|  |
| --- |
| 윈도우즈 프로그래밍 |

-과제-

분수 계산 프로그래밍

20144638

이한범

# 코드 설계

두 개의 클래스를 정의하였다. 하나는 Main을 정의한 Program이라는 클래스이며, 다른 하나는 실제 분수를 계산하기 위한 필드와 메서드를 가지고있는 fraction 클래스이다.

1) 작성한 코드내용 : fraction

|  |
| --- |
| class fraction  {  private int numerator;  private int denominator;  public fraction(int num, int den)  {  this.numerator = num;  this.denominator = den;  }  public int getN() { return numerator; }  public int getD() { return denominator; }  public void setN(int num) { this.numerator = num; }  public void setD(int num) { this.denominator = num; }  override public String ToString() {  return (this.numerator + "/" + this.denominator);  } |

▲ 분자를 나타내는 필드와 분모를 나타내는 필드를 정의.

생성자에 인수 값을 넣어줌으로 써 fraction 객체의 분자, 분모값을 정의해 줄 수 있다. 또한 접근자와 설정자를 통하여 접근이 가능하며, 출력시에는 재정의한 ToString 메서드를 통해 분자/분모 값을 화면에 출력한다.

|  |
| --- |
| public static fraction operator+(fraction f1, fraction f2)  {  fraction dumy = new fraction(0,0);  dumy.numerator = f1.numerator \* f2.denominator + f1.denominator \* f2.numerator;  dumy.denominator = f1.denominator \* f2.denominator;  dumy.simple();  return dumy;  }  public static fraction operator -(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.denominator - f1.denominator \* f2.numerator;  dumy.denominator = f1.denominator \* f2.denominator;  dumy.simple();  return dumy;  }  public static fraction operator \*(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.numerator;  dumy.denominator = f1.denominator \* f2.denominator;  dumy.simple();  return dumy;  }  public static fraction operator /(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.denominator;  dumy.denominator = f1.denominator \* f2.numerator;  dumy.simple();  return dumy;  } |

▲연산자 오버로딩 부분이다.

|  |
| --- |
| public int gcd(int u, int v)  {  if (v == 0) return u;  else return gcd(v, u%v);  }  public fraction sign()  {  if (Math.Sign(this.numerator) == Math.Sign(this.denominator))  {  Math.Abs(this.numerator);  Math.Abs(this.denominator);  }  else if(Math.Sign(this.numerator) != Math.Sign(this.denominator))  {  this.numerator = Math.Abs(this.numerator) \* -1;  this.denominator = Math.Abs(this.denominator);  }  return this;  }  public fraction simple()  {  int simple\_fraction = gcd(Math.Abs(this.numerator), Math.Abs(this.denominator));  this.numerator = this.numerator / simple\_fraction;  this.denominator = this.denominator / simple\_fraction;  this.sign();  return this;  } |

▲부호값 설정 및 최대공약수를 구하는 알고리즘이 포함된 메서드들

2) 작성한 코드내용 : Program : main

|  |
| --- |
| namespace ThirdWeek  {  class Program  {  static void Main(string[] args)  {  fraction a = new fraction(3, 6); // 3/6  fraction b = new fraction(4, 5); // 4/5  fraction result = new fraction(0,0);  result = a + b;  result.getFraction();  result = a - b;  result.getFraction();    result = a \* b;  result.getFraction();  result = a / b;  result.getFraction();  }  }  } |

▲두 개의 분수와 그 결과값을 저장해 줄 fraction 객체 생성.

연산자 오버로딩을 통해서 위의 코드처럼 fraction 객체사이에 ' + ' 연산자 하나만 넣어주어도 두 객체의 분자 / 분모 값이 더해지는 연산을 수행할 수 있도록 해주었다.

# 문제 풀이

Q.1) 한 개의 정수를 받아 초기화 하는 생성자 작성

Q.2) 두 개의 정수를 받아 초기화 하는 생성자 작성

ㄱ) fraction 클래스

|  |
| --- |
| namespace ThirdWeek  {  class fraction  {  private int numerator;  private int denominator;  // Q.1  public fraction(int num)  {  this.numerator = num;  this.denominator = num;  }  // Q.2  public fraction(int num, int den)  {  this.numerator = num;  this.denominator = den;  }  public int getN() { return numerator; }  public int getD() { return denominator; }  public void setN(int num) { this.numerator = num; }  public void setD(int num) { this.denominator = num; } |

ㄴ) Program (Main) 클래스

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  fraction a = new fraction(2);  fraction b = new fraction(2, 7);  Console.WriteLine(a.getN() + "/" + a.getD());  Console.WriteLine(b.getN() + "/" + b.getD());  }  } |

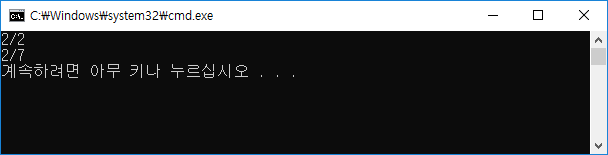
Q.1, Q.2) 설명

fraction 클래스에 생성자를 정의. 매개변수가 1개인 것과 2개인 것을 중복정의. 생성자의 경우 인수의 개수가 다르면 여러개를 정의 가능.

Q.1, Q.2) 실행



▲ 인수 1개를 써서 만든 객체와 2개를 써서 만든 객체



Q.3) 하나의 분수를 분자 / 분모 형태로 변환하는 ToString 메소드 작성.

ㄱ) fraction 클래스

|  |
| --- |
| override public String ToString() {  return (this.numerator + "/" + this.denominator);  } |

ㄴ) Program (Main) 클래스

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  fraction a = new fraction(2);  fraction b = new fraction(6, 12);  Console.WriteLine(b.ToString());  }  } |

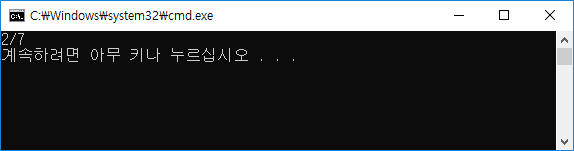
Q.3) 설명

fraction 클래스에 ToString 메서드 정의. 단, ToString 메서드의 경우 Object라는 최상위 클래스에 정의되어있는 메서드중 하나이기 때문에 override가 필요하다. 따라서 메서드에서 분수값으로 표현되는 String 값을 반환.

Q.3) 실행



▲ 인수 2개를 써서 만든 객체 2/7 을 표현한다.



Q.4) 최대 공약수를 구하는 메서드와 기약 분수로 만드는 메서드 작성.

ㄱ) fraction 클래스

|  |
| --- |
| public int gcd(int u, int v)  {  if (v == 0) return u;  else return gcd(v, u % v);  }  public fraction sign()  {  if (Math.Sign(this.numerator) == Math.Sign(this.denominator))  {  Math.Abs(this.numerator);  Math.Abs(this.denominator);  }  else if (Math.Sign(this.numerator) != Math.Sign(this.denominator))  {  this.numerator = Math.Abs(this.numerator) \* -1;  this.denominator = Math.Abs(this.denominator);  }  return this;  }  public fraction simple()  {  int simple\_fraction = gcd(Math.Abs(this.numerator), Math.Abs(this.denominator));  this.numerator = this.numerator / simple\_fraction;  this.denominator = this.denominator / simple\_fraction;  this.sign();  return this;  } |

ㄴ) Program (Main) 클래스

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  fraction a = new fraction(2);  fraction b = new fraction(6, 12);  b.simple();  Console.WriteLine(b.ToString());  }  } |

Q.4) 설명

gcd : 유클리드 알고리즘을 사용하여 최대공약수를 구하는 메서드

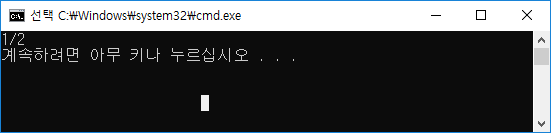
sign : 부호값을 정돈해주는 메서드. Math.Sign / Math.Abs 메서드를 사용하여 부호값을 추출, 절대값 설정을 시켜주었다. 기약분수화 시킨 객체의 분자/분모가 가진 부호를 정돈시켜준다.

simple : 기약분수로 만들어주는 메서드. 분자 / 분모의 절대값을 유클리드 알고리즘에 넣어 최대공약수를 구하고, 이 후 부호값을 반환시켜 최종적인 값을 fraction 자료형으로 반환.

Q.4) 실행

fraction b = new fraction(6, 12);

▲ 6/12 라는 분수 값을 축약.



Q.5) 분수에 대한 4칙연산을 수행하는 메소드 작성

\* 따로 메서드작성을 하지않고 연산자를 오버로딩 하였음.

ㄱ) fraction 클래스

|  |
| --- |
| public static fraction operator +(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.denominator + f1.denominator \* f2.numerator;  dumy.denominator = f1.denominator \* f2.denominator;  dumy.simple();  return dumy;  }  public static fraction operator -(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.denominator - f1.denominator \* f2.numerator;  dumy.denominator = f1.denominator \* f2.denominator;  dumy.simple();  return dumy;  }  public static fraction operator \*(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.numerator;  dumy.denominator = f1.denominator \* f2.denominator;  dumy.simple();  return dumy;  }  public static fraction operator /(fraction f1, fraction f2)  {  fraction dumy = new fraction(0, 0);  dumy.numerator = f1.numerator \* f2.denominator;  dumy.denominator = f1.denominator \* f2.numerator;  dumy.simple();  return dumy;  } |

ㄴ) Program (Main) 클래스

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  fraction a = new fraction(4, 5);  fraction b = new fraction(5, 8);  fraction result = new fraction(0,0);  result = a + b;  Console.WriteLine(result.ToString());    result = a - b;  Console.WriteLine(result.ToString());  result = a \* b;  Console.WriteLine(result.ToString());  result = a / b;  Console.WriteLine(result.ToString());  }  } |

Q.5) 설명

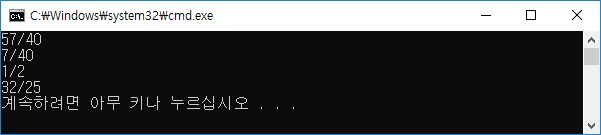
덧셈, 뺄셈, 곱셈, 나눗셈 이렇게 4가지의 연산자를 오버로딩해 주었으며, 새로운 더미 fraction 객체를 만들어 주어 연산을 해준 값을 더미 객체에 저장 후, 더미 객체를 반환시켜 주었다.

Q.5, Q.6) 실행

fraction a = new fraction(4, 5);

fraction b = new fraction(5, 8);

▲ 4/5 와 5/8 이라는 분수값을 연산시켜주었다.

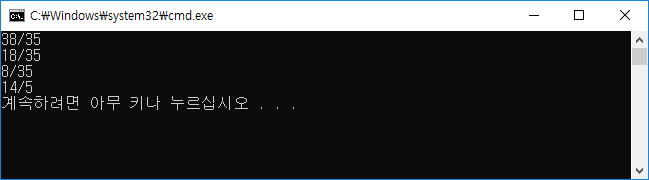


# 전체 코드 실행

// 연산 결과

fraction a = new fraction(4, 5); // 3/5

fraction b = new fraction(2, 7); // 2/3



// 만일 연산 내용 중 음수값이 나오는 경우

fraction a = new fraction(3, 6); // 3/5

fraction b = new fraction(4, 5); // 2/3

