浙江理工大学 2018—2019 学年第 2 学期

《C#程序设计》期末试卷（B）卷

本人郑重承诺：本人已阅读并且透彻地理解《浙江理工大学考场规则》，愿意在考试中自觉遵守这些规定，保证按规定的程序和要求参加考试，如有违反，自愿按《浙江理工大学学生违纪处分规定》有关条款接受处理。

承诺人签名：_____ 学号：_____ 班级：_____

一、判断题（判断下列各题是否正确，正确的在题后的括号里打 √，错误的打×。每小题 1 分，共 10 分。）

(1) a>b>c 是不合法的 ( )

(2) int[][] a = new int[][3]; 是合法的 ( )

(3) as 类似于强制类型转换但不抛出异常 ( )

(4) enum 本质是上符号化的整数 ( )

(5) 要重写父类的方法，使用关键词 override。 ( )

(6) this 指当前对象，后面用->符号来访问其成员。 ( )

(7) Lambda 表达式不能作为函数的参数 ( )

(8) 在 C#中，要注意[ ](索引)还有运算符也都是函数 ( )

(9) boxing（装箱）与 unboxing（拆箱）是引用类型与值类型之间的转换 ( )

(10) File 类的方法都是 static 的，而 FileInfo 则可以 new 一个实例。 ( )

二、单选题（在每小题的四个备选答案中选出一个正确答案，并将正确答案的序号填入题后的括号内。每小题 2 分，共 20 分。）

(1) 已知：

```
int a = 100;
void Func(ref int b){ }
```

则以下函数调用正确的是（　　）。

A. Func(ref(10*a))　　B. Func(ref 10)

C. Func(a)　　D. Func(ref a)

(2) 以下关于抽象类的叙述中正确的是（　　）。

A．抽象类可以包含非抽象方法

B．抽象类一定包含抽象方法

C．抽象类不能被实例化

D．抽象类可以是密封类

(3) （　　）方法执行指定为 Command 对象的命令文本的 SQL 语句，并返回受 SQL 语句影响或检索的行数。

A．ExecuteNonQuery　　B．ExecuteReader

C．ExecuteQuery　　D．ExecuteScalar

(4) 以下语句定义和初始化一个整形数组 a：

```
int[] a = new int[400];
for(int i=0; i<400; i++) a[i] = i;
```

为了将数组 a 的所有元素值写入 FileStream 流中，可创建（　　）类的实例对该流进行写入。

A．BinaryWriter　　B．StreamWriter

C．TextWriter　　D．PrintWriter

(5) 已知类 MyOwnClass，下面（　　）是合法的构造函数的函数头。

A. public static MyOwnClass(){}

B. public void MyOwnClass(){}

C. private int MyOwnClass(){}

D. private MyOwnClass(){}

(6) 以下关于数组的声明、实例化和初始化，不正确的是（　　）

A. int[3] a = {1, 3, 5};

B. int[] a = new int[3](1, 3, 5);

C. int[] a = new int[]{1, 3, 5};

D. int[] a = {1, 3, 5};

(7) 以下关于静态方法的说法，不正确的是（　　）

A. 静态方法不对特定实例进行操作

B. 静态方法只能直接访问静态成员

C. 在静态方法中引用 this 会导致编译错误

D. 静态方法通过类的实例来访问

(8) 在 WPF 应用程序开发环境中，在窗体上显示控件的文本，用（　　）属性

A. Text　　B. Name

C. Caption　　D. Connect

(9) 以下修饰符中，结构成员可以使用的是（　　）

A. protected　　B. abstract

C. virtual　　D. internal

(10) 以下关于密封类的说法，正确的是（　　）

A. 密封类可以用作基类

B. 密封类可以是抽象类

C. 密封类永远不会有任何派生类

D. 密封类或密封方法可以重写或继承

二、程序设计题（本题共 70 分）

1．（共 27 分）按要求定义以下类：

1）编写出一个通用的人员类(Person)，该类具有姓名(Name)、年龄(Age)、性别(Gender) 等属性，为其定义三参数的构造函数。[7 分]

2）然后对 Person 类继承，生成一个学生类(Student)，该类能够存放学生的 5 门课的成绩（百分制）。[3 分]

3）Student 类中定义一个 bool 类型只读属性 NoPass，只要有一门课不及格就为 true。[4 分]

4）Student 类定义一个方法 double Average()求出平均成绩。[3 分]

5）对 Student 类的构造函数进行重载，至少给出三种形式。[6 分]

6）对 Student 类的功能进行验证。[4 分]

2．（共 23 分）定义一个类 ComplexNumber，表示一个复数：

1）两个属性：实部(Real)，虚部(Image)，均为 double 类型[2 分]

2）一个无参构造方法和一个双参数构造方法[3 分]

3）重载 ToString 方法[2 分]

4）重载两个运算符+和-，代表两个复数的加和减[4 分]

5）编写静态方法 static bool TryParse(string s, out Complex complex)，将一个字符串解析为一个复数并输出，返回 true。如果解析不成功，返回 false。[8 分]

提示：可使用 double.TryParse(string s, out double value)方法；可使用 string 的 IndexOf(char c)来搜索某个字符在字符串中的位置；可使用 string 的 SubString(int start, int length)来提取子串。

6）针对上述所有功能写出测试程序。[4 分]

3．（共 20 分）在下划线处添加表达式、一行或多行程序完成程序中注释所要求的功能。

```
class Delegates
{
    public delegate bool NumberPredicate( int number );
    static void Main( string[] args )
    {
        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        NumberPredicate evenPredicate = IsEven;
        //利用委托变量，以 11 为参数调用 IsEven[2 分]
        Console.WriteLine( "使用委托变量调用 IsEven 方法：{0}",    ①    );
        //选出偶数
        List< int > evenNumbers = FilterArray( numbers, evenPredicate );
        DisplayList( "过滤后的偶数包含：", evenNumbers );

        //写一个 Lambda 表达式，重新选出偶数[2 分]
        ②
        //选出素数并输出[3 分]
        ③
    }

    //选择满足 predicate 的数组元素[5 分]
    private static List< int > FilterArray( int[] intArray,
        NumberPredicate predicate )
    {
        ④
    }
    //偶数判断函数[4 分]
    private static bool IsEven( int number )
    {
        ⑤
    }
    //判断是否素数[4 分]
    private static bool IsPrime( int number )
    {
        ⑥
    }
    //列表元素输出
    private static void DisplayList( string description, List< int > list )
    {
```

```
            Console.WriteLine(description);
            foreach (int i in list)
                Console.Write("{0}\t", i);
            Console.WriteLine();
    }
}
```

浙江理工大学 2018—2019 学年第 2 学期

《C#程序设计》期末试卷（B）卷标准答案和评分标准

一、判断题（本大题共 10 分，每小题 1 分）

1	2	3	4	5
√	X	√	√	√
6	7	8	9	10
X	X	√	√	√

二、单选题（本大题共 20 分，每小题 2 分）

1	2	3	4	5
D	A	A	A	D
6	7	8	9	10
A	D	A	D	C

二、程序设计题（共 70 分）

1、

```
class Person//1分
{
    public String Name { get; set; }//3个字段2分
    public int Age { get; set; }
    public bool Gender { get; set; }
    public Person(string name, int age, bool gender)//2分
    {
        Name = name;
        Age = age;
        Gender = gender;
    }
}

class Student : Person//1分
{
    //平均成绩，1分
    double[] scores = new double[5];//也可以拆分为多double型成绩字段

    public Student() : base("", 0, false)//2分
    {
        for (int i = 0; i < 5; i++)
        {
            scores[i] = 0.0;
        }
```

```
    }

    public Student(params double[] scores1) : base("", 0, false) //2分
    {
        for (int i = 0; i < 5; i++)
        {
            scores[i] = scores1[i];
        }
    }

    public Student(string name, int age, bool gender, params double[] scores1)
        : base(name, age, gender) //2分
    {
        for (int i = 0; i < 5; i++)
        {
            scores[i] = scores1[i];
        }
    }

    public double Average()//平均成绩，属性或函数均可
    {
        double sum = 0.0;
        for (int i = 0; i < 5; i++)
        {
            sum += scores[i];
        }
        return sum / 5;
    }

    public bool NoPass
    {
        get
        {
            for (int i = 0; i < 5; i++)
            {
                if (scores[i] < 60)
                    return true;
            }
            return false;
        }
    }
}

class Test
{
    public static void Main()//验证，1分
    {
        Student stu1 = new Student(50, 60, 70, 80, 90);
        Console.WriteLine(stu1.Average());
        Console.WriteLine(stu1.NoPass);
    }
}
```

2

```
class Complex
```

```
{
    public double Real { get; set; }
    public double Image { get; set; }
    public Complex(double real, double image)
    {
        Real = real;
        Image = image;
    }
    public static Complex operator+(Complex c1, Complex c2)
    {
        return new Complex(c1.Real + c2.Real, c1.Image + c2.Image);
    }
    public static Complex operator -(Complex c1, Complex c2)
    {
        return new Complex(c1.Real - c2.Real, c1.Image - c2.Image);
    }
    public static bool TryParse(string s, out Complex complex)
    {
        complex = new Complex(0, 0);
        if (s == null)
            return false;

        bool negativeImage = false;
        s = s.ToUpper();
        int pos = s.IndexOf('+');
        if (pos < 0)
        {
            pos = s.IndexOf('-');
            if (pos >= 0)
                negativeImage = true;
        }

        int iPos = s.IndexOf('I');
        if (iPos < 0)
        {
            if (pos > 0)
                return false;
            else
            {
                double value;
                if (double.TryParse(s, out value))
                {
                    complex = new Complex(value, 0);
                    return true;
                }
                else
                    return false;
            }
        }
        else
        {
            string s1 = s.Substring(0, pos);
            string s2 = s.Substring(pos + 1, iPos - pos - 1);
            double real, image;
            if (!double.TryParse(s1, out real) || !double.TryParse(s2, out image))
```

```
                return false;
            else
            {
                complex = new Complex(real, negativeImage? -image : image);
                return true;
            }
        }
    }
    public override string ToString()
    {
        if (Image == 0)
            return string.Format($"{Real}");
        else
            return string.Format($"{Real}" + (Image > 0 ? '+' : '-') +
$"{Math.Abs(Image)}i");
    }
}
```

测试：

```
class Ex3
{
    public static void Main()
    {
        Complex c = new Complex();
        Console.WriteLine(c);
        Complex result;
        bool ok = Complex.TryParse(c.ToString(), out result);
        if (!ok)
            Console.WriteLine("错了");
        Console.WriteLine(result);
        Complex c2 = c + result;
        Console.WriteLine(c2);
    }
}
```

3

（1）evenPredicate(11)

（2）evenNumbers = FilterArray(numbers, (x => x%2 != 0);

DisplayList("过滤后的偶数包含：", evenNumbers);

（3）List<int> primeNumbers = FilterArray(numbers, IsPrime);

DisplayList("Use IsPrime to filter prime numbers: ", primeNumbers);

（4）

```
List<int> ilist = new List<int>();
foreach(int i in intArray)
{
    if(predicate(i))
```

```
                ilist.Add(i);
            }
            return ilist;
```

（5）return (number % 2 == 0);

（6）

```
            int count = 0;
        if (number < 2) return false;
        for (int i = 2; i < number; i++)
        {
            if (number % i != 0)
                continue;
            count++;
        }
        return (count == 0);
```