通識計算機程式設計期末考參考解答

6/21/2019

試題共 7 題，兩面印製 18 頁，滿分 103

1. 。

(a) 撰寫結構(struct) **MusicNote**，宣告的成員變數為整數的 **pitch** 和 **length**。其中 **pitch** 代表音高，以樂器自動演奏控制程式檔案格式 midi 中，音高的整數表示為準。表 1/圖 4 為音樂音高(固定唱名)與 midi 音高的對應。由表 1/圖 4 可知台大校歌最後兩小節的各音符音高 pitch 為 74、77、70、72、70。成員變數 **length** 則代表音符長度，以 1/32 音符音長的倍數表示。所以我們的五個音符長度分為 32、48、16、32、96。此外還要加上一個建構式，設定 **pitch 及 length** 之值 (6%)

Ans.

```
struct MusicNote
{
    public int pitch;   // based on midi notation
    public int length;  // relative to 1/32 note
    public MusicNote(int pitch, int length)
    {
        this.pitch = pitch;
        this.length = length;
    }
}
```

(b) 宣告類別 **Voice** 的成員變數:**note**(音符)及 **lyric**(歌詞單字)。 (3%)

(c) 實作 **Voice** 的建構式。(3%)

(d) 參考圖 3，實作 **Voice** 中的函式 **Sing**，在螢幕印出單一 **Voice** 物件的音高、音長、及對應歌詞單字。這模擬機器人唱出的單一聲音。 (6%)

Ans.

```
class Voice
{
    private MusicNote note;
    private char lyric;
    public Voice(MusicNote note, char lyric)
    {
        this.note = note;
```

```
            this.lyric = lyric;
        }
        public void Sing()
        {
            Console.Write("pitch = " + note.pitch + ", ");
            Console.Write("length = " + note.length + ", ");
            Console.Write("lyric = " + lyric);
            Console.WriteLine();
        }
    }
```

(e) 寫類別 **Program** 中的主程式 **Main**，建立 **MusicNote** 及字元陣列，分別命名為 **notes** 及 **lyrics**。依照前述說明，設定這兩個陣列的內容。(6%)

(f) 完成類別 **Program** 之主程式 **Main**。用一個 **for** 迴圈，建立五個 **Voice** 物件，呼叫其 **Sing** 函式，印出每一字歌詞及對應音符。最後的螢幕輸出如圖 3。 (6%)

Ans.

```
class Program
{
    static void Main(string[] args)
    {
        // 台大校歌最後兩小節
        Voice[] song = new Voice[5];
        MusicNote[] notes = new MusicNote[5];
        notes[0] = new MusicNote(74, 32);
        notes[1] = new MusicNote(77, 48);
        notes[2] = new MusicNote(70, 16);
        notes[3] = new MusicNote(72, 32);
        notes[4] = new MusicNote(70, 96);
        char[] lyrics = {'事', '業', '都', '成', '功'};
        for(int i = 0; i < 5; ++i)
        {
            song[i] = new Voice(notes[i], lyrics[i]);
            song[i].Sing();
        }
```

```
        // ending the program
         Console.WriteLine();
        Console.WriteLine("按 enter/return 鍵結束");
        Console.ReadLine();
    }
}
```

2. 找出以下程式片段之錯誤，並予更正.

(a) (3%) 一個錯誤 (以 **Debug.Assert** 敘述的要求為準)

Ans. 沒有使用正確的建構式

```
class Circle {
  private double radius = 0;
  public Circle() {}
  public Circle(double radius)
  {
      this.radius = radius;
  }
  public double Area(double radius){// 無法設定成員變數 radius 之值
     return Math.PI*radius*radius;
  }
  public double Perimeter() {
    return 2.0*Math.PI*radius;
  }
}

class Program {
   static void Main(string[] args) {
     Circle c = new Circle(1.0);  // 決定 radius 為 1.0
     Console.WriteLine("area of c = " + c.Area(1.0));
                                          //不必要的參數
     double p = c.Perimeter();
     Debug.Assert(Math.Abs(p-2.0*Math.PI) < 1.0e-8);
   }
}
```

(b) (3%) 一個錯誤 (以 **Debug.Assert** 敘述的要求為準)

Ans.

struct 內的成員變數不宣告為 public，就使其保護層級提高，不能由其他函式直接使用

```
struct Student
{
    public string name;             // 以供其他函式直接應用
    public string registerNumber;   // 以供其他函式直接應用
    public Student(string na, string rn)
    {
        name = na;
        registerNumber = rn;
    }
}

class Program  {
  static void Main(string[] args)  {
     Student b = new Student("sk", "B645331");
     Debug.Assert(b.name == "sk");  // 直接應用 struct b 中的 name
}
```

(c) (3%) 一個錯誤。

Ans. 兩個同名函式 Get_c 僅有傳回值型別不同，不符合多載(overloading)的要求，會被視為重複宣告的函式，無法建置. 因此須將其中之一改名.

```
class Test  {
  double c;
  public Test(double c)  {
    this.c = c;
  }
  public double Get_c()  {
    return c;
  }
  public int Get_c_int()  {
    return (int) c;
  }
}
```

(d) (3%) 一種錯誤 (以 **Debug.Assert** 敘述的要求為準)

Ans. **nGenerated** 為靜態(static)成員變數，傳回其值的函式 **NGenerated** 也需宣告為靜態，應用時也應以類別名稱，而非物件名稱，加上靜態函式名呼叫.

```
class STest {
 int d;

 // number of STest objects constructed
 static int nGenerated = 0;
 public STest() {
     d = 0;
     ++nGenerated;
 }
 public STest(int i)  {
     d = i;
     ++nGenerated;
 }
 static public int NGenerated() {
     return nGenerated;
 }
}

class Program  {
 static void Main(string[] args)  {
     STest ds1 = new STest();
     STest ds2 = new STest(3);
     Assert(STest.NGenerated() == 2);
  }
 }
```

(e) (3%) 一種錯誤 (以 **Debug.Assert** 敘述的要求為準)

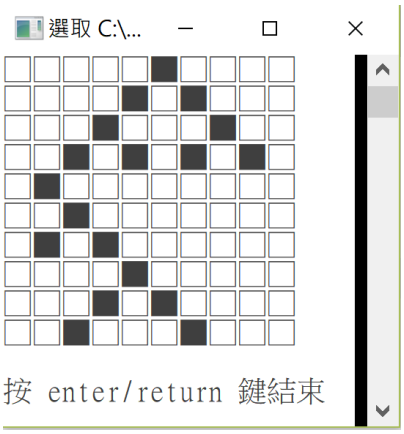
Ans. 敘述 **TestE te2 = te1;** 僅會將物件 **te1** 的參考複製給 **te2** (淺層複製)，因此二者的成員變數 e 地址相同. **te2** 對 e 的修改，也同時改變了 **te1** 的 **e**. 要避免此種情況，最好使用複製建構式(copy constructor)進行深層複製，即可避免淺層複製的問題.

```
class TestE {
  private int e;
  public TestE() {
      e = 0;
  }
  public TestE(int a) { // copy constructor
      e = a;
  }
  public TestE(TestE te)
  {
     e = te.e;
  }
  public int E  {
      set { e = value; }
      get { return e; }
  }
}

class Program  {
  static void Main(string[] args)  {
    TestE te1 = new TestE(2);
    TestE te2 = te1; new TestE(te1);
    te2.E = 4;
    Debug.Assert(te1.E == 2);
  }
}
```

3. 試寫出下列程式的輸出 (12%)

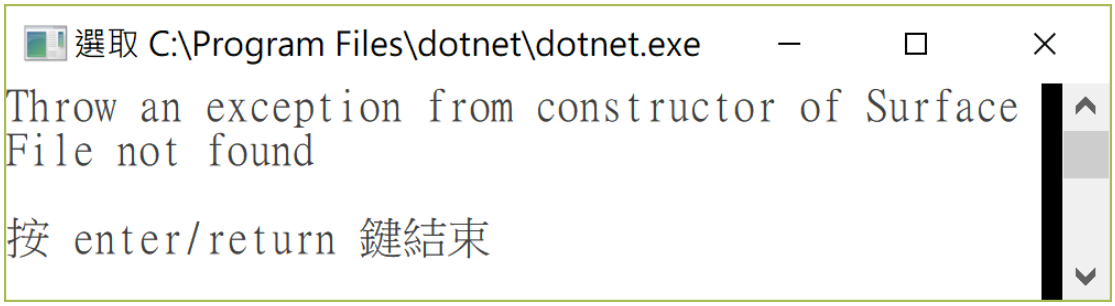
Ans.



4.

(a) (3%) 檔案 **test.surface** 尚未建立。

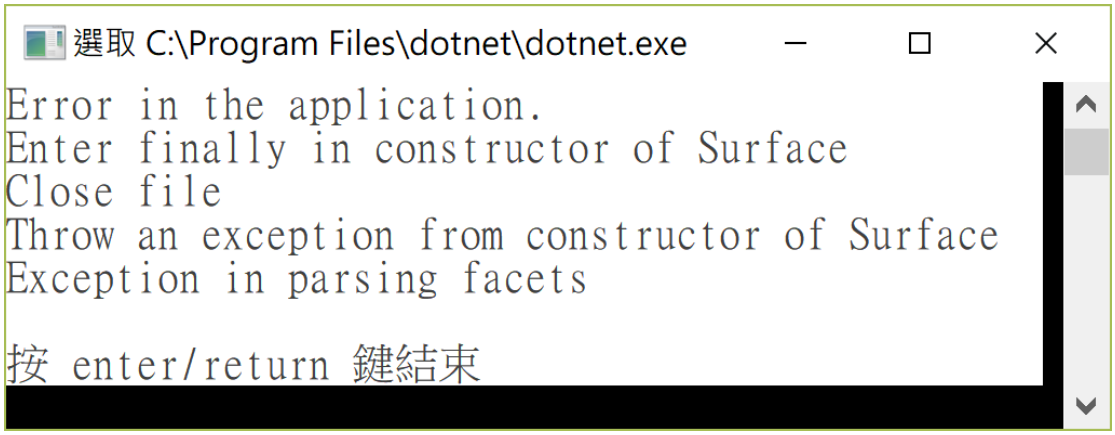
Ans.



(b) (3%) 檔案 **test.surface** 已在正確位置，且內容為

```
3 2
v 0 0
v 0.25 0
v 0 0.3
v 0.25 0.3
f 1 2 3
f 2 4 3
```

Ans.

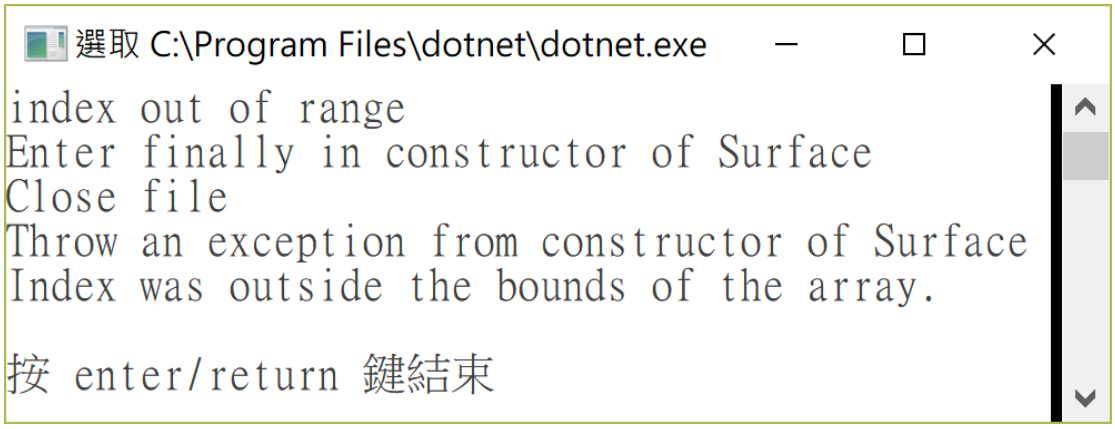


(c) (3%) 檔案 **test.surface** 已在正確位置，且內容為

```
4 2
v 0 0
v 0.25 0
```

```
v 0 0.3
v 0.25 0.3
f 0 1 2
f 1 3 2
```

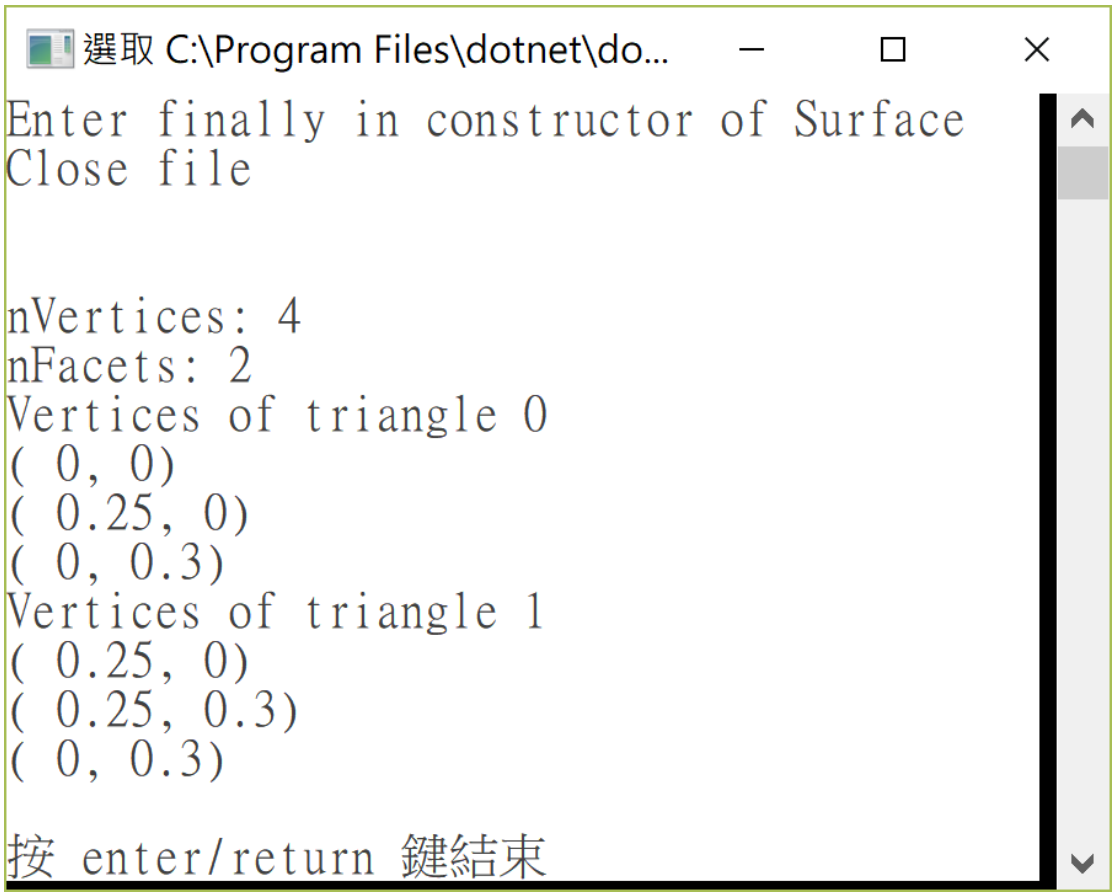
Ans.



(d) (3%) 檔案 **test.aiml** 已在正確位置，且內容為

```
4 2
v 0 0
v 0.25 0
v 0 0.3
v 0.25 0.3
f 1 2 3
f 2 4 3
```

Ans.



5. 依據以下描述及 Unity C# 腳本程式，回答問題。 (6%)

Ans.

第 18 行

```
EditorApplication.isPlaying = false;
```

表示若程式是在 Unity editor 中執行，就由這一行結束.

第 20 行

```
Application.Quit();
```

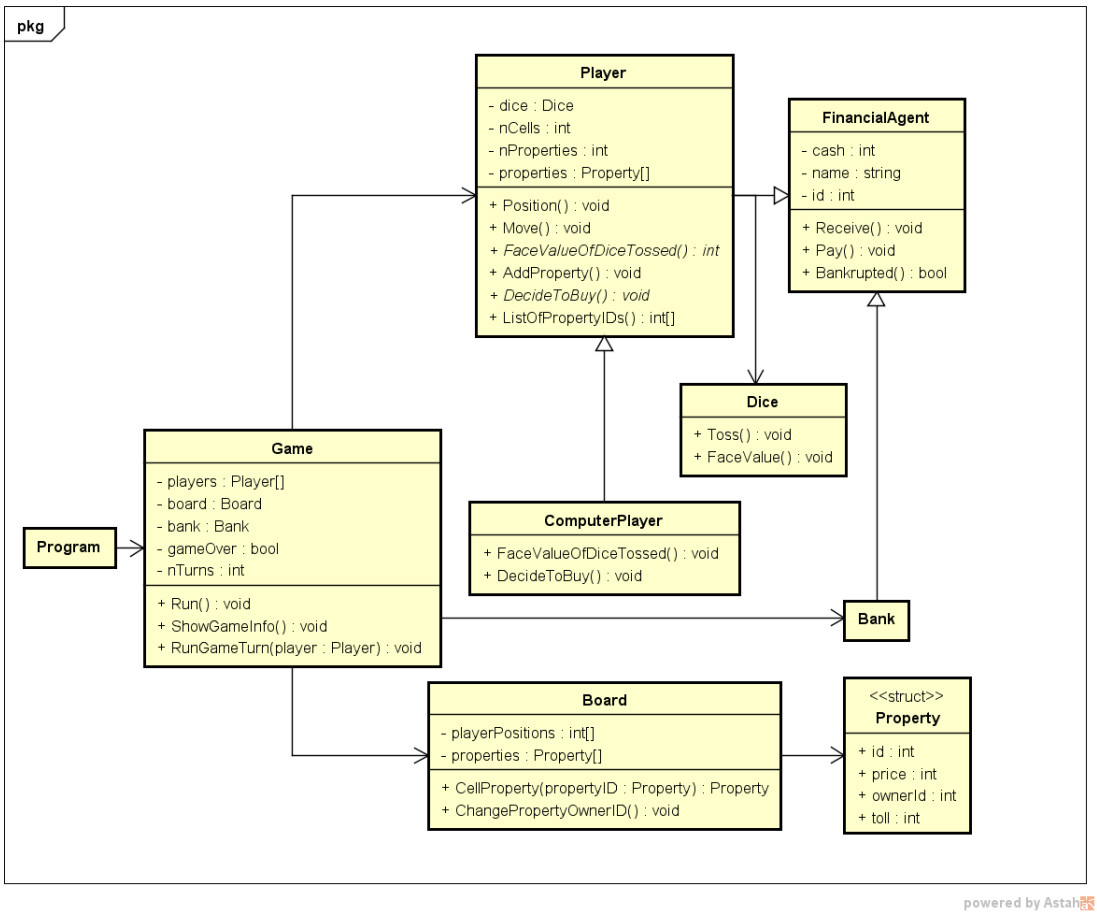
表示若程式獨立被作業系統執行，則由這行敘述結束.

6. 大富翁遊戲程式

Ans.

程式概念架構 UML 類別圖

(本圖接近最後版本. 但是在程式發展過程中，由簡而繁，經過許多次修訂，盡量表達各版本的架構概念. 若干細節並未呈現於圖中.)




```
/*
This program is intended to simulate a simplified
"Monopoly" table game.

The simplified game rules are summarized below:

1. Played by a human player and a computer player
2. Each player has $2000 initially and stays at the starting position
3. A bank is for dealing with money
4. A game board is with 16 cells
5. 12 cells are properties that can be purchased
6. A player moves on the gameboard according to a dice tossed
7. A player can purchase a property when he/she lands on it and
   the owner of the property is the bank by default
8. A player must pay 50% of the property price to the owner,
   if the owner is the other player
9. The other cells are not for sale and free for the players to land
on it
10. A player wins if the other player get bankrupted

The pseudo code of the game is given below:
1. initialize the game
2. while(both players are not bankrupted) {
2.1  human player's turn
2.2  computer player's turn
2.3  Display status for each player
  }

Pseudo code for a player's turn
```

```
1. Toss a dice
2. Move to the cell according to the face value of dice
3. If the cell property's owner is the bank,
   the player can purchase the property if he/she/computer has
   sufficient money and is willing to buy
3.1 The computer decides to buy the property, if
    a random integer generated is an even number
4. If the cell property's owner is the other player,
   the player has to pay a toll being half of the property price
5. If the player cannot afford the toll,
   he/she/computer is bankrupted

 */


using System;

namespace Problem6
{
    class Program
    {
        static void Main(string[] args)
        {
            Game game = new Game(666, 777, 888);
            game.Run();

            // ending the program
            Console.WriteLine();
            Console.WriteLine("按 enter/return 鍵結束");
            Console.ReadLine();
        }
    }
}


// Game.cs
using System;

namespace Problem6
{
    class Game
    {
        public Game(int seed1, int seed2, int seed3)
        {
            const int N_PLAYERS = 2;
            const int INITIAL_CASH = 2000;
            const int N_CELLS = 16;
            const int BANK_CASH = 100*INITIAL_CASH;
            players = new Player[N_PLAYERS];
            players[0] = new Player(
                "skjeng", 0, INITIAL_CASH, N_CELLS, seed1);
            players[1] = new ComputerPlayer(
                N_PLAYERS, INITIAL_CASH, N_CELLS, seed2, seed3);
            board = new Board(N_CELLS, N_PLAYERS);
            bank = new Bank(BANK_CASH);
            gameOver = false;
            nTurns = 0;
        }
        public Game()
        {
            const int N_PLAYERS = 2;
```

```
    const int INITIAL_CASH = 2000;
    const int N_CELLS = 16;
    const int BANK_CASH = 100*INITIAL_CASH;
    players = new Player[N_PLAYERS];
    players[0] = new Player("skjeng", 0, INITIAL_CASH, N_CELLS);
    players[1] = new ComputerPlayer(
      N_PLAYERS, INITIAL_CASH, N_CELLS);
    board = new Board(N_CELLS, N_PLAYERS);
    bank = new Bank(BANK_CASH);
    gameOver = false;
    nTurns = 0;
}

public void Run()
{
    while(!gameOver)
    {
        ++nTurns;
        ShowGameInfo();
        for(int i = 0; i < players.Length; ++i)
        {
            RunGameTurn(players[i]);
            if(players[i].Bankrupted())
            {
                Console.WriteLine(players[i].Name + "破產");
                gameOver = true;
                break;
            }
        }
    }
    return;
}

public void ShowGameInfo()
{
    Console.WriteLine("第"+ nTurns + "回合");
    for(int i = 0; i < players.Length; ++i)
    {
        Player player = players[i];
        Console.WriteLine(player.Name + "資產:");
        Console.WriteLine("現金 " + player.Cash);
        Console.Write("地產: ");
        int[] list = player.ListOfPropertyIDs();
        if(list == null)
        {
            Console.WriteLine("無");
            Console.WriteLine();
            continue;
        }
        int nProperties = list.Length;
        for(int j = 0; j < nProperties - 1; ++j)
        {
            Console.Write(list[j] + ", ");
        }
        Console.WriteLine(list[nProperties-1]);
        Console.WriteLine();
    }
}
public void RunGameTurn(Player player)
{
```

```
            int nSteps = player.FaceValueOfDiceTossed();
            player.Move(nSteps);
            int position = player.Position;
            Console.WriteLine("抵達 cell " + position);
            if( position % 4 == 0 )
            {
                Console.WriteLine("免費過境");
                Console.WriteLine();
                return;
            }
            Property property = board.CellProperty(position);
            if(property.ownerId < 0)
            {
                if(player.DecideToBuy(property))
                {
                    property.ownerId = player.ID;
                    player.Pay(property.price);
                    bank.Receive(property.price);
                    player.AddProperty(property);
                    board.ChangePropertyOwnerID(
                        position, property.ownerId);
                    Console.WriteLine(
                        player.Name + "購買了地產 " + property.id);
                }
                else
                {
                    Console.WriteLine(
                        player.Name+"沒有買地產 "+property.id);
                }
            }
            else
            {
                Player owner = players[property.ownerId];
                int toll = property.toll;
                Console.WriteLine("此為"+owner.Name+"地產， 付過路費"+toll);
                player.Pay(toll);
                owner.Receive(property.toll);
            }
            Console.WriteLine();
            return;
        }

        private Player[] players;
        private Board board;
        private Bank bank;

        private bool gameOver;
        private int nTurns;

    }
}

// FinancialAgent.cs
using System;

namespace Problem6
{
    class FinancialAgent
    {
        public FinancialAgent(string name, int id, int cash)
```

```
        {
            this.name = name;
            this.id = id;
            this.cash = cash;
        }

        public string Name
        {
            get { return name; }
        }

        public int ID
        {
            get { return id; }
        }
        public void Receive(int amount)
        {
            cash += amount;
        }

        public void Pay(int amount)
        {
            cash -= amount;
        }

        public int Cash
        {
            get { return cash; }
        }

        public bool Bankrupted()
        {
            return (cash < 0);
        }
        private string name;
        private int id;
        private int cash;
    }
}

// Player.cs
using System;


namespace Problem6
{


    class Player : FinancialAgent
    {
        public Player(string name, int id, int cash, int nCells):
            base(name, id, cash)
        {
            this.nCells = nCells;
            dice = new Dice();
```

```
    position = 0;
    properties = new Property[nCells];
    nProperties = 0;
}
public Player(
    string name, int id, int cash, int nCells, int seed):
    base(name, id, cash)
{
    this.nCells = nCells;
    dice = new Dice(seed);
    position = 0;
    properties = new Property[nCells];
    nProperties = 0;
}

virtual public int FaceValueOfDiceTossed()
{
    Console.WriteLine("按 enter 鍵以擲出骰子");
    Console.ReadLine();
    dice.Toss();
    Console.WriteLine("擲出了 " + dice.FaceValue);
    return dice.FaceValue;
}
public int Position
{
    get { return position; }
}

public void Move(int nSteps)
{
    position += nSteps;
    position = (position % nCells);
}

public void AddProperty(Property property)
{
    properties[nProperties] = property;
    ++nProperties;
```

```
}
virtual public bool DecideToBuy(Property property)
{
    bool result;
    if(property.ownerId >= 0) return false;
    Console.WriteLine();
    Console.WriteLine("地產 " + property.id + " 出售");
    Console.WriteLine("價格: " + property.price);
    Console.WriteLine("過路費: " + property.toll);
    if(Cash < property.price)
    {
        Console.WriteLine("現金不足, 無法購買");
        result = false;
    }
    else
    {
        Console.WriteLine("要購買嗎?");
        string answer = Console.ReadLine();
        result = (answer == "Y" || answer == "y");
    }
    return result;
}

public int[] ListOfPropertyIDs()
{
    if(nProperties < 1) return null;
    int[] list = new int[nProperties];
    for(int i = 0; i < nProperties; ++i)
    {
        list[i] = properties[i].id;
    }
    Array.Sort(list);
    return list;
}
protected Dice dice;
private int nCells;
private int position;
private int nProperties;
```

```
        private Property[] properties;

    }
}

// ComputerPlayer.cs
using System;

namespace Problem6
{
    class ComputerPlayer : Player
    {
        public ComputerPlayer(
            int nPlayers, int cash, int nCells, int seed,
              int nseed_dice) :
            base("Computer", nPlayers-1, cash, nCells, nseed_dice)
        {
            rand = new Random(seed);
        }

        public ComputerPlayer(int nPlayers, int cash, int nCells) :
            base("Computer", nPlayers-1, cash, nCells)
        {
            rand = new Random();
        }

        override public int FaceValueOfDiceTossed()
        {
            dice.Toss();
            int faceValue = dice.FaceValue;
            Console.WriteLine("電腦擲出 " + faceValue);
            return faceValue;
        }
        override public bool DecideToBuy(Property property)
        {
            if(property.ownerId >= 0) return false;
            int rv = rand.Next();
```

```
            bool result = ((rv % 2 == 1) && (Cash> property.price));
            return result;
        }

        private Random rand;
    }
}

// Bank.cs
using System;

namespace Problem6
{
    class Bank : FinancialAgent
    {
        public Bank(int cash) :
            base("Bank", -1, cash) {}
    }
}

// Board.cs
using System;

namespace Problem6
{
    class Board
    {
        public Board(int nCells, int nPlayers)
        {
            properties = new Property[nCells];
            for(int i = 0; i < nCells; ++i)
            {
                properties[i] = new Property(i, i*100, -1);
            }
            properties[0].price = 0;
            properties[nCells/4].price = 0;
            properties[nCells/2].price = 0;
            properties[3*nCells/4].price = 0;
```

```
            playerPositions = new int[nPlayers];
        }

        public Property CellProperty(int position)
        {
            return properties[position];
        }
        public int[] PlayerPositions()
        {
            return playerPositions;
        }

        public void ChangePropertyOwnerID(int position, int ownerId)
        {
            properties[position].ownerId = ownerId;
        }
        private int[] playerPositions;
        private Property[] properties;
    }
}

// Property.cs
using System;

namespace Problem6
{
    public struct Property
    {
        public int id;
        public int price;
        public int ownerId;
        public int toll;

        public Property(int id, int price, int ownerId)
        {
            this.id = id;
            this.price = price;
```

```
            this.ownerId = ownerId;
            toll = price / 2;
        }
    }
}

// Dice.cs
using System;

namespace Problem6
{
    class Dice
    {
        public Dice(int seed)
        {
            rand = new Random(seed);
            faceValue = 1;
        }

        public Dice()
        {
            rand = new Random();
            faceValue = 1;
        }

        public void Toss()
        {
            faceValue = rand.Next() % 6 + 1;
        }

        public int FaceValue
        {
            get { return faceValue; }
        }
        private Random rand;
        private int faceValue;
    }
}
```

本題滿分 25 分，全部程式集中寫成一個大 **Main** 函式，不區分其他函式者，最高得 18 分；善用函式者，最高得 20 分；能利用虛擬碼或 UML 類別圖思考，適當劃分類別(class)者，最高得 22 分；善用類別繼承與多型(polymorphism)者，最高得 25 分。 (25%)