|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **윈도우프로그래밍\_과제\_6주** | **이름 :** | **이갑성** | **학번 :** | **20135151** |

* **내용 점검**

|  |
| --- |
| 1. 제네릭 클래스 사용 예 |
| class SimpleGeneric<T> {  private T[] values;  private int index;  public SimpleGeneric(int len) { // Constructor  values = new T[len];  index = 0;  }  public void Add(params T[] args){  foreach (T e in args)  values[index++] = e;  }  public void Print(){  foreach (T e in values)  Console.Write(e + " ");  Console.WriteLine();  }  }  class Program{  static void Main(string[] args){  SimpleGeneric<Int32> gInteger = new SimpleGeneric<Int32>(10);  SimpleGeneric<Double> gDouble = new SimpleGeneric<Double>(10);  gInteger.Add(1, 2);  gInteger.Add(1, 2, 3, 4, 5, 6, 7);  gInteger.Add(0);  gInteger.Print();  gDouble.Add(10.0, 20.0, 30.0);  gDouble.Print();  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 제네릭 메소드 사용 예 |
| class GenericMethodApp{  static void Swap<T>(ref T x, ref T y) {  T temp;  temp = x; x = y; y = temp;  }  public static void Main() {  int a = 1, b = 2; double c = 1.5, d = 2.5;  Console.WriteLine("Before: a = {0}, b = {1}", a, b);  Swap(ref a, ref b);       // 정수형 변수로 호출, 타입 생략 가능            Console.WriteLine(" After: a = {0}, b = {1}", a, b);  Console.WriteLine("Before: c = {0}, d = {1}", c, d);  Swap<double>(ref c, ref d);    // 실수형 변수로 호출            Console.WriteLine(" After: c = {0}, d = {1}", c, d);  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 제네릭 인터페이스 사용 예 |
| interface IGenericInterface<T> {  void SetValue(T x);  string GetValueType();  }  class GenericClass<T> : IGenericInterface<T> {  private T value;  public void SetValue(T x){  value = x;  }  public String GetValueType(){  return value.GetType().ToString(); //GetType() :자료형 반환  }  }  public class GenericInterfaceApp{  public static void Main(){  GenericClass<Int32> gInteger = new GenericClass<Int32>();  GenericClass<String> gString = new GenericClass<String>();  gInteger.SetValue(10);  gString.SetValue("Text");  Console.WriteLine(gInteger.GetValueType());  Console.WriteLine(gString.GetValueType());  }  } |
| **[실행결과]** |

* 프로그램 과제

1. 교재 260p 5.8

|  |
| --- |
| **프로그램 소스** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace HomeWork  {  abstract class Figure  {  public abstract void Area();  public abstract void Girth();  public abstract void Draw();  }  class Rectangle:Figure {  int a, b;  public Rectangle(int a, int b)  {  this.a = a;  this.b = b;  }  public override void Area() {  Console.WriteLine("사각형의 넓이: {0}", a \* b);  }  public override void Girth()  {  Console.WriteLine("사각형의 둘레: {0}", 2 \* (a + b));  }  public override void Draw()  {  Console.WriteLine("사각형을 그렸습니다. ㅁ");  }  }  class Circle : Figure {  double r;  public Circle(double r) {  this.r = r;  }  public override void Area()  {  Console.WriteLine("원의 넓이: {0}", 3.14 \* r \* r);  }  public override void Girth()  {  Console.WriteLine("원의 둘레: {0}", 2 \* 3.14 \* r);  }  public override void Draw()  {  Console.WriteLine("원을 그렸습니다. O");  }  }  class Program  {  static void Main(string[] args)  {  Rectangle rec = new Rectangle(10, 5);  rec.Area(); //사각형의 넓이  rec.Girth(); //사각형의 둘레  rec.Draw(); //사각형을 출력  Circle cir = new Circle(1.5);  cir.Area(); //원의 넓이  cir.Girth(); //원의 둘레  cir.Draw(); //원을 출력  }  }  } |
| **[실행결과]** |

1. 교재 318p 6.11

|  |
| --- |
| **프로그램 소스**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace HomeWork02  {  class ISwap<T> {  public T x, y;  public void swap() {  T temp;  temp = x; x = y; y = temp;  }  }  public class ExerciseCh6\_11 {  public static void Main(string[] args)  {  ISwap<Int32> i = new ISwap<Int32>();  i.x = 1; i.y = 2;  Console.WriteLine("x: " + i.x + " y: " + i.y);  i.swap();  Console.WriteLine("x: " + i.x + " y: " + i.y);  ISwap<Double> d = new ISwap<Double>();  d.x = 1.0; d.y = 2.0;  Console.WriteLine("x: " + d.x + " y: " + d.y);  d.swap();  Console.WriteLine("x: " + d.x + " y: " + d.y);  }  }  } |
|  |
| **[실행결과]** |

1. 다음과 같은 인터페이스를 제네릭을 이용하여 다양한 형에서 동작이 가능하도록 구현하는 클래스를 작성하고 테스트 하시오. 단, 스택의 크기는 100.

interface IOperation {

void Insert(string str); //매개변수로 받은 str을 스택에 삽입(push)

string Delete(); //스택 탑 원소 제거(pop) 후 반환

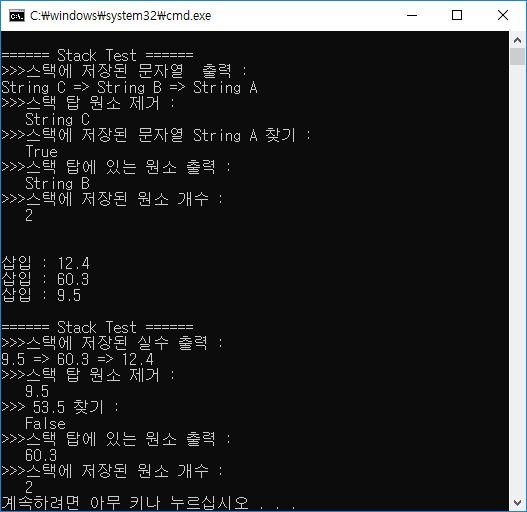
bool Search(string str); //매개변수로 받은 str 원소의 존재 여부 반환

string GetCurrentElt(); //현재 스택 탑에 있는 원소 반환

int NumOfElements(); //스택에 존재하는 원소 개수 반환

void PrintStack(); //스택에 저장된 모든 원소 출력

}



|  |
| --- |
| **프로그램 소스** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace HomeWork03  {  interface IOperation<T>  {  void Insert(T obj); //매개변수로 받은 str을 스택에 삽입(push)  T Delete(); //스택 탑 원소 제거(pop) 후 반환  bool Search(T obj); //매개변수로 받은 str 원소의 존재 여부 반환  T GetCurrentElt(); //현재 스택 탑에 있는 원소 반환  int NumOfElements(); //스택에 존재하는 원소 개수 반환  void PrintStack(); //스택에 저장된 모든 원소 출력  }  class Stack<T> : IOperation<T>  {  int top; //위치 지정해주는 Top  T[] arr;  public Stack()  {  this.top = -1;  arr = new T[100]; //stack 배열 생성  }  public void Insert(T obj) //매개변수로 받은 str을 스택에 삽입(push)  {  arr[++top] = obj;  }  public T Delete() //스택 탑 원소 제거(pop) 후 반환  {  if (top == -1)  {  Console.WriteLine("값이 비어있습니다.");  }  T value = arr[top];  arr[top] = default(T);  top--;  Console.WriteLine("{0} 제거가 되었습니다.", value);  return value;  }  public bool Search(T obj)  {    for (int i = 0; i <= top; i++)  {  if (arr[i].Equals(obj))  {  return true;  }  }    return false;  }  public T GetCurrentElt()  {  if (top == -1)  {  Console.WriteLine("값이 비어있습니다.");  }  T value = arr[top];  return value;  }  public int NumOfElements()  {  return top+1;  }  public void PrintStack()  {  for (int i = top; i >= 0; i--)  {  Console.Write("{0}", arr[i]);  if (!(i == 0))  Console.Write(" => ");  }  Console.WriteLine();  }  }  class Program  {  static void Main(string[] args)  {  Console.WriteLine("============== Stack Test ==============");  Stack<String> s1 = new Stack<string>();  Console.WriteLine("스택에 원소 삽입");  Console.WriteLine("삽입: String A \n삽입: String B\n삽입: String B");  s1.Insert("String A");  s1.Insert("String B");  s1.Insert("String C");  Console.Write("스택에 저장된 문자열 출력>>> ");  s1.PrintStack();  Console.WriteLine();  Console.Write("스택 탑 원소 제거>>> ");  s1.Delete();  Console.WriteLine();  Console.Write("스택에 저장된 문자열 String B 찾기>>> ");  Console.WriteLine("{0}",s1.Search("String B"));  Console.WriteLine();  Console.Write("스택 탑에 있는 원소 출력>>> ");  Console.WriteLine("{0}",s1.GetCurrentElt());  Console.WriteLine();  Console.Write("스택에 저장된 원소 개수>>> ");  Console.WriteLine("{0}",s1.NumOfElements());  Console.WriteLine();  Console.WriteLine("============== Stack Test ==============");  Stack<Double> s2 = new Stack<Double>();  Console.WriteLine("스택에 원소 삽입");  Console.WriteLine("삽입: 3.14 \n삽입: 65.5\n삽입: 12.3");  s2.Insert(3.14);  s2.Insert(65.5);  s2.Insert(12.3);  Console.Write("스택에 저장된 문자열 출력>>> ");  s2.PrintStack();  Console.WriteLine();  Console.Write("스택 탑 원소 제거>>> ");  s2.Delete();  Console.WriteLine();  Console.Write("스택에 저장된 문자열 3.14 찾기>>> ");  Console.WriteLine("{0}", s2.Search(3.14));  Console.WriteLine();  Console.Write("스택 탑에 있는 원소 출력>>> ");  Console.WriteLine("{0}", s2.GetCurrentElt());  Console.WriteLine();  Console.Write("스택에 저장된 원소 개수>>> ");  Console.WriteLine("{0}", s2.NumOfElements());  Console.WriteLine();  }  }  } |
| **[실행결과]** |

1. 제시된 결과처럼 실행될 수 있도록 3)번 프로그램을 수정하세요.

class Product{

//멤버 변수 : 제목, 연도

//생성자 : 매개변수로 받은 값을 멤버 변수 초기화

//객체 내용을 문자열로 반환하는 ToString() 메소드 재정의

}

class Book //Product 상속

{

//멤버 변수 : 저자,출판사

//생성자 : 매개변수로 받은 값을 멤버 변수초기화

//객체 내용을 문자열로 반환하는 ToString() 메소드 재정의

}

class CD //Product 상속

{

//멤버 변수 : 수록곡

//생성자 : 매개변수로 받은 값을 멤버 변수 초기화

//객체 내용을 문자열로 반환하는 ToString() 메소드 재정의

}

class ProductManage //제네릭 클래스, Product 파생 클래스로 type parameter 제한

{

public ProductManage(int size) // 생성자, 매개변수 값 만큼 type parameter 배열 생성

{

}

public void Insert() // 매개변수로 받은 type parameter의 객체를 배열에 저장

{

}

public void display() // 배열 원소 출력, 매개변수 없음

{

}

}

class Program {

static void Main(string[] args) {

ProductManage<Book> book = new ProductManage<Book>(20);

ProductManage<CD> cd = new ProductManage<CD>(20);

bool flag = true;

while (flag) {

Console.Write("1. Book 2.CD 3.Exit \_\_\_\_\_\_ ");

switch ()

{

case 1:

break;

case 2:

break;

case 3:

flag = false;

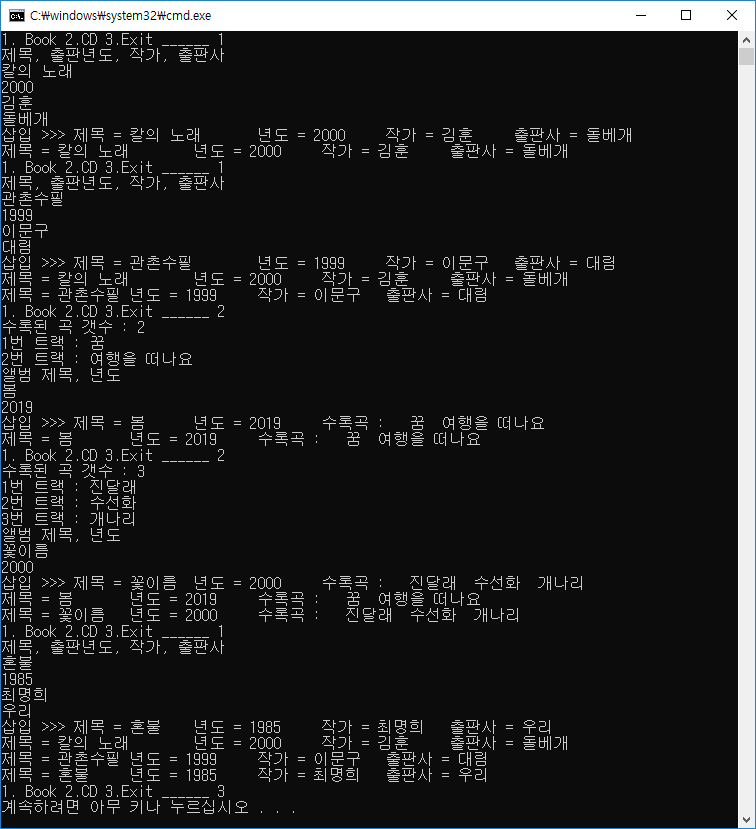
break;

}

}

}

}



Hint 가변매개변수 (params 이용) -> 매개변수가 유동적

|  |
| --- |
| **프로그램 소스** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace HomeWork04  {  class Product  {  string title; //멤버 변수 : 제목, 연도  int year;  public Product(string title, int year) //생성자 : 매개변수로 받은 값을 멤버 변수 초기화  {  this.title = title;  this.year = year;  }  virtual public string ToString()  { //객체 내용을 문자열로 반환하는 ToString() 메소드 재정의  string str;  string strYear = year.ToString();  str = " 제목 = " + title + ", 년도 = " + strYear;  return str;  }  }  class Book : Product //Product 상속  {  string writer;//멤버 변수 : 저자,출판사  string publisher;  public Book(string name, int year, string producer, string publisher) : base(name, year)  { //생성자 : 매개변수로 받은 값을 멤버 변수초기화  this.writer = producer;  this.publisher = publisher;  }  public override string ToString()  { //객체 내용을 문자열로 반환하는 ToString() 메소드 재정의  string str = base.ToString(); //제목과 년도가 들어가있음  str += ", 작가 = " + writer + ", 출판사 = " + publisher; //작가와 출판사를 연결해줌  return str;  }  }  class CD : Product //Product 상속  {  string[] song;//멤버 변수 : 수록곡  public CD(string title, int year, params string[] s) :base(title, year) //생성자 : 매개변수로 받은 값을 멤버 변수 초기화  {  song = s;  }  public override string ToString() //객체 내용을 문자열로 반환하는 ToString() 메소드 재정의  {  string str = base.ToString(); //제목과 년도가 들어가 있음  str += " 수록곡 : ";  for (int i = 0; i < song.Length; i++)  {  str += song[i];  str += " ";  }  return str;    }  }  class ProductManage<T> where T : Product //제네릭 클래스, Product 파생 클래스로 type parameter 제한  {  T[] type;  int index;  public ProductManage(int size) // 생성자, 매개변수 값 만큼 type parameter 배열 생성  {  type = new T[size]; //Book 객체로 받았을때 book에 대한 객체 배열을 생성 or CD의 객체로 받았을때는 cd에 대한 객체 배열 생성  int index = 0;  }  public void Insert(T obj) // 매개변수로 받은 type parameter의 객체를 배열에 저장  {  type[index] = obj;  index++;  Console.WriteLine("삽입 >>>" + obj.ToString());  }  public void display() // 배열 원소 출력, 매개변수 없음  {  for (int i = 0; i < index; i++)  {  Console.WriteLine("{0}",type[i].ToString());  }  }  }  class Program  {  static void Main(string[] args)  {  int songCnt;  string title, producer, publisher;  int year;  string songTitle;  ProductManage<Book> book = new ProductManage<Book>(20);  ProductManage<CD> cd = new ProductManage<CD>(20);  bool flag = true;  while (flag)  {  Console.Write("1. Book 2.CD 3.Exit \_\_\_\_\_\_ ");  switch (Console.ReadLine())  {  case "1":  Console.WriteLine("제목, 출판년도, 작가, 출판사");  title = Console.ReadLine();  year = int.Parse(Console.ReadLine());  producer = Console.ReadLine();  publisher = Console.ReadLine();  Book b1 = new Book(title, year, producer, publisher);  book.Insert(b1);  book.display();  break;  case "2":  Console.Write("수록된 곡의 개수: ");  songCnt = int.Parse(Console.ReadLine());  string[] arr = new string[songCnt];  for (int i = 0; i < arr.Length; i++) {  Console.Write("{0}번 트랙: ", (i + 1));  arr[i] = Console.ReadLine();  }  Console.WriteLine("앨범 제목, 년도");  songTitle = Console.ReadLine();  year = int.Parse(Console.ReadLine());  CD c1 = new CD(songTitle, year, arr);  cd.Insert(c1);  cd.display();  break;  case "3":  flag = false;  break;  }  }  }  }  } |
| **[실행결과]** |