通識計算機程式設計期末考參考解答

1/13/2017

1.

(a) (3%)

```
abstract class MusicNote
{
    private Pitch pitch;
    public abstract void Draw();
 }
```

(b) (6%)

```
    public MusicNote(Pitch pitch)
    {
        this.pitch = pitch;
    }
```

(c) (6%)

```
    public string PitchName()
    {
        string name;
        switch(pitch)
        {
            case Pitch.C:
                name = "C";
                break;
            case Pitch.C_SHARP:
                name = "C#";
                break;
            case Pitch.D:
                name = "D";
                break;
            case Pitch.D_SHARP:
                name = "D#";
                break;
            case Pitch.E:
                name = "E";
                break;
```

```
            case Pitch.F:
                name = "F";
                break;
            case Pitch.F_SHARP:
                name = "F#";
                break;
            case Pitch.G:
                name = "G";
                break;
            case Pitch.G_SHARP:
                name = "G#";
                break;
            case Pitch.A:
                name = "A";
                break;
            case Pitch.A_SHARP:
                name = "A#";
                break;
            case Pitch.B:
                name = "B";
                break;
            default:
                name = "Error";
                break;
        }
        return name;
    }
```

(d) (3%)

```
class QuarterNote : MusicNote
{
    public QuarterNote(Pitch pitch) :
        base(pitch) { }

    public override void Draw()
    {
        Console.WriteLine("QuarterNote:" + PitchName());
    }
}
```

(e) (6%)

```
static void DrawAllNotes(MusicNote[] notes)
{
    for (int i = 0; i < notes.Length; ++i)
    {
        notes[i].Draw();
    }
}
```

(f) (6%)

```
static void Main(string[] args)
{
    MusicNote[] notes = new MusicNote[5];
    notes[0] = new QuarterNote(Pitch.F);
    notes[1] = new DottedEighthNote(Pitch.D_SHARP);
    notes[2] = new SixteenthNote(Pitch.E);
    notes[3] = new QuarterNote(Pitch.F);
    notes[4] = new QuarterNote(Pitch.A_SHARP);
    DrawAllNotes(notes);
}
```

2. .

(a) (3%) 一個錯誤

```
class TestA_Parent
{
    private int m;
    public TestA_Parent()
    {
        m = 3;
    }
    public int M
    {
        get { return m; }
    }
}
```

```
class TestA : TestA_Parent
{
    private int n;
    public TestA()
        : base()
    {
        n = 2;
    }
    public int sum()
    {
        return m + n; // m 為父類別中的private成員變數
                      // 不可在子類別直接使用
                      // 可將TestA_Parent中的m改為protected
                      // 或將此一敘述改為
                      // return base.M + n;
    }
}
```

(b) (3%) 一個錯誤

```
class TestB
{
    private int m;
    public TestB(int m)
    {
        this.m = m;
    }
    public static int FuncB()
    {
        return m;              // 宣告為static的成員函式
                               // 只能利用也宣告為static的成員變數
                               // 不可以使用一般成員變數
                               // 此處可將static刪除，成為一般函式
                               // 問題即可解決
    }
}
```

(c) (3%) 一種錯誤。

```
class TestC
{
    private int m;
    public TestC(int m)
    {
        this.m = m;
    }

    public int M
    {
        get { return m; }
    }

    // 要把成員變數m除以2
    public void Func(int m)
    {
        m /= 2;                 // 成員函式的輸入參數 m 與成員變數 m 相同
                                // 函式中的 m 為輸入參數 m，因此成員變數 m
                                // 並未在此除以2
                                // 此處只要移除輸入參數 m，
                                // 函式中的 m 就等於成員變數 m，會除以2
    }
}
```

(d) (3%) 一種錯誤

```
class TestD
{
    private int m;
    public TestD()
    {
        m = 0;
    }
    public int M
    {
        set { m = value; }
```

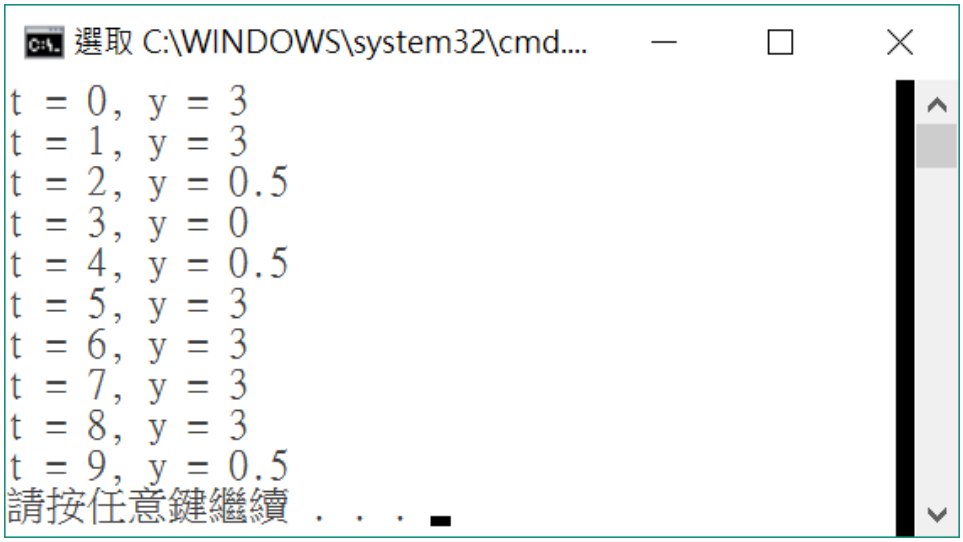
```
        get { return m; }
    }
    // 此處須加入複製建構式的宣告
    public TestD(TestD d)
    {
        m = d.m;
    }
}

class Program
{
    static void Main(string[] args)
    {
        TestD d1 = new TestD();
        TestD d2 = new TestD();

        // 錯誤! 這一行需改為利用複製建構式的寫法
        d2 = d1;                        // 此行敘述使 d1 和 d2 的物件位置
                                        // 重疊, 因此 d1.M = 15 會使
                                        // d1.m 和 d2.m 都變成 15,
                                        // 接下來的 d2.M = 40 讓 d1.m
                                        // 和 d2.m 都成為 40, 無法達成
                                        // d1和d2的 m 分別等於15和40的
                                        // 目的
                                        // 此敘述應該改為利用複製建構式的
                                        // 寫法
                                        // d2 = new TestD( d1 );

        // 要讓d1和d2的m分別等於15和40
        d1.M = 15;
        d2.M = 40;
    }
}
```

(e) (3%) 一種錯誤

```
interface TestE_IF
{
```

```
        int Triple();
        void SetM(int m);
    }
    class TestE : TestE_IF
    {
        private int m;
        public TestE()
        {
            m = 5;
        }
        public int Triple()
        {
            return 3 * m;
        }
        public void SetM(int m)  // 類別實作介面時，介面中宣告的所有
                                 // 函式都必須實作
                                 // 本題漏掉TestE_IF中宣告的void SetM
        {
            this.m = m;
        }
    }
```

3. 試寫出下列程式的輸出 (12%)



4.

(a) (3%) 檔案 **tortoisebot.urdf** 尚未建立。

```
選取 C:\WINDOWS\system32\cmd.exe
Throw a file-not-found exception from outer try-catch in XMLVersion
Throw an exception from constructor of SimpleRobot
File not found
請按任意鍵繼續 . . .
```

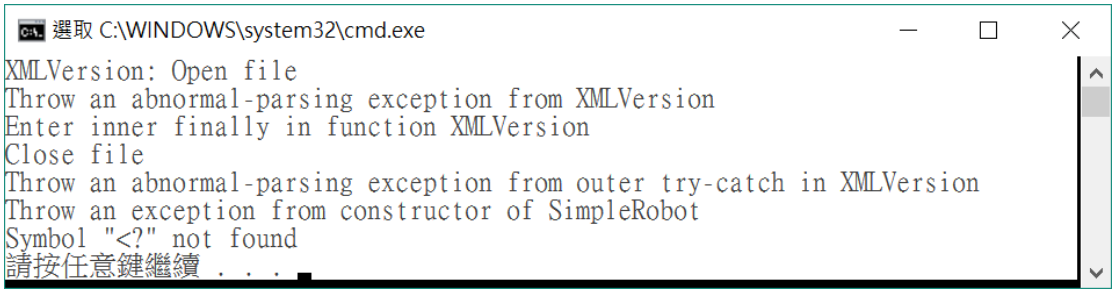
(b) (3%) 檔案 **tortoisebot.urdf** 已在正確位置，且內容為

```
<?xml version="1.0"?>
<robot name="tortoisebot">
  <link name="base_link">
  </link>
  <link name="front_caster">
  </link>
  <joint name="front_caster_joint" type="continuous">
  </joint>
</robot>
```

```
選取 C:\WINDOWS\system32\cmd.exe
XMLVersion: Open file
Enter inner finally in function XMLVersion
Close file
RobotName: Open file
Enter inner finally in RobotName
Close file
ParseForElements: Open file
Enter inner finally in ParseForElements
Close file
ParseForElements: Open file
Enter inner finally in ParseForElements
Close file


Robot name : tortoisebot
Robot links:
        base_link
        front_caster
Robot joints:
        front_caster_joint
請按任意鍵繼續 . . .
```

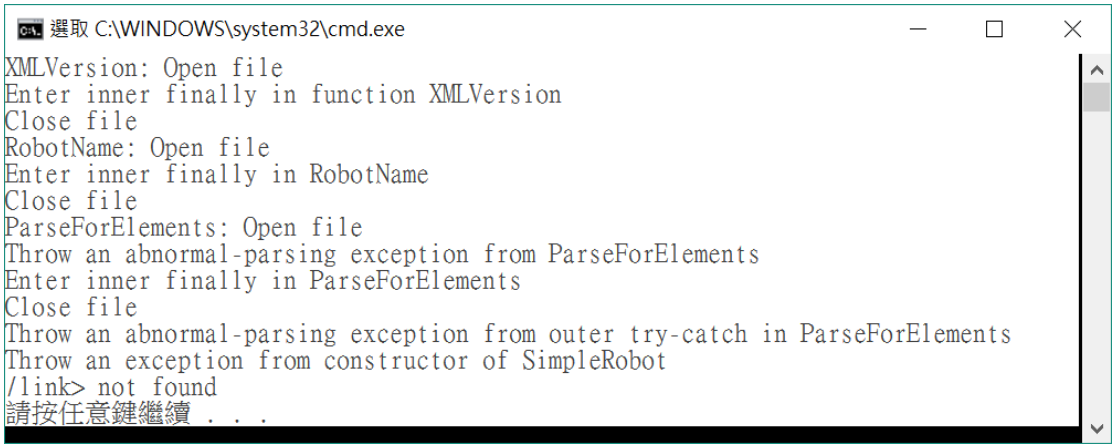
(c) (3%) 檔案 **tortoisebot.urdf** 已在正確位置，且內容為

```
<robot name="tortoisebot">
  <link name="base_link">
  </link>
  <link name="front_caster">
  </link>
  <joint name="front_caster_joint" type="continuous">
  </joint>
</robot>
```



(d) (3%) 檔案 **tortoisebot.urdf** 已在正確位置，且內容為

```
<?xml version="1.0"?>
<robot name="tortoisebot">
  <link name="base_link">
  </link>
  <link name="front_caster">
  <joint name="front_caster_joint" type="continuous">
  </joint>
</robot>
```

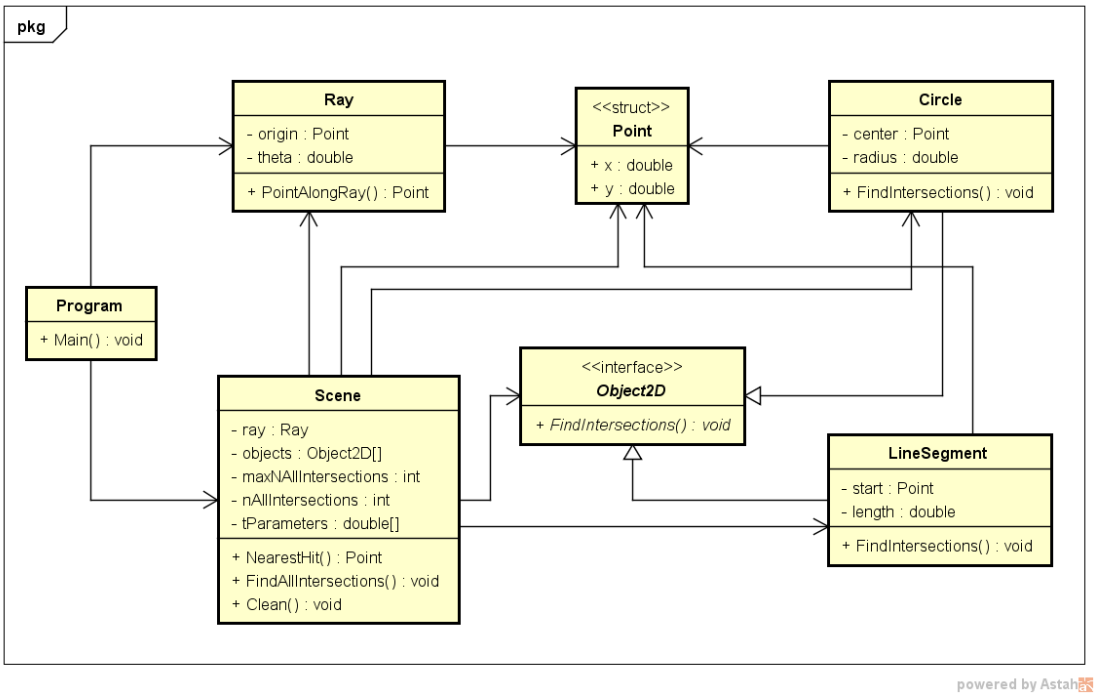


5. (6%)

```
private void DrawContents()
{
    //*****************************************
    // 在此加入必要之程式碼
    Graphics g = this.CreateGraphics();
    g.DrawLine(new Pen(Color.Black), 300, 10, 10, 300);
    int radius = 100;
    g.DrawArc(new Pen(Color.Black), 50, 50,
      200, 200, 0, 360);
    g.DrawLine(new Pen(Color.Black), 20, 250,
      220, 250);
    g.Dispose();
    //*****************************************
}
```

6. (25%)

類別圖



```
// Problem6.Program
using System;

namespace Problem6
{
    public class Program
    {
        static void Main(string[] args)
        {
            double x;
            double y;
            double theta;
            InputRayParameters(out x, out y, out theta);

            Point origin = new Point(x, y);
            theta *= Math.PI / 180.0;
            Ray ray = new Ray(origin, theta);
            Object2D[] objects = CreateObjects2D();
            int maxNAllIntersections = 2*objects.Length;
            Scene scene = new Scene(
              ray, objects, maxNAllIntersections);

            Point nearestHit = scene.NearestHit();
            OutputNearestPoint(ray.ORIGIN, nearestHit);
        }

        public static Object2D[] CreateObjects2D()
        {
            Object2D[] objects = new Object2D[2];

            double a = 1.0;
            Point center = new Point(0.0, 0.0);
            double radius = a;
            objects[0] = new Circle(center, radius);

            Point start = new Point(-1.5 * a, -a);
            double length = 2.0 * a;
            objects[1] = new HorizontalLineSegment(start, length);
```

```
            return objects;
        }

        private static void InputRayParameters(out double x,
         out double y, out double theta)
        {
            Console.WriteLine("輸入射線起點座標x, y, 以逗點分隔");
            string line = Console.ReadLine();
            string[] coord_str = new string[2];
            coord_str = line.Split(',');
            x = double.Parse(coord_str[0]);
            y = double.Parse(coord_str[1]);

            Console.WriteLine("輸入射線與x軸的夾角theta (度) ");
            theta = double.Parse(Console.ReadLine());
        }

        private static void OutputNearestPoint(
         Point rayOrigin, Point nearestHit)
        {
            double dx = nearestHit.x - rayOrigin.x;
            double dy = nearestHit.y - rayOrigin.y;
            double dist = Math.Sqrt(dx * dx + dy * dy);
            const double SMALL = 1.0e-6;
            if (dist > SMALL)
            {
                Console.WriteLine(
               "最接近交點座標 = ({0}, {1})", nearestHit.x, nearestHit.y);
            }
            else
            {
                Console.WriteLine("沒有交點");
            }
            return;
        }
    }
}
```

```
// Problem6.Point
using System;

namespace Problem6
{
    public struct Point
    {
        public double x;
        public double y;

        public Point(double x, double y)
        {
            this.x = x;
            this.y = y;
        }
    }
}
//Problem6.Ray
using System;

namespace Problem6
{
    public class Ray
    {
        private Point origin;
        private double theta;

        public Ray()
        {
            origin = new Point(0.0, 0.0);
            theta = 0.0;
        }

        public Ray(Point origin, double theta)
        {
            this.origin = origin;
            this.theta = theta;
```

```
        }

        public Point ORIGIN
        {
            get { return origin; }
        }

        public double THETA
        {
            get { return theta; }
        }

        public Point PointAlongRay(double t)
        {
            Point point;
            point.x = origin.x - t * Math.Cos(theta);
            point.y = origin.y - t * Math.Sin(theta);
            return point;
        }
    }
}

// Problem6.Scene
using System;

namespace Problem6
{
    public class Scene
    {
        private Ray ray;
        private Object2D[] objects;
        private int maxNAllIntersections;
        private int nAllIntersections;
        private double[] tParameters;

        public Scene(
         Ray ray, Object2D[] objects, int maxNAllIntersections)
        {
```

```
    this.ray = ray;
    this.objects = objects;
    this.maxNAllIntersections = maxNAllIntersections;
    double[] potential_tParameters;
    FindAllIntersections(out potential_tParameters);
    tParameters = new double[nAllIntersections];
    Clean(potential_tParameters);
    Array.Sort(tParameters);
}

public Point NearestHit()
{
    Point nearestHit = ray.ORIGIN;
    if (nAllIntersections > 0)
    {
        nearestHit = ray.PointAlongRay(tParameters[0]);
    }
    return nearestHit;
}

private void FindAllIntersections(
 out double[] potential_tParameters)
{
    potential_tParameters = new double[maxNAllIntersections];
    for (int i = 0; i < maxNAllIntersections; ++i)
    {
        potential_tParameters[i] = -1.0;
    }
    nAllIntersections = 0;
    int nIntersections;
    for(int n = 0; n < objects.Length; ++n)
    {
        objects[n].FindIntersections(
           ray, out nIntersections, out tParameters);
        if(nIntersections > 0)
        {
            for(int i = 0; i < nIntersections; ++i)
            {
```

```
                    potential_tParameters[nAllIntersections] =
                        tParameters[i];
                    ++nAllIntersections;
                }
            }
        }
    }

    void Clean(double[] potential_tParameters)
    {
        int pos = 0;
        for (int i = 0; i < potential_tParameters.Length; ++i)
        {
            if (potential_tParameters[i] > 0)
            {
                tParameters[pos] = potential_tParameters[i];
                ++pos;
            }
        }
    }
  }
}

// Problem6.Object2D
using System;

namespace Problem6
{
  public interface Object2D
  {
    void FindIntersections(
     Ray ray, out int nIntersections, out double[] tParameters);
  }

  public class Circle : Object2D
  {
    private Point center;
    private double radius;
```

```
public Circle(Point center, double radius)
{
    this.center = center;
    this.radius = radius;
}
public void FindIntersections(
 Ray ray, out int nIntersections, out double[] tParameters)
{
    double theta = ray.THETA;
    double m = (ray.ORIGIN.x - center.x)*Math.Sin(theta) -
      (ray.ORIGIN.y - center.y)*Math.Cos(theta);
    double discrim = radius * radius - m * m;
    const double SMALL = 1.0e-6;
    double b = (ray.ORIGIN.x - center.x) * Math.Cos(theta) +
      (ray.ORIGIN.y - center.y) * Math.Sin(theta);
    double t;
    nIntersections = 0;
    tParameters = new double[2];
    tParameters[0] = -1.0;
    tParameters[1] = -1.0;
    if( discrim > SMALL)
    {
        t = b - Math.Sqrt(discrim);
        if (t > 0) { tParameters[nIntersections] = t;
           ++nIntersections; }
        t = b + Math.Sqrt(discrim);
        if (t > 0) { tParameters[nIntersections] = t;
           ++nIntersections; }
    }
    else
    {
        if( discrim < SMALL && discrim > -SMALL )
        {
            t = b;
            if (t > 0) { tParameters[nIntersections] = t;
               ++nIntersections; }
        }
    }
```

```
        }
    }

    public class HorizontalLineSegment : Object2D
    {
        private Point start;
        private double length;
        public HorizontalLineSegment(Point start, double length)
        {
            this.start = start;
            this.length = length;
        }
        public void FindIntersections(
         Ray ray, out int nIntersections, out double[] tParameters)
        {
            double theta = ray.THETA;
            double t = (ray.ORIGIN.y - start.y) / Math.Sin(theta);
            double x = ray.ORIGIN.x - t * Math.Cos(theta);

            tParameters = new double[1];
            tParameters[0] = -1.0;
            nIntersections = (t > 0 &&
              x > start.x && x < start.x + length) ? 1 : 0;
            if (nIntersections > 0 )
            {
                tParameters[0] = t;
            }
        }

    }
}
```