**南昌大学2021—2022学年第一学期期中考试试卷**

学院：软件学院 课程：C#程序设计 时间：80分钟

**一．单选题（4分 \* 10 = 40分）**

1．在类作用域中能够通过直接使用该类的（  D ）成员名进行访问。

A. 私有 B. 公用  C. 保护 D. 任何

2.下列选项中，（ C ）是引用类型。

A. enum类型 B. struct类型 C. string类型 D. int类型

3.在C#中无需编写任何代码就能将int型数值转换为double，称为（ B ）。

A.显式转换 B.隐式转换 C.数据类型变换 D.变换

4.在类的成员中，用于存储属性值的是（ D ）

A.属性  B.方法  C.事件 D.成员变量

5.C#程序中，可以使用try…catch机制来处理程序出现的（ B ）错误。

A. 语法 B. 运行 C. 逻辑 D. 拼写

6.下列关于try…catch…finally语句说明中，不正确的是（ D ）。

A. catch块可以有多个 B. catch块是可选的

C. finally块可以没有 D.可以只有try块

7.下列关于构造函数的描述正确的是（ C ）

A.构造函数可以声明返回类型。 B.构造函数不可以用private修饰

C.构造函数必须与类名相同   D.构造函数不能带参数

8.引用类型主要有4种：类类型、数组类型、接口类型和（ C    ）。

A.对象类型 B.字符串类型 C.委托类型 D.整数类型

9.数据类型转换的类是（  B   ）。

A. Mod B. Convert C. Const D.  Single

10.在C#编制财务软件时，需要创建存储流动资金的临时变量，则应使用下列哪条语句？（ A ）

A. decimal theMoney B. int theMoney

C. string theMoney D. double theMoney

**二．简答题（5分 \* 2 = 10分）**

1. C#中静态方法与非静态方法在定义和使用过程中有什么区别？

定义：使用了static 修饰符的方法为静态方法，反之则是非静态方法。

使用：

1. C#静态方法属于类所有，类实例化前即可使用。
2. 非静态方法可以访问类中的任何成员，静态方法只能访问类中的静态成员。
3. 因为静态方法在类实例化前就可以使用，而类中的非静态变量必须在实例化之后才能分配内存，这样，C#静态方法调用时无法判断非静态变量使用的内存地址。所以无法使用。而静态变量的地址对类来说是固定的，故可以使用。
4. 静态方法效率上要比实例化高，静态方法的缺点是不自动进行销毁，而实例化的则可以做销毁。
5. 静态方法和静态变量创建后始终使用同一块内存，而使用实例的方式会创建多个内存

简述break和continue语句的区别？

continue语句只结束本次循环，而不是终止整个循环的执行。

break语句则是结束整个循环过程，不再判断执行循环的条件是否成立。break语句可以用在循环语句和switch语句中。在循环语句中用来结束内部循环；在switch语句中用来跳出switch语句。

1. **用C#语言编程题（10分 + 40分 = 50分）（其他语言不得分）**

1.编程计算N的阶乘最后非0的三位数。要求：

（1）N从键盘输入，且1<N<100000，（注意N的范围）

（2）不够三位的补齐三位，如3! = 6，应打印006，4!=24，应打印024。

using System;

namespace midtest1

{

class Program

{

static void Main(string[] args)

{

long number;

Console.Write("请输入一个100000以内的值：");

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

long mul=1;

for (int index = 1; index <= number; index++)

{

mul = mul \* index;

mul = mul % 1000;

}

Console.WriteLine("{0}的阶乘结果后三位为：{1}", number,String.Format("{0:D3}", mul));

/\*

Console.Write("请输入一个整数：");

long n = long.Parse(Console.ReadLine());

long i = 1;

long result = 1;

while (i <= n)

{

result = ((result % 1000) \* (i % 1000)) % 1000;

i++;

}

Console.WriteLine("{0}!={1}", n, String.Format("{0:D3}", result));

Console.ReadLine();\*/

}

}

}

2.设计一个分数类Fraction，

（1）成员变量：分数的分子和分母，均为整数，访问级别为私有；√

（2）属性：可读不可写，分母必须为正，分子符号随意，若不满足，则作变换；

如3/-4，应设置为-3/4；

（3）构造函数：无参构造函数，默认初始化为1/1；

有参构造函数，给定分子和分母的值（注意分母为0的情况）；

（4）重载分数的运算符+（结果应为最简分数的形式，注意约分）；

（5）在测试类中测试。

评分说明：（2）（4）10分，其它均为5分

using System;

namespace midtest2

{

class Program

{

static void Main(string[] args)

{

Fraction fenshu=new Fraction();

fenshu.print();

Fraction fenshu2 = new Fraction(-155,-25);

fenshu2 = +fenshu2;

fenshu2.print();

Fraction fenshu3 = new Fraction(-4, 0);

fenshu3 = +fenshu3 ;

fenshu3.print();

Fraction fenshu4 = new Fraction(3, -9);

fenshu4 = +fenshu4;

fenshu4.print();

}

}

class Fraction

{

private

int molecule;//分子

int denominator;//分母

bool isError = false;

public Fraction()

{

molecule = 1;

denominator = 1;

}

public Fraction(int a,int b)

{

molecule = a;

denominator = b;

if (b == 0)

{

Console.WriteLine("出现错误！分母不能为0！构造失败！");

isError = true;

}

}

public void print()

{

if (this.denominator\*this.molecule < 0)

{

this.molecule = -Math.Abs(this.molecule);

}

if (isError == false)

Console.WriteLine("分数为：{0}/{1}", this.molecule, this.denominator);

else

Console.WriteLine("出现错误！分母不能为0！打印失败！");

}

public static Fraction operator +(Fraction x)

{

bool isfunshu = false;

if (x.molecule \*x.denominator<0) isfunshu = true;

int min = Math.Min(Math.Abs(x.molecule), Math.Abs(x.denominator));

x.denominator = Math.Abs(x.denominator);

x.molecule = Math.Abs(x.molecule);

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

{

if (x.molecule % i == 0 && x.denominator % i == 0)

{

x.molecule /= i;

x.denominator /= i;

i = 1;

min = Math.Min(x.molecule, x.denominator);

}

}

if (isfunshu == true) x.molecule = -x.molecule;

return x;

}

}

}