|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Text;  using System.IO;  using System.Reflection;  using System.Linq;  namespace Экзамен  {  public class AnyDay  {  private int day;  private int mounth;  public int Day  {  get => day;  set  {  if(value >= 1 && value <= 31)  {  day = value;  return;  }  day = 31;  using (StreamWriter writer = new StreamWriter("text.txt", true))  {  writer.WriteLine("Попытка установки неверного дня в объекте");  }  }  }  public int Mounth  {  get => mounth;  set  {  if (value >= 1 && value <= 12)  {  mounth = value;  return;  }  mounth = 12;  using (StreamWriter writer = new StreamWriter("text.txt", true))  {  writer.WriteLine("Попытка установки неверного месяца в объекте");  }  }  }  public override bool Equals(object obj)  {  if(obj is AnyDay day1)  {  return day1.day == this.day && day1.mounth == this.mounth;  }  return false;  }  public void Print()  {  Console.WriteLine(this.day + " " + this.mounth);  }  public static bool operator ==(AnyDay day1, AnyDay day2)  {  return day1.Equals(day2);  }  public static bool operator !=(AnyDay day1, AnyDay day2)  {  return !day1.Equals(day2);  }  public static bool operator >=(AnyDay day1, AnyDay day2)  {  if(day1.mounth == day2.mounth)  {  return day1.day >= day2.day;  }  return (day1.mounth >= day2.mounth);  }  public static bool operator <=(AnyDay day1, AnyDay day2)  {  if (day1.mounth == day2.mounth)  {  return day1.day <= day2.day;  }  return (day1.mounth <= day2.mounth);  }  }  public static class Reflector  {  public static void GetConstructors(Type Class)  {  ConstructorInfo[] ctor = Class.GetConstructors();  for (int i = 0; i < ctor.Length; i++)  {  Console.WriteLine(ctor[i]);  }  }  public static void GetProperties(Type Class)  {  PropertyInfo[] properties = Class.GetProperties();  for (int i = 0; i < properties.Length; i++)  {  Console.WriteLine(properties[i]);  }  }  }  public class Holidays<T> where T: AnyDay  {  public List<T> list = new List<T>();    public void GetHolidays()  {  var Selected = from selected in this.list  where selected.Mounth == 12 ||  selected.Mounth == 1 ||  selected.Mounth == 2  orderby selected.Mounth, selected.Day  select selected;  foreach (var item in Selected)  {  Console.WriteLine(item.Day + " " + item.Mounth);  }    }  }  public class Decorator : AnyDay  {  public new void Print()  {  Console.Write(this.Day + " " + this.Mounth + " ");  switch (this.Mounth)  {  case 1:  case 2:  case 12:  {  Console.WriteLine("Зима");  break;  }  case 3:  case 4:  case 5:  {  Console.WriteLine("Весна");  break;  }  case 6:  case 7:  case 8:  {  Console.WriteLine("Лето");  break;  }  case 9:  case 10:  case 11:  {  Console.WriteLine("Осень");  break;  }  default:  break;  }  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Text;  using System.IO;  using System.Reflection;  using System.Linq;  namespace Экзамен  {  public enum WeekDay  {  Monday = 1,  Tuesday,  Wednesday,  Thursday,  Friday,  Saturday,  Sunday  }  public class AnyDayTimeTable  {  private WeekDay day;  public WeekDay Day  {  get => day;  set  {  if(value >= WeekDay.Monday && value <= WeekDay.Sunday)  {  day = value;  return;  }  throw new Exception(message: "Неправильно введен день");  }  }    public override bool Equals(object obj)  {  if(obj is AnyDayTimeTable day1)  {  return day1.day == this.day;  }  return false;  }  public static bool operator ==(AnyDayTimeTable day1, AnyDayTimeTable day2)  {  return day1.Equals(day2);  }  public static bool operator !=(AnyDayTimeTable day1, AnyDayTimeTable day2)  {  return !day1.Equals(day2);  }    }  public static class Reflector  {  public static void GetConstructors(Type Class)  {  ConstructorInfo[] ctor = Class.GetConstructors();  for (int i = 0; i < ctor.Length; i++)  {  Console.WriteLine(ctor[i]);  }  }  public static void GetProperties(Type Class)  {  PropertyInfo[] properties = Class.GetProperties();  for (int i = 0; i < properties.Length; i++)  {  Console.WriteLine(properties[i]);  }  }  }  public class TimeTable<T> where T: AnyDayTimeTable  {  public List<T> list = new List<T>();    public void GetDay()  {  var Selected = (from selected in this.list  group selected by selected.Day  into g  orderby g.Count() descending  select g).Take(1);  foreach (var item in Selected)  {  Console.WriteLine(item.Key);  }  }  }  public class Singleton  {  private static readonly Singleton instance = new Singleton();  public static Singleton Instance  {  get { return instance; }  }  protected Singleton() { }  }  } |

|  |
| --- |
| ----------------------------------------------------------сериализация--------------------------------------  using System.Runtime.Serialization.Formatters.Soap;  using System.Runtime.Serialization.Json;  using System.Xml;  using System.Xml.Linq;  using System.Xml.Serialization;  using System.IO;  using System;  using System.Collections.Generic;  using System.Linq;  namespace Laba\_13  {  public class Programm  {  public static void Main()  {  //-------------------------------1 задание -------------------------------------//  //Goods cakeBinary = new Cake(11230, "BSTU", "XZ", "OOO Dostavka.by", "Imperatriza");  ООП\_лаб\_4\_5.Rose roseBinary = new ООП\_лаб\_4\_5.Rose("rose");  BinaryFormatter binaryFormatter = new BinaryFormatter();  using (FileStream fs = new FileStream("BinaryFormat.dat", FileMode.OpenOrCreate))  {  binaryFormatter.Serialize(fs, roseBinary);  };  using (FileStream fs = new FileStream("BinaryFormat.dat", FileMode.OpenOrCreate))  {  ООП\_лаб\_4\_5.Rose roseBinaryDes = (ООП\_лаб\_4\_5.Rose)binaryFormatter.Deserialize(fs);  Console.WriteLine("=====Дисериализация BinaryFormatter=====");  Console.WriteLine(roseBinaryDes.ToString());  Console.WriteLine("========================");  };  ООП\_лаб\_4\_5.Rose roseSoap = new ООП\_лаб\_4\_5.Rose("roze");  SoapFormatter soapFormatter = new SoapFormatter();  using (FileStream fl = new FileStream("SoapFormat.xml", FileMode.OpenOrCreate))  {  soapFormatter.Serialize(fl, roseSoap);  };  using (FileStream fl = new FileStream("SoapFormat.xml", FileMode.OpenOrCreate))  {  Console.WriteLine("=====Дисериализация SoapFormatter=====");  ООП\_лаб\_4\_5.Rose roseSoapFormat = (ООП\_лаб\_4\_5.Rose)soapFormatter.Deserialize(fl);  Console.WriteLine(roseSoapFormat.ToString());  Console.WriteLine("========================");  }  ООП\_лаб\_4\_5.Rose roseJSON = new ООП\_лаб\_4\_5.Rose("rose");  DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(ООП\_лаб\_4\_5.Rose));  using (FileStream fs = new FileStream("JsonFormat.json", FileMode.OpenOrCreate))  {  jsonSerializer.WriteObject(fs, roseJSON);  };  using (FileStream fs = new FileStream("JsonFormat.json", FileMode.OpenOrCreate))  {  Console.WriteLine("=====Дисериализация DataContractJsonSerializer=====");  ООП\_лаб\_4\_5.Rose roseJSONform = (ООП\_лаб\_4\_5.Rose)jsonSerializer.ReadObject(fs);  Console.WriteLine(roseJSONform.ToString());  Console.WriteLine("========================");  };  ООП\_лаб\_4\_5.Rose roseXML = new ООП\_лаб\_4\_5.Rose("rose");  XmlSerializer xmlSerializer = new XmlSerializer(typeof(ООП\_лаб\_4\_5.Rose));  using (FileStream fs = new FileStream("XmlFormat.xml", FileMode.OpenOrCreate))  {  xmlSerializer.Serialize(fs, roseXML);  };  using (FileStream fs = new FileStream("XmlFormat.xml", FileMode.OpenOrCreate))  {  Console.WriteLine("=====Дисериализация XmlSerializer=====");  ООП\_лаб\_4\_5.Rose xmlrose = (ООП\_лаб\_4\_5.Rose)xmlSerializer.Deserialize(fs);  Console.WriteLine(xmlrose.ToString());  Console.WriteLine("========================");  };  //---------------------- 2 задание -------------------------//  ООП\_лаб\_4\_5.Rose[] arrRose = new ООП\_лаб\_4\_5.Rose[] { roseBinary, roseSoap, roseXML };  XmlSerializer xmlSerializerArr = new XmlSerializer(typeof(ООП\_лаб\_4\_5.Rose[]));  using (FileStream fs = new FileStream("arrCakeXml.xml", FileMode.OpenOrCreate))  {  xmlSerializerArr.Serialize(fs, arrRose);  }  using (FileStream fs = new FileStream("arrCakeXml.xml", FileMode.OpenOrCreate))  {  Console.WriteLine("=====Дисериализация массива Cake=====");  ООП\_лаб\_4\_5.Rose[]? newRose = xmlSerializerArr.Deserialize(fs) as ООП\_лаб\_4\_5.Rose[];  if (newRose != null)  {  foreach (ООП\_лаб\_4\_5.Rose good in newRose)  {  Console.WriteLine(good.ToString());  }  }  Console.WriteLine("========================");  };  //---------------------- 3 задание -------------------------//  XmlDocument xDoc = new XmlDocument();  xDoc.Load("arrCakeXml.xml");  XmlElement? xRoot = xDoc.DocumentElement;  XmlNodeList? nodes = xRoot?.SelectNodes("Cake");  if (nodes != null)  {  foreach (XmlNode nod in nodes)  Console.WriteLine(nod.OuterXml);  }  XmlNodeList? node = xRoot?.SelectNodes("\*");  if (nodes != null)  {  foreach (XmlNode nod in node)  Console.WriteLine(nod.OuterXml);  }  //---------------------- 4 задание -------------------------//  Console.WriteLine("=====================Задание 4======================");  XDocument xDoc2 = new XDocument();  XElement root = new XElement("Rose");  XElement Rose;  XElement color;  var list = new List<ООП\_лаб\_4\_5.Rose>();  list.Add(roseXML);  list.Add(roseJSON);  list.Add(roseSoap);  foreach (var item in list)  {  if (item is ООП\_лаб\_4\_5.Rose)  {  Rose = new XElement("Rose");  color = new XElement("red");  color.Value = item.color;  Rose.Add(color);  root.Add(Rose);  }  else  {  Console.WriteLine("Лох");  }  }  xDoc2.Add(root);  var elements = from p in root.Elements("Cake").Where(p => Convert.ToInt32(p.Attribute("Cost").Value) < 10000)  select p;  foreach (var item in elements)  {  Console.WriteLine(item.Value);  }  }  } |

|  |
| --- |
| ----------------------------------------------------------рефлектор--------------------------------------  static class Reflector  {  public static string GetAssemblyName(Type Class)  {  string AssemblyInfo ="Имя сборки: " + Class.AssemblyQualifiedName;  return AssemblyInfo;  }  public static IEnumerable<string> PublicConstuctors(Type Class)  {  ConstructorInfo[] PublicConstructorInfo = Class.GetConstructors();  List<string> str = new List<string>() { Convert.ToString("Всего открытых конструкторов: " + PublicConstructorInfo.Length) };  for (int i = 0; i < PublicConstructorInfo.Length ; i++)  {  str.Add(Convert.ToString(PublicConstructorInfo[i]));  }  return str;  }  public static IEnumerable<string> GetSetFieldInfo(Type Class)  {  FieldInfo[] fieldInfo = Class.GetFields(BindingFlags.NonPublic | BindingFlags.Instance | BindingFlags.Public);  PropertyInfo[] propertyInfo = Class.GetProperties(BindingFlags.NonPublic | BindingFlags.Instance | BindingFlags.Public);  List<string> str = new List<string>();  for (int i = 0; i < fieldInfo.Length + propertyInfo.Length; i++)  {  if(i < fieldInfo.Length)  {  if (i == 0) str.Add("Поля класса:");  str.Add(Convert.ToString(fieldInfo[i]));  }  else  {  if (i == fieldInfo.Length) str.Add("Свойства класса:");  str.Add(Convert.ToString(propertyInfo[i - fieldInfo.Length]));  }  }  return str;  }  public static IEnumerable<string> InterfaceInfo(Type Class)  {  List<string> interfaces = new List<string>() { "Реализованные интерфейсы:" };  foreach (var item in Class.GetInterfaces())  {  interfaces.Add(Convert.ToString(item));  }  return interfaces;  }  public static IEnumerable<string> AllMethods(Type Class)  {  MethodInfo[] methodInfos = Class.GetMethods(BindingFlags.NonPublic | BindingFlags.Instance | BindingFlags.Public);  var Paremeter = (int)2;  Type ParameterType = Paremeter.GetType();  List<string> Methods = new List<string>() { $"Все методы с типом параметра {ParameterType.Name}:" };  var MethodInfos = from selected in methodInfos  where !selected.Name.StartsWith("get\_") && !selected.Name.StartsWith("set\_")  select selected;  foreach (var Method in MethodInfos)  {    foreach (var parameter in Method.GetParameters())  {  if(Convert.ToString(parameter.ParameterType.Name) == Convert.ToString(ParameterType.Name))  {  Methods.Add(Convert.ToString(Method));  }  }  }  return Methods;  }  public static object? Invoke(object? obj, string MethodName, object[] Parameters = null)  {  var Method = obj.GetType().GetMethod(MethodName);  return Method?.Invoke(obj, Parameters);  }  public static object? Create(Type type, object[] Parameters)  {  ConstructorInfo[] constructors= type.GetConstructors();  foreach (var constructor in constructors)  {  try  {  return constructor.Invoke(Parameters);  }  catch (Exception) { }  }  return null;  }  } |

|  |
| --- |
| delegate void Help();  class Doctor  {  public event Help help= null;  public void HELP(Person person)  {  if (person.GetTemp() == true)  {  if (help != null)  help.Invoke();  Console.WriteLine("Температура снята");  }  else  {  Console.WriteLine("температуры не было");  }  }  }  class Person  {  bool temperature;  public Person(bool Temp)  {  temperature = Temp;  }  public void GiveTemp()  {  temperature = true;  }  public bool GetTemp()  {  return temperature;  }  public void help()  {  temperature = false;  }  } |

----------------------------------------------------------событие--------------------------------------

|  |
| --- |
| --------------------------коллекция  public interface IAddRemove<T>  {  public void Add(T item);  public void Remove(T item);  public void Browse(int index);  }  public partial class Set<T> : IEnumerable, IAddRemove<T> where T: notnull  {  private List<T> \_items = new List<T>();  public int Count => \_items.Count;//  public void Add(T item)  {  if (item == null)  {  throw new ArgumentNullException(nameof(item));  }  if (!\_items.Contains(item))  {  \_items.Add(item);  }  }  public void Remove(T item)  {  if (item == null)  {  throw new ArgumentNullException(nameof(item));  }  if (!\_items.Contains(item))  {  throw new KeyNotFoundException($"Элемент {item} не найден в множестве.");  }  \_items.Remove(item);  }  public void Browse(int index)  {  if (index > Count)  {  throw new System.NullReferenceException("Нет элемента с таким индексом");  }  Console.WriteLine($"Элемент с индексом {index}: {\_items[index]}");  }  public IEnumerator GetEnumerator()  {  return \_items.GetEnumerator();  } |