학과 ______________ 학번 ______________ 이름 ______________

# 중간고사

**담당교수: 단국대학교 멀티미디어공학전공 박경신**

- **답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번대신 암호를 사용할 것임.**

**1. 다음은 for 루프를 이용해서 변수의 값을 매 번 3배로 증가시켜서 1 3 9 27 81 243을 출력하는 프로그램이다. *while* 루프로 바꾸어서 다시 작성하라. (5점)**

```
for (int i = 1; i <= 243; i *= 3) {
        Console.Write("{0} ", i);
}
```

```
int i = 1;
while( i <= 243) {
        Console.Write("{0} ", i);
        i *= 3;
}
```

**2. break, continue 문의 차이점을 예를 들어서 간단히 설명하라. (5점)**

Break문은 반복문 (for, while, do while, foreach)이나 switch 문의 코드 블록 밖으로 제어권이 나오게 하는 것임
Continue문은 break 문과 비슷하나 제어권을 반복문의 조건식으로 이동시킴

```
int k = 0;
do {
        k++;
        if (k==5)
            break;
        Console.Write("{0} ", k)
} while(k < 10);
// break 을 사용했을 시 1 2 3 4 출력

int k = 0;
do {
        k++;
        if (k==5)
            continue;
        Console.Write("{0} ", k)
} while(k < 10);
// continue 을 사용했을 시 1 2 3 4 6 7 8 9 10 출력
```

학과 ____________ 학번 ____________ 이름 ____________

**3. 다음은 C#의 값 형식 (Value Type)과 참조 형식 (Reference Type)를 비교하는 간단한 프로그램이다. 이 프로그램이 실행되는 출력결과를 쓰시오. (10점)**

```
class IntClass
{
        public int value = 0;
}

class TypeTest
{
    static void Main() {
        int val1 = 0;
        int val2 = val1;
        if (val1 == val2)
                Console.WriteLine("val1 == val2");
        else
                Console.WriteLine("val1 != val2");
        val2 = 123;
        object val3 = val2;
        if (val1 == val2)
                Console.WriteLine("val1 == val2");
        else
                Console.WriteLine("val1 != val2");
        if ((int)val3 == val2)
                Console.WriteLine("val3 == val2");
        else
                Console.WriteLine("val3 != val2");

        IntClass ref1 = new IntClass();
        IntClass ref2 = new IntClass();
        if (ref1 == ref2)
                Console.WriteLine("ref1 == ref2");
        else
                Console.WriteLine("ref1 != ref2");
        IntClass ref3 = ref1;
        if (ref3 == ref1)
                Console.WriteLine("ref3 == ref1");
        else
                Console.WriteLine("ref3 != ref1");
        ref3.value = 123;
        if (ref1.value == ref2.value)
                Console.WriteLine("ref1.value == ref2.value");
        else
                Console.WriteLine("ref1.value != ref2.value");
        if (ref1.value == ref3.value)
                Console.WriteLine("ref1.value == ref3.value");
        else
                Console.WriteLine("ref1.value != ref3.value");

        Console.WriteLine("Value type: val1={0}, val2={1}, val3={2}", val1, val2, val3);
        Console.WriteLine("Reference type: ref1.value={0}, ref2.value={1}, ref3.value={2}",
                        ref1.value, ref2.value, ref3.value);
    }
}
```

학과 ____________ 학번 ____________ 이름 ____________

**출력결과:**

```
val1 == val2
val1 != val2
val3 == val2
ref1 != ref2
ref3 == ref1
ref1.value != ref2.value
ref3.value == ref1.value
Value type: val1=0, val2=123, val3=123
Reference type: ref1.value=123,
ref2.value=0, ref3.value=123
```

**4. 다음은 메소드의 매개변수 전달방식을 테스트하는 프로그램이다. 이 프로그램이 실행되는 출력결과를 써라. (5점)**

```
class MethodTest
{
    void Add(int a, int b, ref int c) {
        c = a + b;
    }
    void Mul(ref int a, out int b) {
        b = 2 * a;
    }
    int AddList(params int[] v) {
        int total, i;
        for (i=0, total=0; i<v.Length; i++) {
                total += v[i];
        }
        return total;
    }
    static void Main() {
        int x = 3;
        int y = 4;
        int z = 5;
        int w;
        MethodTest m = new MethodTest();
        m.Add(x, y, ref z);
        Console.WriteLine("Add : x={0} y={1} z={2}", x, y, z);
        m.Mul(ref x, out w);
        Console.WriteLine("Mul : w={0}", w);
        w = m.AddList(2, x, y, z, 6);
        Console.WriteLine("AddList : w={0}", w);
    }
}
```

**출력결과:**

```
Add : x=3 y=4 z=7
Mul : w=6
AddList : w=22
```

학과 ____________ 학번 ____________ 이름 ____________

**5. 4번 프로그램을 참고하여 메소드의 매개 변수로 Pass-by-value, Pass-by-reference, Pass-by-output 방법에 대해 간단히 설명하라. (5점)**

- Pass-by-value
  - 값형식(value type)은 값에 의한 매개변수 전달 방식을 사용
  - void Add(int a, int b, ref int c)에서 a와 b는 pass-by-value 방식
- Pass-by-reference
  - 참조형식(reference type)은 참조에 의한 전달방식을 사용
  - ref 키워드를 사용하며 호출 전에 매개변수는 초기화가 필요함
  - void Add(int a, int b, ref int c)에서 c는 pass-by-reference 방식
- Pass-by-output
  - 출력에 의한 전달방식으로 값을 반환 받을 때만 사용
  - out 키워드를 사용하며 호출 전에 매개변수는 초기화가 필요하지 않음
  - void Mul(ref int a, out int b)에서 b는 pass-by-output 방식

**6. 메소드 오버로딩 (method overloading)과 메소드 오버라이딩 (method overriding)에 대해 예를 들어서 간단히 설명하라 (5점)**

- 메소드 오버로딩 (method overloading)
  - 동일한 함수명에 매개변수가 다른 함수를 둘 이상 정의하는 것으로, 동일한 함수 기능을 수행하지만 다른 매개변수의 경우를 처리할 때 사용

```
int Max(int a, int b);
double Max(double a, double b);
```

- 메소드 오버라이딩 (method overriding)
  - 상속받은 파생 클래스에서 동일한 함수명에 동일한 매개변수로 정의하여 함수를 재정의하는 것으로, 상속되어진 함수의 기능을 변경해서 재사용하고 싶을 때 사용
  - Virtual 메소드와 override 메소드는 이름, 접근 제한자, 반환 값, 및 매개변수 리스트가 동일해야 함

```
class Point
{
        public virtual void Print();// Print()함수가 x,y를 출력
}
class Point3D : Point
{
        public override void Print();// Print()함수가 x,y,z를 출력
}
```

학과 ____________ 학번 ____________ 이름 ____________

**7. 다음은 Point 클래스를 사용한 간단한 프로그램이다. 이 프로그램에 이탤릭체로 된 메소드 내부에서 오류를 찾아서 지적하고 그 이유를 설명하라. 그리고 오류를 수정하고 난 후 출력 결과를 써라. (10점)**

```
class Point
{
    public int x;
    public static int y;
    public Point() {
        x = 1;
        y = 1;
    }
    public static void Set(int x, int y) {
        this.x = x;        // Point p = new Point(); p.x = x; 정적메소드 안에서는 인스턴스필드 사용불가
        this.y = y;        // Point.y = y; 정적메소드 안에서는 this 키워드 사용불가 Point.y로 사용해야함
    }
    public void Move(int x, int y) {
        this.x += x;
        this.y += y;       // Point.y += y;  정적필드를 this 키워드 사용불가 Point.y로 사용해야함
    }
    public void Print() {
        Console.WriteLine("x={0} y={1}", x, y);
    }
}

class StaticTest
{
    static void Main() {
        Point p = Point();// Point p = new Point(); 클래스의 개체 생성시 new 키워드 사용
        p.Print();
        p.x = 10;
        p.y = 10;          // Point.y = 10; 정적필드는 반드시 클래스명.정적필드명 사용
        p.Print();
        p.Move(10, 10);
        p.Set(10, 10);     // Point.Set(10, 10); 정적메소드는 반드시 클래스명.정적메소드명 사용
        Point.Print();     // p.Print(); 인스턴스메소드는 반드시 생성한객체명.인스턴스메소드명 사용
    }
}
```

**출력결과:**

```
x=1 y=1
x=10 y=10
x=20 y=10
```

학과 ______________ 학번 ______________ 이름 ______________

**8. 다음 프로그램을 참고하여, 다형성 (Polymorphism)에 대해 간단히 설명하라. (10점)**

```
public class Person
{
    protected string name;
    protected int age;
    public Person(string name, int age) {
        this.name = name;
        this.age = age;
    }
    public virtual void Print() {
        Console.WriteLine("Person name={0} age={1}", name, age);
    }
}
public class Student : Person
{
    protected int id;
    public Student(string name, int age, int id) : base(name, age) {
        this.id = id;
    }
    public override void Print() {
        Console.WriteLine("Student name={0} age={1} id={2}", name, age, id);
    }
}
public class PersonTest
{
    public static void Main() {
        Person p = new Person("길동이", 40);
        p.Print();
        Student s = new Student("둘리", 1000, 2);
        s.Print();
        p = s;
        p.Print();
        s = (Student) p;
        s.Print();
    }
}
```

학과 ____________ 학번 ____________ 이름 ____________

- **다형성** - 상위클래스에서 선언된 메소드를 하위클래스에서 재구현 (override)하고 실행시간에 새로 정의된 override된 멤버의 내용으로 처리(late binding)

```
Person p = new Person("길동이", 40);
p.Print(); // Person의 Print 호출
Student s = new Student("둘리", 1000, 2);
s.Print(); // Student의 Print 호출
p = s;
p.Print(); // 그러나 p는 Student이므로 Student의 Print 호출
           // 실행시간에 새로 재정의된 멤버의 내용으로 처리함 –late binding
s = (Student) p;
s.Print(); // s는 Student이므로 Student의 Print 호출
           // 실행시간에 새로 재정의된 멤버의 내용으로 처리함 –late binding
```

**9. 8번 프로그램을 참고하여 Upcasting과 Downcasting을 간단히 설명하라. (5점)**

```
p = s;           // upcasting – 암시적으로 Student인 s는 Person으로 형변환되어 p로 저장
s = (Student) p; // downcasting – 상위클래스가 하위클래스로 형변환시 명시적인 형변환
```

**10. 8번 프로그램에서 Student 클래스가 아래의 Record 클래스를 갖도록 (Has-a model) 클래스 내부를 수정하는 코드를 작성하라. 이 때, Main() 메소드의 테스트 내용이 정상적으로 작동되도록 Student 클래스에 Record에 관련된 멤버 필드와 메소드를 추가한다. (10점)**

```
public enum Grade { A, B, C, D, F };
public class Record
{
    int kor, eng, math;
    public Record() : this(0, 0, 0) { }
    public Record(int kor, int eng, int math) {
        this.kor = kor;
        this.eng = eng;
        this.math = math;
    }
    public int GetAverage() {
        return (kor + eng + math) / 3;
    }
```

```
    public Grade GetGrade() {
        // 중간생략...
        return grade;
    }
    public void Print() {
        Console.WriteLine("Record kor={0} eng={1} math={2} grade={3}", kor, eng, math,
GetGrade());
    }
}
public class Student : Person
{       // 아래에 Student 클래스가 Record 클래스를 갖는 코드를 추가할 것
    protected int id;
    protected Record course;
    public Student(string name, int age, int id) : base(name, age) {
        this.id = id;
        course = new Record();
    }
    public Student(string name, int age, int id, int kor, int eng, int math) : base(name, age) {
        this.id = id;
        course = new Record(kor, eng, math);
    }
    public Grade GetGrade() {
        return course.GetGrade();
    }
    public override void Print() {
        Console.WriteLine("Student name={0} age={1} id={2}", name, age, id);
        course.Print();
    }
}
public class PersonTest
{
    public static void Main() {
        Student s = new Student("희동이", 3, 3, 70, 80, 90);
        s.Print();          // 학생의 이름, 나이, ID, 국어, 영어, 수학, 성적 출력
        Grade g = s.GetGrade();
        Console.WriteLine("희동이 성적: {0}", g); // "희동이 성적: B" 출력
    }
}
```

학과 ____________ 학번 ____________ 이름 ____________

**11. 다음 대리자(Delegate)를 테스트하는 프로그램에서 빈 칸을 채워라. (10점)**

```
public delegate void Operator(int value);

class IntClass
{
    public int value = 0;
    public void Add(int value) {
        this.value += value;
    }
    public void Substract(int value) {
        this.value -= value;
    }
    public void Multiply(int value) {
        this.value *= value;
    }
    public void Divide(int value) {
        this.value /= value;
    }
    public override string ToString() {
        return string.Format("value = {0}", value);
    }
}
class DelegateTest
{
    static void Main() {
        IntClass num = new IntClass();              // num.value = 0

        Operator Op = new Operator(num.Add);
        Op(10);                                     // num.Add(10) 호출
        Console.WriteLine(num);                     // value=10

        Op += new Operator(num.Multiply);
        Op(10);                                     // num.Add(10)와 num.Multiply(10) 호출
        Console.WriteLine(num);                     // value=200

        Op -= new Operator(num.Add);
        Op(10);                                     // num.Multiply(10) 호출
        Console.WriteLine(num);                     // value=2000

        Op = Op - new Operator(num.Multiply) + new Operator(num.Divide);
        Op(10);                                     // num.Divide(10) 호출
        Console.WriteLine(num);                     // value=200

        Op += new Operator(num.Substract);
        Op(10);                                     // num.Divide(10)와 num.Substract(10) 호출
        Console.WriteLine(num);                     // value=10

        Op -= new Operator(num.Divide);
        Op(10);                                     // num. Substract (10) 호출
        Console.WriteLine(num);                     // value=0
    }
}
```

**12. 추상 클래스 (abstract class)와 봉인 클래스 (sealed class)와 인터페이스 (interface)에 대해 간단히 설명하라. 그리고 Shape(도형), Rectangle(사각형), Square(정사각형) 클래스에 void Draw() 그리기 메소드와 int Area() 크기 메소드를 예로 들어서 비교 설명하라. (10점)**

- **추상 클래스 (abstract class)**
  - 개체들의 표준을 추상화한 클래스로 상속관계에서 가장 상위에 존재
  - abstract 키워드를 사용하여 정의
- **봉인 클래스 (sealed class)**
  - 추가 파생을 방지하기 위하여 클래스 자신 또는 멤버를 봉인하는 선언을 하여 다른 클래스가 상속하는 것을 막음
  - sealed 키워드를 사용하여 정의
- **인터페이스 (interface)**
  - 하위 클래스에서 구현되어야 하는 기능을 선언할 시 일반적으로 이질적인 클래스들이 공통으로 제공해야 할 메소드들을 선언할 때 사용
  - class 대신 interface 키워드를 사용하여 선언

```
public abstract class Shape
{
        public abstract void Draw();// 실제 도형에서 그림을 그리는 Draw() 호출
        public abstract int Area();// 실제 도형에서 크기를 계산하는 Area() 호출
}
public class Rectangle : Shape
{
        protected int w, h;
        public Rectangle(int w, int h) { this.w = w; this.h = h; }
        public override void Draw() { Console.WriteLine("사각형 그리기"); }
        public override int Area() { return w*h; }
}
public sealed class Square : Rectangle   // sealed는 Square class를 상속할 수 없게 함
{
        public Square(int size) : base(size, size) {}
        public override void Draw(){ Console.WriteLine("정사각형 그리기"); }
        public override int Area(){ base.Area(); }
}
interface IShape    // abstract method를 가진 abstract class는 인터페이스로 전환가능
{
        void Draw();// 실제 도형에서 그림을 그리기 호출을 위한 인터페이스 메소드
        int Area();// 실제 도형에서 크기를 계산 호출을 위한 인터페이스 메소드
}
```

학과 ____________ 학번 ____________ 이름 ____________

**13. 12번 프로그램의 Shape(도형), Rectangle(사각형), Square(정사각형) 클래스를 이용하여 아래의 Main () 안에 5개짜리 Shape 배열을 생성하고 사각형과 정사각형 데이터를 추가한 후, foreach를 사용하여 Draw(그리기)와 Area(크기계산)을 출력하는 코드를 작성하시오. (10점)**

```
class ShapeTest
{
    static void Main() {

        // 5개짜리 Shape 배열 생성
        Shape [] dlist = new Shape[5];

        // 객체 추가
        dlist[0] = new Rectangle(4, 5);
        dlist[1] = new Square(7);
        dlist[2] = new Square(2);
        dlist[3] = new Rectangle(2, 3);
        dlist[4] = new Rectangle(1, 2);

        // foreach 사용하여 Draw와 Area 호출
        foreach (Shape d in dlist)
        {
                d.Draw();
                Console.WriteLine("Area={0}", d.Area());
        }

    }
}
```

**-끝-**