|  |
| --- |
| 윈도우즈 프로그래밍 |

-과제-

4주차

20144638

이한범

# 1번문제

이름, 학번, 학년, 중간 점수, 기말 점수, 합계, 평균을 필드로 한 student 클래스 정의.

생성자, 합계를 구하는 메소드, 평균을 구하는 메소드, 중간점수 및 기말 점수를 할당하는 메소드를 만들고 테스트할 것.

1. 작성한 코드내용

# Program class (main)

|  |
| --- |
| namespace fourthWeek  {  class Program  {  static void Main(string[] args)  {  string info, info\_num;  string name, s\_num;  int grade;  Console.WriteLine("==================================");  Console.WriteLine(">>> 이름, 학번, 학년 입력");  info = Console.ReadLine();  name = info.Split(' ')[0];  s\_num = info.Split(' ')[1];  grade = Int32.Parse(info.Split(' ')[2]);  student stu = new student(name,s\_num,grade);  Console.WriteLine(">>> 중간, 기말 점수 입력");  info\_num = Console.ReadLine();    stu.Mid\_score = Int32.Parse(info\_num.Split(' ')[0]);  stu.Final\_score = Int32.Parse(info\_num.Split(' ')[1]);    Console.WriteLine("이름 : {0}, 학번 : {1}, 학년 : {2}", stu.getName(),  stu.getS\_num(), stu.getGrade());  Console.WriteLine("총점 : {0}, 평균 : {1}", stu.Sum(), stu.Avg());    }  }  }  } |

▲ 메인 클래스. split() 메서드를 통해 공백으로 문자열을 잘라주어 필드들을 나누어 정의시켜줌.

# student class

|  |
| --- |
| namespace fourthWeek  {  class student  {  //필드 선언  private string name;  private string s\_num;  private int grade;  private int mid\_score = 0;  private int final\_score = 0;  private double sum = 0;  private double avg = 0;  //접근자 설정자 정의  public string getName() { return name; }  public string getS\_num() { return s\_num; }  public int getGrade() { return grade; }  //중간, 기말 점수를 위한 프로퍼티  public int Mid\_score  {  get { return mid\_score; }  set { mid\_score = value; }  }  public int Final\_score  {  get { return final\_score; }  set { final\_score = value; }  }  //1-1  public student(string name, string s\_num, int grade)  {  this.name = name;  this.s\_num = s\_num;  this.grade = grade;  }  //1-2  public double Sum()  {  this.sum = this.mid\_score + this.final\_score;  return sum;  }  public double Avg()  {  this.avg = (this.mid\_score + this.final\_score) / 2;  return avg;  }  }  } |

▲student 클래스 정의. 생성자를 통해 필드들을 넣은 인수 값으로 초기화시켜주고, sum, avg 라는 합계를 구하는 메서드와 평균을 구하는 메서드를 선언.

2) 실행결과

|  |
| --- |
|  |

# 2번문제

이름, 학번, 학년을 "", '", 0으로 초기화 하는 디폴트 생성자 추가 및 소멸자를 작성하여 Destructor... 이라는 문장이 출력되게 할 것. 또한 학생 수를 입력받아 학생 수만큼 객체를 생성하여 총 학생 수 및 평균 중간, 기말 점수를 출력.

1. 작성한 코드내용

# Program class (main)

|  |
| --- |
| namespace fourthWeek  {  class Program  {  static void Main(string[] args)  {  string info, info\_num;  string name, s\_num;  int grade;  int loop;  double sum=0, avg=0;  //2번 문제  Console.Write("학생의 명 수를 입력하세요 : ");  loop = Int32.Parse(Console.ReadLine());  student[] stu = new student[loop];  for (int i = 0; i < loop; i++)  {  Console.WriteLine("==================================");  Console.WriteLine(">>> 이름, 학번, 학년 입력");  info = Console.ReadLine();  name = info.Split(' ')[0];  s\_num = info.Split(' ')[1];  grade = Int32.Parse(info.Split(' ')[2]);  stu[i] = new student(name, s\_num, grade);  Console.WriteLine(">>> 중간, 기말 점수 입력");  info\_num = Console.ReadLine();  stu[i].Mid\_score = Int32.Parse(info\_num.Split(' ')[0]);  stu[i].Final\_score = Int32.Parse(info\_num.Split(' ')[1]);  Console.WriteLine("이름 : {0}, 학번 : {1}, 학년 : {2}", stu[i].getName(),  stu[i].getS\_num(), stu[i].getGrade());  Console.WriteLine("총점 : {0}, 평균 : {1}", stu[i].Sum(), stu[i].Avg());  sum += stu[i].Mid\_score;  avg += stu[i].Final\_score;  }  Console.WriteLine("총 학생 수는 {0}명이며, 평균 중간 점수는 {1}점, 평균 기말 점수는 {2}점 입니다.",  loop, (sum/loop).ToString("N2"), (avg/loop).ToString("N2"));  }  }  } |

▲ 메인 클래스. 1번과 유사하나 입력받은 학생의 수만큼 반복. 마지막엔 총 평균을 출력.

# student class

|  |
| --- |
| namespace fourthWeek  {  class student  {  //필드 값 선언  private string name;  private string s\_num;  private int grade;  private int mid\_score = 0;  private int final\_score = 0;  private double sum = 0;  private double avg = 0;  //접근자 설정자 선언  public string getName() { return name; }  public string getS\_num() { return s\_num; }  public int getGrade() { return grade; }  //중간, 기말점수를 위한 프로퍼티  public int Mid\_score  {  get { return mid\_score; }  set { mid\_score = value; }  }  public int Final\_score  {  get { return final\_score; }  set { final\_score = value; }  }  //2-1  public student()  {  this.name = "";  this.s\_num = "";  this.grade = 0;  }  //1-1  public student(string name, string s\_num, int grade)  {  this.name = name;  this.s\_num = s\_num;  this.grade = grade;  }  //2-2  ~student()  {  Console.WriteLine("Destructor...");  }  //1-2  public double Sum()  {  this.sum = this.mid\_score + this.final\_score;  return sum;  }  public double Avg()  {  this.avg = (this.mid\_score + this.final\_score) / 2;  return avg;  }  }  } |

▲student 클래스 정의. 생성자를 통해 필드들을 넣은 인수 값으로 초기화시켜주고, sum, avg 라는 합계를 구하는 메서드와 평균을 구하는 메서드를 선언. 소멸자를 따로 선언해주었으며 디폴트 생성자 또한 만들어주었음.

2) 실행결과

|  |
| --- |
|  |

# 3번문제

사칙연산이 가능한 계산기 클래스 Calc 선언.

split을 이용하여 연산자 / 피연산자 구분 및 생성자를 이용한 초기화 구현. 또한 사칙연산의 결과를 저장하는 필드를 정의. switch를 통해 결과를 출력하는 result 메소드 작성.

1. 작성한 코드내용

# Program class (main)

|  |
| --- |
| namespace fourthWeek  {  class Program  {  static void Main(string[] args)  {  String s;  Calc a;  Console.WriteLine("수식을 입력하세요 : ");  s = Console.ReadLine();  a = new Calc(s);  a.result();  }  }  } |

▲ 메인 클래스. 사용자가 문자열 입력 후 Calc 객체의 인수로 사용.

# Calc class

|  |
| --- |
| namespace fourthWeek  {  class Calc  {  private String res;  private String[] numacial;  public Calc(String num)  {  numacial = num.Split(' ');  }  //연산자 우선순위 구현 x  public void result()  {  //switch문을 통해서 쪼개진 단어중에 연산자들만을 구분.  for (int i =0;i<numacial.Length;i++)  {  switch (numacial[i])  {  case "+":  res = (Int32.Parse(numacial[i - 1]) + Int32.Parse(numacial[i + 1])).ToString();  numacial[i + 1] = res;  break;  case "\*":  res = (Int32.Parse(numacial[i - 1]) \* Int32.Parse(numacial[i + 1])).ToString();  numacial[i + 1] = res;  break;  case "-":  res = (Int32.Parse(numacial[i - 1]) - Int32.Parse(numacial[i + 1])).ToString();  numacial[i + 1] = res;  break;  case "/":  res = (Int32.Parse(numacial[i - 1]) - Int32.Parse(numacial[i + 1])).ToString();  numacial[i + 1] = res;  break;  default:  break;  }  }  Console.WriteLine("답 = " + res);  }  }  } |

▲문자열을 받아 연산자로 나누어 연산을 구현하기위해 선언한 클래스 Calc. switch문을 통하여 숫자값이 아닌 +, -, \*, / 만을 구분하여, 연산자의 앞 오프셋과 뒤 오프셋 (피연산자)을 연산.

2) 실행결과

|  |
| --- |
|  |

# 4번문제

클래스를 사용하여 주차장 구현. 주차장 클래스의 필드는 차량 번호, 모델명, 들어온 시간, 나간 시간, 요금으로 구성.

car\_in 메서드 구현. 이를 통해 사용자로부터 차량 번호, 모델명, 들어온 시간을 가진 객체를 반환시킨다. 또한 car\_out 함수를 통해 나간시간을 기록 후 요금을 계산.

1. 작성한 코드내용

# last class (main)

|  |
| --- |
| namespace fourthWeek  {  class last  {  static void Main(string[] args)  {  List<car> list = new List<car>(); //car 리스트  car one = new car();  String info; //사용자 입력  int n,i=1;  int outtime;  //무한 반복. 단 0,1외의 다른 값이 입력되면 탈출  while (i == 1)  {  Console.Write("Enter number(0 : car in, 1 : car out) : ");  n = Int32.Parse(Console.ReadLine());  switch (n) {  case 0: //car in  Console.Write("enter car information(5885,k5,1): ");  info = Console.ReadLine();  list.Add(one.car\_in(info.Split(',')[0], info.Split(',')[1],  Int32.Parse(info.Split(',')[2])));  break;  case 1: //car out  Console.Write("enter car number: ");  info = Console.ReadLine();  Console.Write("enter exit time: ");  outtime = Int32.Parse(Console.ReadLine());  for (int j = 0; j < list.Count; j++)  {  if (list[j].getCarNum() == info) {  list[j].setOut\_hour(outtime);  list[j].car\_out(outtime);  list.RemoveAt(j);  }  }    break;  default: //예외  i = 0;  break;  }  }  Console.WriteLine("exit");  }  }  } |

▲ 메인 클래스. car 리스트를 만들어주어 car 객체들을 이 리스트에 추가시켜줌 (add 메서드 이용) 그 후 switch 문을 통해 car\_in과 car\_out을 구분시켜주었음. 0, 1 이외의 값이 입력될 경우 반복문을 탈출하여 exit를 화면에 출력.

# car class

|  |
| --- |
| namespace fourthWeek  {  class car  {  private String carNum;  private String model;  private int in\_hour, out\_hour;  private int fee;  //각종 접근자 설정자  public String getCarNum() { return this.carNum; }  public String getModel() { return this.model; }  public int getInHour() { return this.in\_hour; }  public int getOutHour() { return this.out\_hour; }  public void setFee(int fee) { this.fee = fee; }  public void setOut\_hour(int out\_hour) { this.out\_hour = out\_hour; }  public car()  {  this.carNum = "init";  this.model = "init";  this.in\_hour = 0;  }  // car in. 생성자처럼 받으나, 반환값이 car  public car car\_in(String carNum, String model, int h)  {  car n = new car();  n.carNum = carNum;  n.model = model;  n.in\_hour = h;  return n;  }  // car out. 시간값을 받아서 현재 객체의 in hour와 비교.  public void car\_out(int outtime)  {  this.fee = 3 \* (outtime-this.in\_hour);  Console.WriteLine("Enter exit time : {0}hour, car has been out : {1}hour", this.getInHour(), outtime);  Console.WriteLine("시간당 요금 3$, 따라서 요금 {0}$", fee);  }  /\*  public void display(car a)  {  Console.WriteLine(a.getCarNum() + "," + a.getModel() + " " + a.getInHour());  }  \*/  }  } |

▲ 자동차의 설정 값들을 가지는 클래스 car 선언. car\_in, car\_out 메서드를 가지고있으며 출력값을 임의로 확인하기위해 display 메서드 정의. car\_in부분은 마치 생성자처럼 값을 받았으나, car 객체를 반환시켜주었다.

car\_out 부분의 경우는 나가는 시간을 받아 사용자가 입력한 차량 번호를 가지는 car 객체의 들어온 시간과 비교하여 요금을 계산한다.

2) 실행결과

|  |
| --- |
|  |