|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **윈도우프로그래밍\_과제\_5주** | **이름 :** |  | **학번 :** |  |

* **내용 점검**

|  |
| --- |
| 1. 델리게이트(위임형) 사용 예 |
| delegate void DelegateOne(); // 매개변수와 반환값이 없는 델리게이트 정의  delegate void DelegateTwo(int i); // 한 개의 매개변수와 반환값이 없는 델리게이트 정의  class DelegateClass{  public void MethodA() {  Console.WriteLine("In the DelegateClass.MethodA ...");  }  public void MethodB(int i) {  Console.WriteLine("DelegateClass.MethodB, i = " + i);  }  }  class DelegateCallApp {  public static void Main() {  //델리게이트할 메소드를 갖는 객체를 먼저 생성해야 함  DelegateClass obj = new DelegateClass();  DelegateOne d1 = new DelegateOne(obj.MethodA); //델리게이트 선언 및 생성  DelegateTwo d2 = new DelegateTwo(obj.MethodB);  d1(); // 델리게이트 객체를 통하여 메소드 MethodA() 호출  d2(10); // invoke MethodB(10) in DelegateClass  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 델리게이트(위임형) 사용 예 |
| delegate void MultiDelegate(); //매개변수와 반환값이 없는 델리게이트 정의  class DelegateClass {  public void MethodA() { Console.WriteLine("In the DelegateClass.MethodA ..."); }  public void MethodB() { Console.WriteLine("In the DelegateClass.MethodB ..."); }  public void MethodC() { Console.WriteLine("In the DelegateClass.MethodC ..."); }  }  class Program {  static void Main(string[] args) {  DelegateClass obj = new DelegateClass(); // 클래스객체생성  MultiDelegate dg1, dg2, dg3; // 델리게이트선언  dg1 = new MultiDelegate(obj.MethodA); // 델리게이트객체생성  dg2 = new MultiDelegate(obj.MethodB); // 델리게이트객체생성  dg3 = new MultiDelegate(obj.MethodC); // 델리게이트객체생성  dg1 = dg1 + dg2; dg1 += dg3; // 메소드추가  dg2 = dg1 - dg2; // 메소드제거  dg1(); Console.WriteLine("After dg1 call ...");  dg2(); Console.WriteLine("After dg2 call ...");  dg3();  }  } |
| **[실행결과]** |

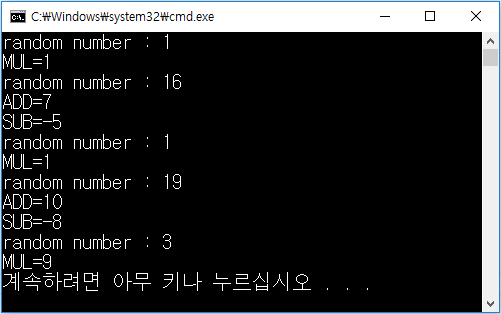
|  |
| --- |
| 1. 연산자 중복 사용 예 |
| class Complex {  private double realPart;            // 실수부          private double imagePart;          // 허수부          public Complex(double rVal, double iVal) {  realPart = rVal;  imagePart = iVal;  }  public static Complex operator +(Complex x1, Complex x2) { //연산자 중복              Complex x = new Complex(0, 0);  x.realPart = x1.realPart + x2.realPart;  x.imagePart = x1.imagePart + x2.imagePart;  return x;  }  override public string ToString() { return "(" + realPart + "," + imagePart + "i)"; }  }  class Program {  static void Main(string[] args) {  Complex c, c1, c2;  c1 = new Complex(1, 2);  c2 = new Complex(3, 4);  c = c1 + c2; //중복된 연산자 사용              Console.WriteLine(c1 + " + " + c2 + " = " + c);  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 상속 |
| class BaseClass {  public int a, b;  public BaseClass() { a = 1; b = 1; }  public BaseClass(int a, int b){ this.a = a; this.b = b;}  }  class DerivedClass : BaseClass {  public int c;  public DerivedClass(){ c = 1; }  //베이스 클래스의 중복된 생성자 호출  public DerivedClass(int a, int b, int c): **base(a, b)** {this.c = c; }}  class Program{  static void Main(string[] args){  DerivedClass obj1 = new DerivedClass();  DerivedClass obj2 = new DerivedClass(1, 2, 3);  Console.WriteLine("a = {0}, b = {1}, c = {2}", obj1.a, obj1.b, obj1.c);  Console.WriteLine("a = {0}, b = {1}, c = {2}", obj2.a, obj2.b, obj2.c);  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 메소드 재정의 |
| class BaseClass{  virtual public void MethodA(){  Console.WriteLine("Base MethodA");  }  virtual public void MethodB(){  Console.WriteLine("Base MethodB");  }  virtual public void MethodC(){  Console.WriteLine("Base MethodC");  }  }  class DerivedClass : BaseClass{  new public void MethodA(){  Console.WriteLine("Derived MethodA");  }  override public void MethodB(){  Console.WriteLine("Derived MethodB");  }  public void MethodC(){  Console.WriteLine("Derived MethodC");  }  }  class Program{  static void Main(string[] args){  BaseClass s = new DerivedClass();  s.MethodA(); //베이스 클래스 메소드 호출  s.MethodB(); //파생클래스 메소드 호출  s.MethodC(); //베이스 클래스 메소드 호출  }  } |
| **[실행결과]** |

* 프로그램 과제

1. 1에서 20사이의 난수를 5개 발생시켜 한 자리수 이면 곱을 구하고, 두 자리 수이면 한자리씩 분리하여 합과 차를 구하는 프로그램 (델리게이트를 통하여 메소드 호출)



* 난수 생성 방법

Random rn = new Random();

int r = rn.Next(1, 20);

// 정수형 매개변수 1개인 메소드를 호출할 델리게이트 선언 – 본인 작성

// 정수형 매개변수 2개인 메소드를 호출할 델리게이트 선언 – 본인 작성

class MyClass {

public void MUL(int x) // X\*X 계산 결과 출력하는 함수 – 본인 작성

public void ADD(int x, int y) // x+y 계산 결과 출력하는 함수 – 본인작성

public static void SUB(int x, int y) //x-y 계산 결과 출력하는 함수 – 본인작성

}

class MyClassTest{

static void Main(string[] args){

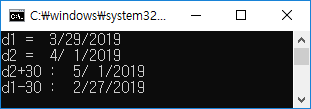
//델리게이트를 통하여 메소드 호출(5회 반복) – 본인작성

}

}

|  |
| --- |
| **프로그램 소스** |
|  |
| **[실행결과]** |

1. 연산자 중복 예제를 참고하여 다음과 같이 실행되는 프로그램을 완성하시오.



|  |
| --- |
| class Date {  private int day, month, year;  public Date(int mm, int dd, int yy){ // 생성자  day = dd;  month = mm;  year = yy;  }  public static Date operator +(Date d, int n) {// 날짜에 대한 + 연산 정의    }  public static Date operator -(Date d, int n){ // 날짜에 대한 - 연산 정의    }  override public string ToString(){ // mm/dd/yy  return string.Format("{0,2}/{1,2}/{2,2}", month, day, year);  }  }  **프로그램 소스** |
|  |
| **[실행결과]** |

1. 학생과 교직원 정보를 출력하는 프로그램을 제시된 조건에 따라 작성하시오.

* Person 클래스 선언
* 멤버변수 : 이름(string), 주민등록번호(string) - protected
* 생성자 : 매개변수로 멤버 초기화
* disPlay() 메소드 : 이름과 주민등록 번호 출력, 가상메소드
* Student 클래스 선언
* Person 클래스를 상속
* 멤버변수: 학번(string), 학과(string)
* 생성자: 매개변수로 멤버변수 초기화 - 상위 클래스 멤버도 초기화
* disPlay() 메소드 재정의 :

이름, 주민등록번호, 학번, 학과 출력

이름과 주민등록번호는 상위 클래스 메소드 사용

* Professor 클래스 선언
* Person 클래스 상속
* 멤버변수 : 사번(string), 전공(string)
* 생성자: 매개변수로 멤버변수 초기화 - 상위 클래스 멤버도 초기화
* disPlay() 메소드 재정의

이름과 주민등록번호,학번, 전공 출력

이름과 주민등록 번호는 상위 클래스 메소드 사용

* In\_PholyTest 클래스 선언
* 객체 확인을 위한 Main() 메소드 포함
* 입력 받은 숫자만큼 Person 타입의 객체 배열 생성
* 메뉴에서 1이면 Student 객체를 생성, 2이면 Professor 객체 생성, 입력 받은 데이터를 매개변수로 사용
* 객체 정보를 출력 – foreach 구문 사용할 것



|  |
| --- |
| **프로그램 소스** |
|  |
| **[실행결과]** |