp += "K"; break; }

case 14: { tmp += "A"; break; }

default: { tmp += value; break; }

}

tmp += Convert.ToChar(s);

if (tmp[1] == '0')

tmp = "10" + tmp[2];

return tmp;

}

};

}

***Deck.cs***

using System;

namespace BJ

{

[Serializable]

public class Deck

{

private Card[] cards = null; //Массив карт в колоде

private int nCurrentCard; //Количество выданных карт

private int N; //Общее число карт в колоде

private void shuffle()

{

Random rand = new Random();

//Мешаем случайное количество раз, но не меньше 5

int count = rand.Next(5, 100);

Card c = null;

for (int i = 0; i < count; i++)

for (int x = 0; x < cards.Length; x++)

{

int r = rand.Next(cards.Length);

c = cards[x];

cards[x] = cards[r];

cards[r] = c;

}

}

public Deck(int N = 52)

{

//Количество карт не больше 52 и кратное 4

this.N = (N <= 52 && (N % 4 == 0)) ? N : 52;

nCurrentCard = 0;

cards = new Card[this.N];

int MinCard = 2; //Минимальный номинал карты

if (this.N == 36)

MinCard = 6;

int MaxCard = this.N / 4 + MinCard; //Максимальный номинал карты

int index = 0;

int mod = this.N / 4;

for (int i = MinCard; i < MaxCard; i++)

{

cards[index] = new Card(i, suit.spades);

cards[index + mod] = new Card(i, suit.hearts);

cards[index + 2 \* mod] = new Card(i, suit.diamonds);

cards[index++ + 3 \* mod] = new Card(i, suit.clubs);

}

shuffle();

}

//Количество карт в колоде

public int GetCountCards()

{ return this.N - this.nCurrentCard; }

//Выдаёт одну карту

public Card GetCard()

{

if (this.nCurrentCard < this.N)

{

this.nCurrentCard++;

return cards[this.nCurrentCard - 1];

}

else return null;

}

public void Reset()

{

this.nCurrentCard = 0;

shuffle();

}

public override string ToString()

{

var res = new System.Text.StringBuilder();

foreach (Card c in cards)

{

res.Append(c.ToString());

}

return res.ToString();

}

};

}

***Player.cs***

using System;

namespace BJ

{

[Serializable]

public class Player

{

private string name;

public Card[] hand;

private int points; //очки

private int nCards; //всего карт в руках

private bool TakeCard; //признак брал ли игрок карту

private int money; //деньги

private int bid; //текущая ставка

public Player(string name)

{

points = 0;

nCards = 0;

TakeCard = true;

money = 100;

bid = 0;

this.name = name;

hand = new Card[20];

}

public void DrawCard(Deck aDeck)

{

hand[nCards] = aDeck.GetCard();

if (hand[nCards] != null)

nCards++;

}

public Card[] GetHand()

{

if (nCards > 0) return hand;

else return null;

}

public void Reset()

{

hand = new Card[20];

nCards = 0;

points = 0;

TakeCard = true;

}

public int GetNCards()

{ return nCards; }

public bool isTakeCard()

{ return TakeCard; }

public void SetTakeCard(bool TakeCard)

{ this.TakeCard = TakeCard; }

public void SetMoney(int money)

{

if (money > 0)

this.money = money;

else

this.money = 0;

}

public int GetMoney()

{ return this.money; }

public bool SetBid(int bid)

{

if (bid <= this.money)

{

money -= bid;

this.bid = bid;

return true;

}

else

return false;

}

public int GetBid()

{ return this.bid; }

public void SetPoints(int points)

{ this.points = points; }

public int GetPoints()

{ return this.points; }

public string GetName()

{

return this.name;

}

public override string ToString()

{

var res = new System.Text.StringBuilder();

res.Append(this.name);

res.Append(": ");

for (int i = 0; i < nCards; i++)

{

res.Append(hand[i].ToString());

}

return res.ToString();

}

};

}

***Table.cs***

using System;

namespace BJ

{

[Serializable]

public class Table

{

public readonly string PCName = "Dealer";

private Player[] players; //Массив игроков

private int nPlayers; //Количество игроков

private Deck aDeck; // Колода карт

public Table(int nCards = 52)

{

nPlayers = 2;

players = new Player[this.nPlayers];

players[0] = new Player("Player");

players[1] = new Player(PCName);

players[1].SetMoney(1000);

aDeck = new Deck(nCards);

}

public Player GetPlayer(int number)

{

if (number > -1 && number < nPlayers) return players[number];

else return null;

}

public Deck GetDeck()

{

return aDeck;

}

public int GetnPlayers()

{

return nPlayers;

}

};

}

***View.cs***

using System;

namespace BJ

{

public class View

{

public struct Coord

{

public Coord(int X, int Y)

{

this.X = X;

this.Y = Y;

}

public int X;

public int Y;

}

//Координаты вывода сообщений

static readonly Coord DeckXY = new Coord(1, 6);

static readonly Coord PlayerXY = new Coord(35, 12);

static readonly Coord DealerXY = new Coord(35, 0);

static readonly Coord WinnerXY = new Coord(30, 10);

static readonly Coord AskXY = new Coord(20, 22);

static readonly Coord BidXY = new Coord(70, 10);

static readonly int MINBID = 10;

public static void AskBid(Player aPlayer)

{

Coord cur = AskXY;

do

{

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write("Enter your bid (minimum 10$): ");

Console.SetCursorPosition(cur.X + 30, cur.Y);

int bid = 0;

try

{

bid = Convert.ToInt32(Console.ReadLine());

if (bid <= aPlayer.GetMoney() && bid >= MINBID)

{

aPlayer.SetBid(bid);

//Затираем запрос и возможное сообщение об ошибке

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write(" ");

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write(" ");

break;

}

if (bid >= aPlayer.GetMoney())

{

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write("ERROR: You do not have enough money!");

}

else if (bid < MINBID)

{

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write("ERROR: The minimum bet is 10$");

}

}

catch (FormatException)

{

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write("ERROR: Please input an integer.");

}

} while (true);

}

//Показ ставки игрока

public static void showBid(Player aPlayer)

{

Coord cur = BidXY;

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write("BID: " + aPlayer.GetBid() + "$ ");

}

//Отрисовка карт

public static void showCard(Coord cur, Card c = null)

{

Console.SetCursorPosition(cur.X, cur.Y);

if (c == null)

{

//Карта не задана - печать рубашки

Console.Write("┌──────┐ ");

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write("│▒░▒▒░▒│ ");

Console.SetCursorPosition(cur.X, cur.Y + 2);

Console.Write("│░▒░░▒░│ ");

Console.SetCursorPosition(cur.X, cur.Y + 3);

Console.Write("│▒░▒▒░▒│ ");

Console.SetCursorPosition(cur.X, cur.Y + 4);

Console.Write("│░▒░░▒░│ ");

Console.SetCursorPosition(cur.X, cur.Y + 5);

Console.Write("│▒░▒▒░▒│ ");

Console.SetCursorPosition(cur.X, cur.Y + 6);

Console.Write("└──────┘ ");

Console.SetCursorPosition(cur.X, cur.Y + 7);

Console.Write(" ");

}

else

{

//Карта задана - печать карты

Console.Write("╔══════╗");

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write("║" + c.ToString() + " ║");

Console.SetCursorPosition(cur.X, cur.Y + 2);

for (int n = 0; n < 4; n++)

{

Console.Write("║ ║");

Console.SetCursorPosition(cur.X, cur.Y + 3 + n);

}

Console.Write("╚══════╝");

}

}

//Показ колоды

public static void showDeck(Deck aDeck)

{

Coord cur = DeckXY;

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write("Deck");

if (aDeck.GetCountCards() > 0)

{

cur.X = 1;

cur.Y++;

View.showCard(cur);

//Если в колоде больше 1 карты - показ стопки карт

if (aDeck.GetCountCards() > 1)

{

cur.X++;

cur.Y++;

View.showCard(cur);

}

}

else

{

Console.SetCursorPosition(cur.X - 5, cur.Y);

for (int i = 0; i < 6; i++)

Console.WriteLine(" ");

}

}

//Показ игрока

public static void showPlayer(Player p)

{

Coord cur = PlayerXY;

Console.SetCursorPosition(cur.X, cur.Y); //Position of player name

Console.Write(p.GetName());

int firstX = cur.X;

cur.X -= 5;

cur.Y++;

// показ всех карт игрока со сдвигом на 4 символа

for (int i = 0; i < p.GetNCards(); i++)

{

View.showCard(cur, p.GetHand()[i]);

cur.X += 4;

}

cur.X = firstX;

cur.Y += 7;

Console.SetCursorPosition(cur.X, cur.Y);

if (p.GetPoints() > 0)

Console.Write(p.GetPoints() + " "); //показ очков если они есть

Console.SetCursorPosition(cur.X + 21, cur.Y);

Console.Write("Money: " + p.GetMoney() + "$ ");

}

//Показ крупье

public static void showDealer(Player p, bool open = false)

{

Coord cur = DealerXY;

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write(p.GetName() + "\t\tMoney: " + p.GetMoney() + "$ ");

int FirstX = cur.X;

cur.X -= 5;

cur.Y++;

if (p.GetNCards() > 0)

View.showCard(cur, p.GetHand()[0]);

cur.X += 4;

for (int i = 1; i < p.GetNCards(); i++)

{

View.showCard(cur, open ? p.GetHand()[i] : null);

cur.X += 4;

}

if (open)

{

cur.X = FirstX;

cur.Y += 7;

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write(p.GetPoints() + " ");

}

}

//Показ стола

public static void showTable(Table aTable, int tNum)

{

Console.Clear();

Console.Write(" Table №" + tNum);

showDeck(aTable.GetDeck());

showDealer(aTable.GetPlayer(1));

showPlayer(aTable.GetPlayer(0));

showBid(aTable.GetPlayer(0));

}

//Объявление победителя

public static void showWinner(Player aPlayer)

{

Coord cur = WinnerXY;

Console.SetCursorPosition(cur.X, cur.Y);

if (aPlayer != null)

{

Console.Write(aPlayer.GetName() + " win!");

showBid(aPlayer);

}

else

{

Console.Write(" DRAW!!! ");

}

}

//Обработчик ответа да/нет

public static bool Select(bool changeTable = false)

{

do

{

var ch = Console.ReadKey(true);

if (ch.Key == ConsoleKey.Y)

{

Console.Write("y");

return true;

}

if (ch.Key == ConsoleKey.N)

{

Console.Write("n");

return false;

}

if (ch.Modifiers == ConsoleModifiers.Alt && changeTable)

{

switch (ch.Key)

{

case ConsoleKey.F1:

case ConsoleKey.F2:

case ConsoleKey.F3:

case ConsoleKey.F4:

case ConsoleKey.F5:

case ConsoleKey.F6:

case ConsoleKey.F7:

case ConsoleKey.F8:

case ConsoleKey.F9:

case ConsoleKey.F10:

{

throw new FException(ch.Key);

}

}

}

} while (true);

}

//Запрос взять карту

public static bool TakeCard()

{

Coord cur = AskXY;

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write("Do you want to take another card? (y/n) ");

bool res = Select();

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write(" ");

return res;

}

//Запрос продолжить игру

public static bool NextPlay()

{

Coord cur = AskXY;

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write("You want to play more? (or change table Alt + F1-F10) (y/n) ");

bool res = Select(true);

Console.SetCursorPosition(cur.X, cur.Y);

Console.Write(" ");

if (!res) View.GameOver(2);

return res;

}

//Объявление о конце игры

public static void GameOver(int isWin)

{

Coord cur = WinnerXY;

Console.SetCursorPosition(cur.X, cur.Y);

switch (isWin)

{

case 0:

{

Console.Write("You lose all the money.");

Console.SetCursorPosition(cur.X - 4, cur.Y + 1);

Console.Write("Home will have to travel on foot.");

Console.ReadKey(true);

break;

}

case 1:

{

Console.SetCursorPosition(cur.X - 8, cur.Y);

Console.Write("Congratulations you have cleaned the casino!");

Console.SetCursorPosition(cur.X - 19, cur.Y + 1);

Console.Write("At the output you expect responsible people, they will guide you ...");

Console.ReadKey(true);

break;

}

case 2:

{

Console.Write("You take your money and go to rest...");

Console.SetCursorPosition(cur.X, cur.Y + 1);

Console.Write("Until we meet again!");

break;

}

}

Console.SetCursorPosition(0, 21);

}

}

}

***FException.cs***

using System;

namespace BJ

{

public class FException : ApplicationException

{

public FException(string message) : base(message)

{

}

public ConsoleKey Key { get; private set; }

public FException(ConsoleKey Key)

{

this.Key = Key;

}

}

}

***Game.cs***

using System;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace BJ

{

public class Game

{

private Table aTable;

private readonly int MINBID = 10;

private int currentTable = 0;

MemoryStream[] Tables;

public void start()

{

aTable = new Table();

Tables = new MemoryStream[10];

bool nextPlay = true;

do

{

try

{

do

{

aTable.GetPlayer(0).Reset();

aTable.GetPlayer(1).Reset();

aTable.GetDeck().Reset();

View.showTable(aTable,currentTable+1);

View.AskBid(aTable.GetPlayer(0));

FirstTurn(); //сдать первые карты

View.showTable(aTable,currentTable+1);

NextTurn(); //пока игроки берут карты

View.showWinner(Winner()); //Объявляем победителя и обрабатываем ставки

if (aTable.GetPlayer(0).GetMoney() < MINBID)

{

View.GameOver(0);

//Игра окончена обновляем стол

aTable = new Table();

break;

}

if (aTable.GetPlayer(1).GetMoney() <= 0)

{

View.GameOver(1);

//Игра окончена обновляем стол

aTable = new Table();

break;

}

} while (nextPlay = View.NextPlay());

}

catch (FException e)

{

Console.WriteLine(e.Key);

int num = Convert.ToInt32(e.Key.ToString().Trim(new char[] { 'F' })) - 1;

if (num != currentTable)

{

Tables[currentTable] = new MemoryStream();

new BinaryFormatter().Serialize(Tables[currentTable], aTable);

if (Tables[num] != null)

{

Tables[num].Seek(0, SeekOrigin.Begin);

aTable = (Table)new BinaryFormatter().Deserialize(Tables[num]);

}

else

{

aTable = new Table();

}

currentTable = num;

}

}

} while (nextPlay);

}

//Первая раздача по 2 карты

void FirstTurn()

{

for (int i = 0; i < 2; i++)

{

for (int n = 0; n < aTable.GetnPlayers(); n++)

aTable.GetPlayer(n).DrawCard(aTable.GetDeck());

}

for (int n = 0; n < aTable.GetnPlayers(); n++)

CalculatePoints(aTable.GetPlayer(n));

}

//Набор карт

void NextTurn()

{

//Игрок берет карты и не перебор

while (aTable.GetPlayer(0).GetPoints() < 21 && View.TakeCard())

{

aTable.GetPlayer(0).DrawCard(aTable.GetDeck()); //тянет карту

CalculatePoints(aTable.GetPlayer(0)); //считаем очки

View.showPlayer(aTable.GetPlayer(0)); //Обновляем игрока

View.showDeck(aTable.GetDeck()); //Обновляем колоду

}

//Крупье берет карты. берёт ещё если меньше чем у игрока и у игрока не перебор

while (aTable.GetPlayer(1).GetPoints() < aTable.GetPlayer(0).GetPoints()

&& aTable.GetPlayer(0).GetPoints() < 22)

{

aTable.GetPlayer(1).DrawCard(aTable.GetDeck());

CalculatePoints(aTable.GetPlayer(1));

View.showDeck(aTable.GetDeck());

View.showDealer(aTable.GetPlayer(1));

}

View.showDealer(aTable.GetPlayer(1), true); //Крупье открывает карты

}

//Подсчёт очков и запись в поле игрока

void CalculatePoints(Player aPlayer)

{

int res = 0;

int countAce = 0;

for (int i = 0; i < aPlayer.GetNCards(); i++)

{

switch (aPlayer.GetHand()[i].GetValue())

{

case 14:

{

if (res + 11 < 22)

{

countAce++;

res += 11;

}

else

res += 1;

break;

}

case 11:

case 12:

case 13:

{

res += 10;

break;

}

default:

{

res += aPlayer.GetHand()[i].GetValue();

break;

}

}

if (res > 21 && countAce > 0)

{

res -= 10;

countAce--;

}

}

aPlayer.SetPoints(res);

}

//Определение победителя и обработка ставок

public Player Winner()

{

//Одинаково очков или у обоих перебор - ничья

if (aTable.GetPlayer(0).GetPoints() == aTable.GetPlayer(1).GetPoints()

|| (aTable.GetPlayer(0).GetPoints() > 21 && aTable.GetPlayer(1).GetPoints() > 21))

{

aTable.GetPlayer(0).SetMoney(aTable.GetPlayer(0).GetMoney() +   
 aTable.GetPlayer(0).GetBid());

aTable.GetPlayer(0).SetBid(0);

return null;

}

//Игрок победил если:

bool isPlayerWin = ((aTable.GetPlayer(1).GetPoints() > 21) //у крупье перебор

|| ((aTable.GetPlayer(0).GetPoints() < 22) //или (у игрока не перебор и больше чем у крупье)

&& ((aTable.GetPlayer(0).GetPoints()) > (aTable.GetPlayer(1).GetPoints()))));

if (isPlayerWin)

{

//снимаем с банка деньги равные ставке игрока

aTable.GetPlayer(1).SetMoney(aTable.GetPlayer(1).GetMoney() -   
 aTable.GetPlayer(0).GetBid());

//заносим на счёт игрока выигрыш = 2 ставки

aTable.GetPlayer(0).SetMoney(aTable.GetPlayer(0).GetMoney() +   
 aTable.GetPlayer(0).GetBid() \* 2);

//обнуляем текущую ставку

aTable.GetPlayer(0).SetBid(0);

}

else

{

//заносим на счёт крупье ставку

aTable.GetPlayer(1).SetMoney(aTable.GetPlayer(1).GetMoney() +   
 aTable.GetPlayer(0).GetBid());

//обнуляем текущую ставку

aTable.GetPlayer(0).SetBid(0);

}

View.showPlayer(aTable.GetPlayer(0));

View.showDealer(aTable.GetPlayer(1), true);

return aTable.GetPlayer(isPlayerWin ? 0 : 1);

}

};

}

***Program.cs***

using System;

namespace BJ

{

class Program

{

public static void Main(string[] args)

{

Game aGame = new BJ.Game();

aGame.start();

Console.Write("\n\nPress any key to exit . . . ");

Console.ReadKey(true);

}

}

}













