Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Отчёт по лабораторной работе №5

Выполнил:

студент 3 курса

Группы ПО-5

Поздняков Д.А.

Проверил:

Крощенко А.А.

Брест 2021

**Цель работы:** приобрести практические навыки в области объектно-ориентированного проектирования

**Вариант 5**

**Задание 1.**

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Здание ← abstract class Общественное Здание ← class Театр.

**Задание 2.**

В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Создать абстрактный класс Работник фирмы и подклассы Менеджер, Аналитик, Программист, Тестировщик, Дизайнер, Бухгалтер. Реализовать логику начисления зарплаты.

**Задание 3.**

В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

**Код программы (задание 1):**

Program.cs

using System;

namespace lab5.\_1.\_5

{

class Program

{

static void Main(string[] args)

{

Theatre v\_Theater = new Theatre(new string("Opera de Paris"), new string("Pl. de I'Opera, 75009"), 2000, 14.00, new string("10:00 - 18:00"));

Console.WriteLine(new string("Name: ") + v\_Theater.GetName());

Console.WriteLine(new string("Address: ") + v\_Theater.GetAddress());

Console.WriteLine(new string("Visitors count: ") + v\_Theater.GetVisitorsCount());

Console.WriteLine(new string("Visit price: ") + v\_Theater.GetVisitPrice());

Console.WriteLine(new string("Work time: ") + v\_Theater.GetWorkTime());

Console.WriteLine();

v\_Theater.ShowPerformance();

}

}

}

Bilding.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_1.\_5

{

public interface Building

{

public void Create();

public void Destroy();

public void Open();

public void Close();

}

}

PublicBilding.cs

namespace lab5.\_1.\_5

{

public abstract class PublicBuilding : TownBuilding

{

private int VisitorsCount = 0;

private double VisitPrice = 0.0;

private string WorkTime = new string("00:00 - 00:00");

public PublicBuilding(string name, string address) : base (name, address)

{

}

public PublicBuilding(string name, string address, int visitorsCount) : base ( name, address)

{

VisitorsCount = visitorsCount;

}

public PublicBuilding(string name, string address, int visitorsCount, double visitPrice) : base( name, address)

{

VisitorsCount = visitorsCount;

VisitPrice = visitPrice;

}

public PublicBuilding(string name, string address, int visitorsCount, double visitPrice, string workTime) : base( name, address)

{

VisitorsCount = visitorsCount;

VisitPrice = visitPrice;

WorkTime = workTime;

}

public string GetWorkTime()

{

return WorkTime;

}

public void SetWorkTime(string workTime)

{

WorkTime = workTime;

}

public int GetVisitorsCount()

{

return VisitorsCount;

}

public void SetVisitorsCount(int visitorsCount)

{

VisitorsCount = visitorsCount;

}

public double GetVisitPrice()

{

return VisitPrice;

}

public void SetVisitPrice(double visitPrice)

{

VisitPrice = visitPrice;

}

}

}

Theatre.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_1.\_5

{

public class Theatre : PublicBuilding

{

public Theatre(string name, string address) : base (name, address)

{

}

public Theatre(string name, string address, int visitorsCount) : base(name, address, visitorsCount)

{

}

public Theatre(string name, string address, int visitorsCount, double visitPrice) : base(name, address, visitorsCount, visitPrice)

{

}

public Theatre(string name, string address, int visitorsCount, double visitPrice, string c\_WorkTime) : base(name, address, visitorsCount, visitPrice, c\_WorkTime)

{

}

public void ShowPerformance()

{

Console.WriteLine("Very interesting performance...");

}

}

}

TownBilding.cs

using System;

namespace lab5.\_1.\_5

{

public abstract class TownBuilding : Building

{

private bool isCreated = false, isOpen = false;

private string Name, Address;

public TownBuilding(string name, string address)

{

Name = name;

Address = address;

}

public void SetName(string name)

{

Name = name;

}

public string GetName()

{

return Name;

}

public void SetAdress(string address)

{

Address = address;

}

public string GetAddress()

{

return Address;

}

public void Create()

{

isCreated = true;

}

public void Destroy()

{

isCreated = false;

}

public bool IsCreated()

{

return isCreated;

}

public void Open()

{

isOpen = true;

}

public void Close()

{

isOpen = false;

}

public bool IsOpen()

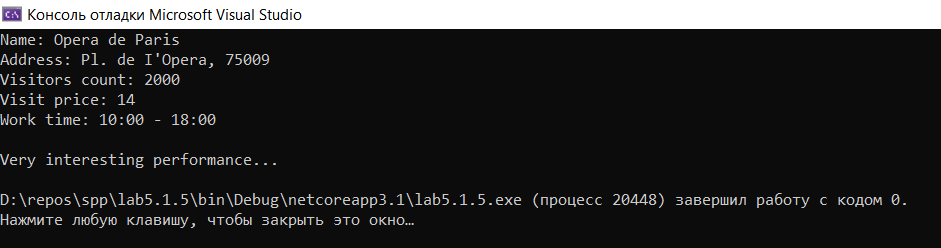
{

return isOpen;

}

}

}



**Код программы (задание 2):**

Program.cs

using System;

using System.Collections.Generic;

namespace lab5.\_2.\_5

{

class Program

{

public enum Speciality

{

MANAGER,

ANALYST,

PROGRAMMER,

TESTER,

DESIGNER,

ACCOUNTANT

}

public static int RandomInt(int Min, int Max)

{

Random rand = new Random();

return rand.Next(Min, Max);

}

public static double RandomDouble(double Min, double Max)

{

Random random = new Random();

return Math.Round(random.NextDouble() \* (Max - Min) + Min, 2);

}

static void Main(string[] args)

{

List<string> Names = new List<string>();

Names.Add(new string("Maya Rogerson"));

Names.Add(new string("Liana Roy"));

Names.Add(new string("Liza Thomson"));

Names.Add(new string("Phoebe Teel"));

Names.Add(new string("Erika Herbertson"));

Names.Add(new string("Vicky Nixon"));

Names.Add(new string("Donelle Joiner"));

Names.Add(new string("Brynlee Alden"));

Names.Add(new string("Mae Isaacson"));

Names.Add(new string("Ocean Sharp"));

Names.Add(new string("Phil Stephenson"));

Names.Add(new string("Tilly Ellington"));

Names.Add(new string("Richmal Statham"));

Names.Add(new string("Margo Gadsby"));

Names.Add(new string("Trudie George"));

Names.Add(new string("Kortney Abbott"));

Names.Add(new string("Amery Lyon"));

Names.Add(new string("Brooklyn Knight"));

int NumberOfWorkers = Names.Count, MinAge = 18, MaxAge = 65;

double MinimalSalary = 100.0, MinSalaryK = 1.0, MaxSalaryK = 2.0, ManagerMaxMinSalary = 1000.0,

AnalystMaxMinSalary = 800.0, ProgrammerMaxMinSalary = 900.0, TesterMaxMinSalary = 500.0,

DesignerMaxMinSalary = 700.0, AccountantMaxMinSalary = 450.0;

List<Employee> Employees = new List<Employee>();

Random rand = new Random();

Type type = typeof(Speciality);

Array values = type.GetEnumValues();

for (int i = 0; i < NumberOfWorkers; ++i)

{

int tempIndex = rand.Next(values.Length);

switch ((Speciality)values.GetValue(tempIndex))

{

case Speciality.MANAGER:

Employees.Add(new Manager(RandomInt(MinAge, MaxAge), Names[i],

RandomDouble(MinimalSalary, ManagerMaxMinSalary),

RandomDouble(MinSalaryK, MaxSalaryK)));

break;

case Speciality.ANALYST:

Employees.Add(new Analyst(RandomInt(MinAge, MaxAge), Names[i],

RandomDouble(MinimalSalary, AnalystMaxMinSalary),

RandomDouble(MinSalaryK, MaxSalaryK)));

break;

case Speciality.PROGRAMMER:

Employees.Add(new Programmer(RandomInt(MinAge, MaxAge), Names[i],

RandomDouble(MinimalSalary, ProgrammerMaxMinSalary),

RandomDouble(MinSalaryK, MaxSalaryK)));

break;

case Speciality.TESTER:

Employees.Add(new Tester(RandomInt(MinAge, MaxAge), Names[i],

RandomDouble(MinimalSalary, TesterMaxMinSalary),

RandomDouble(MinSalaryK, MaxSalaryK)));

break;

case Speciality.DESIGNER:

Employees.Add(new Designer(RandomInt(MinAge, MaxAge), Names[i],

RandomDouble(MinimalSalary, DesignerMaxMinSalary),

RandomDouble(MinSalaryK, MaxSalaryK)));

break;

case Speciality.ACCOUNTANT:

Employees.Add(new Accountant(RandomInt(MinAge, MaxAge), Names[i],

RandomDouble(MinimalSalary, AccountantMaxMinSalary),

RandomDouble(MinSalaryK, MaxSalaryK)));

break;

}

Employees[i].StartWorking();

}

for (int i = 0; i < NumberOfWorkers; ++i)

{

Employee Employee = Employees[i];

Employee.Work();

Console.WriteLine("Salary = " + Employee.GetMinimalSalary() + " \* " + Employee.GetSalary()

+ " = " + Employee.Get\_Salary() + '\n');

}

}

}

}

Accountant.cs

using System;

namespace lab5.\_2.\_5

{

public class Accountant : Employee

{

public Accountant(int Age, string Name, double MinimalSalary) : base(Age, Name, MinimalSalary)

{

}

public Accountant(int Age, string Name, double MinimalSalary, double Salary) : base(Age, Name, MinimalSalary, Salary)

{

}

public override void Work()

{

if (IsWorking())

{

Console.WriteLine("The accountant " + GetName() + " is working...");

return;

}

Console.WriteLine("Accountant " + GetName() + " does not work...");

}

}

}

Analyst.cs

using System;

namespace lab5.\_2.\_5

{

public class Analyst : Employee

{

public Analyst(int Age, string Name, double MinimalSalary) : base(Age, Name, MinimalSalary)

{

}

public Analyst(int Age, string Name, double MinimalSalary, double Salary) : base(Age, Name, MinimalSalary, Salary)

{

}

public override void Work()

{

if (IsWorking())

{

Console.WriteLine("The analyst " + GetName() + " is working...");

return;

}

Console.WriteLine("Analyst " + GetName() + " does not work...");

}

}

}

Designer.cs

using System;

namespace lab5.\_2.\_5

{

public class Designer : Employee

{

public Designer(int Age, string Name, double MinimalSalary) : base(Age, Name, MinimalSalary)

{

}

public Designer(int Age, string Name, double MinimalSalary, double Salary) : base(Age, Name, MinimalSalary, Salary)

{

}

public override void Work()

{

if (IsWorking())

{

Console.WriteLine("The designer " + GetName() + " is working...");

return;

}

Console.WriteLine("Designer " + GetName() + " does not work...");

}

}

}

Employee.cs

namespace lab5.\_2.\_5

{

public abstract class Employee : Person

{

private bool isWorking = false;

private double minimalSalary = 0.0, salary = 1.0, \_salary = 0.0;

public Employee(int Age, string Name, double MinimalSalary) : base(Age, Name)

{

minimalSalary = MinimalSalary;

\_salary = minimalSalary \* salary;

}

public Employee(int Age, string Name, double MinimalSalary, double Salary) : base(Age, Name)

{

minimalSalary = MinimalSalary;

salary = Salary;

\_salary = minimalSalary \* salary;

}

public void StartWorking()

{

isWorking = true;

}

public void StopWorking()

{

isWorking = false;

}

public bool IsWorking()

{

return isWorking;

}

public void SetMinimalSalary(double MinimalSalary)

{

minimalSalary = MinimalSalary;

\_salary = minimalSalary \* salary;

}

public double GetMinimalSalary()

{

return minimalSalary;

}

public void SetSalary(double Salary)

{

salary = Salary;

\_salary = minimalSalary \* salary;

}

public double GetSalary()

{

return salary;

}

public double Get\_Salary()

{

return \_salary;

}

public abstract void Work();

}

}

Manager.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_2.\_5

{

public class Manager : Employee

{

public Manager(int Age, string Name, double MinimalSalary) : base(Age, Name, MinimalSalary)

{

}

public Manager(int Age, string Name, double MinimalSalary, double SalaryK) : base(Age, Name, MinimalSalary, SalaryK)

{

}

public override void Work()

{

if (this.IsWorking())

{

Console.WriteLine("The manager " + this.GetName() + " is working...");

return;

}

Console.WriteLine("Manager " + this.GetName() + " does not work...");

}

}

}

Person.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_2.\_5

{

public abstract class Person

{

private int Age = 0;

private string Name;

public Person(int age, string name)

{

this.Age = age;

this.Name = name;

}

public void SetAge(int age)

{

this.Age = age;

}

public int GetAge()

{

return this.Age;

}

public void SetName(string name)

{

this.Name = name;

}

public string GetName()

{

return this.Name;

}

}

}

Programmer.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_2.\_5

{

public class Programmer : Employee

{

public Programmer(int Age, string Name, double MinimalSalary) : base(Age, Name, MinimalSalary)

{

}

public Programmer(int Age, string Name, double MinimalSalary, double SalaryK) : base(Age, Name, MinimalSalary, SalaryK)

{

}

public override void Work()

{

if (this.IsWorking())

{

Console.WriteLine("The programmer " + this.GetName() + " is working...");

return;

}

Console.WriteLine("Programmer " + this.GetName() + " does not work...");

}

}

}

Tester.cs

using System;

namespace lab5.\_2.\_5

{

public class Tester : Employee

{

public Tester(int Age, string Name, double MinimalSalary) : base(Age, Name, MinimalSalary)

{

}

public Tester(int Age, string Name, double MinimalSalary, double SalaryK) : base(Age, Name, MinimalSalary, SalaryK)

{

}

public override void Work()

{

if (this.IsWorking())

{

Console.WriteLine("The tester " + this.GetName() + " is working...");

return;

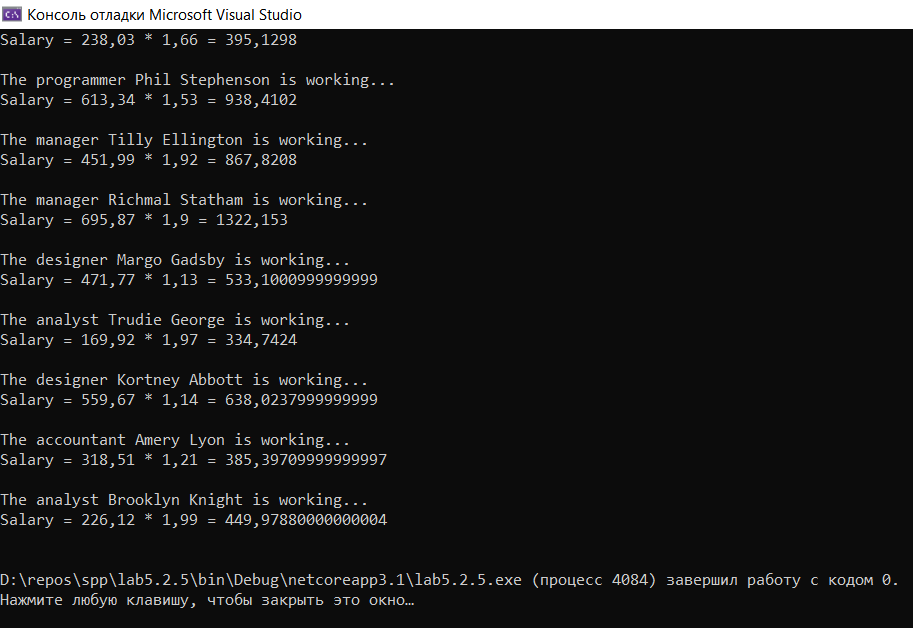
}

Console.WriteLine("Tester " + this.GetName() + " does not work...");

}

}

}



**Код программы (задание 3):**

Program.cs

using System;

namespace lab5.\_3.\_5

{

class Program

{

static void Main(string[] args)

{

Catalog Catalog = new Catalog();

Catalog.AddBook(new Book(2008, "Fahrenheit 451", "Ray Bradbury"));

Catalog.AddBook(new Book(1949, "1984", "George Orwell"));

Catalog.AddBook(new Book(2006, "The Master and Margarita", "Michael Bulgakov"));

Catalog.AddBook(new Book(2003, "Shantaram", "Gregory David Roberts"));

Catalog.AddBook(new Book(2002, "The Lovely Bones", "Alice Sebold"));

Library Library = new Library(new string("Central Library"), new string("blvd. Shevchenko 3"), 400, 10, new string("08:00 - 20:00"));

Library.SetCatalog(Catalog);

Librarian Librarian = new Librarian(65, "Mayson Falconer", Library);

Administrator Administrator = new Administrator(31, "Janette Trueman", Library);

Reader Reader = new Reader(16, "Conner Thacker");

Library.AddLibrarian(Librarian);

Library.AddAdministrator(Administrator);

Library.AddReader(Reader);

Librarian.Work();

Administrator.Work();

Console.WriteLine(new string("Name: ") + Library.GetName());

Console.WriteLine(new string("Address: ") + Library.GetAddress());

Console.WriteLine(new string("Visitors count: ") + Library.GetVisitorsCount());

Console.WriteLine(new string("Visit price: ") + Library.GetVisitPrice());

Console.WriteLine(new string("Work time: ") + Library.GetWorkTime());

Console.WriteLine();

if (Reader.RequestBook(Library, new Book(2006, "The Master and Margarita", "Michael Bulgakov")))

{

Console.WriteLine("Book successfully taken");

}

else

{

Console.WriteLine("Can not take a book");

}

Library.Update();

}

}

}

Administrator.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public class Administrator : WorkingPerson

{

private Library Library = null;

public Administrator(int Age, string Name, Library Library) : base(Age, Name)

{

this.Library = Library;

}

public void SetLibrary(Library Library)

{

this.Library = Library;

}

public Library GetLibrary()

{

return Library;

}

public void AddToBlackList(Reader Reader)

{

Library.AddToBlackList(Reader);

}

public void Update()

{

foreach (Order Order in Library.GetOrders())

{

if (!Library.CheckDeadline(Order))

{

AddToBlackList(Order.GetReader());

}

}

}

}

}

Book.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public class Book

{

private int Year = 0;

private string Title, Author;

public Book(int year, string title, string author)

{

Year = year;

Title = title;

Author = author;

}

public void SetYear(int year)

{

Year = year;

}

public int GetYear()

{

return Year;

}

public void SetTitle(string title)

{

Title = title;

}

public string GetTitle()

{

return Title;

}

public void SetAuthor(string author)

{

Author = author;

}

public string GetAuthor()

{

return Author;

}

public bool equals(object Other)

{

if (GetYear().Equals(((Book)Other).GetYear())

&& GetTitle().Equals(((Book)Other).GetTitle())

&& GetAuthor().Equals(((Book)Other).GetAuthor()))

{

return true;

}

return false;

}

}

}

Building.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public interface Building

{

public void create();

public void destroy();

public void open();

public void close();

}

}

Catalog.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public class Catalog

{

private List<Book> Books = new List<Book>();

public Catalog()

{

}

public Catalog(List<Book> books)

{

Books = books;

}

public void SetBooks(List<Book> books)

{

Books = books;

}

public List<Book> GetBooks() {

return Books;

}

public void AddBook(Book book)

{

Books.Add(book);

}

public void RemoveBook(Book book)

{

Books.Remove(book);

}

public bool BookSearch(Book book)

{

for(int i = 0; i < Books.Count; i++)

{

if (Books[i].GetAuthor() == book.GetAuthor() && Books[i].GetTitle() == book.GetTitle() && Books[i].GetYear() == book.GetYear())

{

return true;

}

}

return false;

}

public Book GiveBook(Book book)

{

int Index = -1;

for (int i = 0; i < Books.Count; i++)

{

if (Books[i].GetAuthor() == book.GetAuthor() && Books[i].GetTitle() == book.GetTitle() && Books[i].GetYear() == book.GetYear())

{

Index = i;

break;

}

}

if (Index == -1)

{

return null;

}

Book Result = Books[Index];

RemoveBook(book);

return Result;

}

}

}

Librarian.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public class Librarian : WorkingPerson

{

private Library Library = null;

public Librarian(int Age, string Name, Library Library) : base(Age, Name)

{

this.Library = Library;

}

public void SetLibrary(Library Library)

{

this.Library = Library;

}

public Library GetLibrary()

{

return Library;

}

public void AddOrder(Order Order)

{

Library.AddOrder(Order);

}

}

}

Library.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public class Library : PublicBuilding

{

private Catalog Catalog = new Catalog();

private List<Administrator> Administrators = new List<Administrator>();

private List<Librarian> Librarians = new List<Librarian>();

private List<Reader> Readers = new List<Reader>(), BlackList = new List<Reader>();

private List<Order> Orders = new List<Order>();

private string Deadline = new string("11/12/2021");

public Library(string Name, string Address) : base(Name, Address)

{

}

public Library(string Name, string Address, int VisitorsCount) : base(Name, Address, VisitorsCount)

{

}

public Library(string Name, string Address, int VisitorsCount, double VisitPrice) : base(Name, Address, VisitorsCount, VisitPrice)

{

}

public Library(string Name, string Address, int VisitorsCount, double VisitPrice, string WorkTime) : base(Name, Address, VisitorsCount, VisitPrice, WorkTime)

{

}

public Library(string Name, string Address, Catalog catalog) : base(Name, Address)

{

Catalog = catalog;

}

public Library(string Name, string Address, Catalog catalog, List<Administrator> administrators) : base(Name, Address)

{

Catalog = catalog;

Administrators = administrators;

}

public Library(string Name, string Address, Catalog catalog, List<Administrator> administrators, List<Librarian> librarians) : base(Name, Address)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

}

public Library(string Name, string Address, Catalog catalog, List<Administrator> administrators, List<Librarian> librarians, List<Reader> readers) : base(Name, Address)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

Readers = readers;

}

public Library(string Name, string Address, Catalog catalog, List<Administrator> administrators, List<Librarian> librarians, List<Reader> readers, List<Reader> blackList) : base(Name, Address)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

Readers = readers;

BlackList = blackList;

}

public Library(string Name, string Address, Catalog catalog, List<Administrator> administrators, List<Librarian> librarians, List<Reader> readers, List<Reader> blackList, List<Order> orders) : base(Name, Address)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

Readers = readers;

BlackList = blackList;

Orders = orders;

}

public void SetCatalog(Catalog catalog)

{

Catalog = catalog;

}

public Catalog GetCatalog()

{

return Catalog;

}

public void SetAdministrators(List<Administrator> administrators)

{

Administrators = administrators;

}

public List<Administrator> GetAdministrators() {

return Administrators;

}

public void SetLibrarians(List<Librarian> librarians)

{

Librarians = librarians;

}

public List<Librarian> GetLibrarians() {

return Librarians;

}

public void SetReaders(List<Reader> readers)

{

Readers = readers;

}

public List<Reader> GetReaders() {

return Readers;

}

public void SetBlackList(List<Reader> blackList)

{

BlackList = blackList;

}

public List<Reader> GetBlackList() {

return BlackList;

}

public void SetOrders(List<Order> orders)

{

Orders = orders;

}

public List<Order> GetOrders() {

return Orders;

}

public void AddBook(Book book)

{

Catalog.AddBook(book);

}

public void RemoveBook(Book book)

{

Catalog.RemoveBook(book);

}

public void AddAdministrator(Administrator administrator)

{

Administrators.Add(administrator);

}

public void RemoveAdministrator(Administrator administrator)

{

Administrators.Remove(administrator);

}

public void AddLibrarian(Librarian librarian)

{

Librarians.Add(librarian);

}

public void RemoveLibrarian(Librarian librarian)

{

Librarians.Remove(librarian);

}

public void AddReader(Reader reader)

{

Readers.Add(reader);

}

public void RemoveReader(Reader reader)

{

Readers.Remove(reader);

}

public void AddToBlackList(Reader reader)

{

BlackList.Add(reader);

}

public void RemoveFromBlackList(Reader reader)

{

BlackList.Remove(reader);

}

public void AddOrder(Order order)

{

Orders.Add(order);

}

public void RemoveOrder(Order order)

{

Orders.Remove(order);

}

public bool BookSearch(Book book)

{

return Catalog.BookSearch(book);

}

public string GenerateDeadline()

{

return Deadline;

}

public bool CheckDeadline(Order order)

{

return Deadline.Equals(order.GetDeadline());

}

public Order ProcessOrder(Reader reader, Book book)

{

Book OrderedBook = Catalog.GiveBook(book);

if (OrderedBook == null)

{

return null;

}

string Deadline = this.GenerateDeadline();

Order order = new Order(OrderedBook, reader, Deadline);

bool Processed = false;

while (!Processed)

{

foreach (Librarian librarian in Librarians)

{

if (librarian.IsWorking())

{

librarian.AddOrder(order);

Processed = true;

break;

}

}

}

Orders.Add(order);

return order;

}

public void Update()

{

bool Updated = false;

while (!Updated)

{

foreach (Administrator administrator in Administrators)

{

if (administrator.IsWorking())

{

administrator.Update();

Updated = true;

break;

}

}

}

}

}

}

Order.cs

namespace lab5.\_3.\_5

{

public class Order

{

private Book Book = null;

private Reader Reader = null;

private string Deadline;

public Order(Book book, Reader reader, string deadline)

{

Book = book;

Reader = reader;

Deadline = deadline;

}

public void SetBook(Book book)

{

Book = book;

}

public Book GetBook()

{

return Book;

}

public void SetReader(Reader reader)

{

Reader = reader;

}

public Reader GetReader()

{

return Reader;

}

public void SetDeadline(string deadline)

{

Deadline = deadline;

}

public string GetDeadline()

{

return Deadline;

}

}

}

Person.cs

namespace lab5.\_3.\_5

{

public abstract class Person

{

private int Age = 0;

private string Name;

public Person(int age, string name)

{

Age = age;

Name = name;

}

public void SetAge(int age)

{

Age = age;

}

public int GetAge()

{

return Age;

}

public void SetName(string name)

{

Name = name;

}

public string GetName()

{

return Name;

}

}

}

PublicBilding.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public abstract class PublicBuilding : TownBuilding

{

private int VisitorsCount = 0;

private double VisitPrice = 0.0;

private string WorkTime = new string("00:00 - 00:00");

public PublicBuilding(string name, string address) : base(name, address)

{

}

public PublicBuilding(string name, string address, int visitorsCount) : base(name, address)

{

VisitorsCount = visitorsCount;

}

public PublicBuilding(string name, string address, int visitorsCount, double visitPrice) : base(name, address)

{

VisitorsCount = visitorsCount;

VisitPrice = visitPrice;

}

public PublicBuilding(string name, string address, int visitorsCount, double visitPrice, string workTime) : base(name, address)

{

VisitorsCount = visitorsCount;

VisitPrice = visitPrice;

WorkTime = workTime;

}

public string GetWorkTime()

{

return WorkTime;

}

public void SetWorkTime(string workTime)

{

WorkTime = workTime;

}

public int GetVisitorsCount()

{

return VisitorsCount;

}

public void SetVisitorsCount(int visitorsCount)

{

VisitorsCount = visitorsCount;

}

public double GetVisitPrice()

{

return VisitPrice;

}

public void SetVisitPrice(double visitPrice)

{

VisitPrice = visitPrice;

}

}

}

Reader.cs

using System.Collections.Generic;

namespace lab5.\_3.\_5

{

public class Reader : Person

{

private List<Order> Orders = new List<Order>();

public Reader(int Age, string Name) : base(Age, Name)

{

}

public Reader(int Age, string Name, List<Order> Order) : base(Age, Name)

{

Orders = Order;

}

public void SetOrders(List<Order> Order)

{

Orders = Order;

}

public List<Order> GetOrders()

{

return Orders;

}

public void AddOrder(Order order)

{

Orders.Add(order);

}

public void RemoveOrder(Order order)

{

Orders.Remove(order);

}

public bool RequestBook(Library Library, Book Book)

{

if (Library.BookSearch(Book))

{

Order order = Library.ProcessOrder(this, Book);

if (order == null)

{

return false;

}

AddOrder(order);

return true;

}

return false;

}

}

}

TownBilding.cs

using System;

namespace lab5.\_3.\_5

{

public abstract class TownBuilding : Building

{

private bool IsCreated = false, IsOpen = false;

private string Name, Address;

public TownBuilding(string name, string address)

{

Name = name;

Address = address;

}

public void SetName(string name)

{

Name = name;

}

public string GetName()

{

return Name;

}

public void SetAddress(string address)

{

Address = address;

}

public string GetAddress()

{

return Address;

}

public void create()

{

IsCreated = true;

}

public void destroy()

{

IsCreated = false;

}

public bool isCreated()

{

return IsCreated;

}

public void open()

{

IsOpen = true;

}

public void close()

{

IsOpen = false;

}

public bool isOpen()

{

return IsOpen;

}

}

}

Worker.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public interface Worker

{

public void Work();

public bool IsWorking();

public void StopWorking();

}

}

WorkingPerson.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5.\_3.\_5

{

public abstract class WorkingPerson : Person, Worker

{

private bool isWorking = false;

public WorkingPerson(int Age, string Name) : base(Age, Name)

{

}

public void Work()

{

isWorking = true;

}

public bool IsWorking()

{

return isWorking;

}

public void StopWorking()

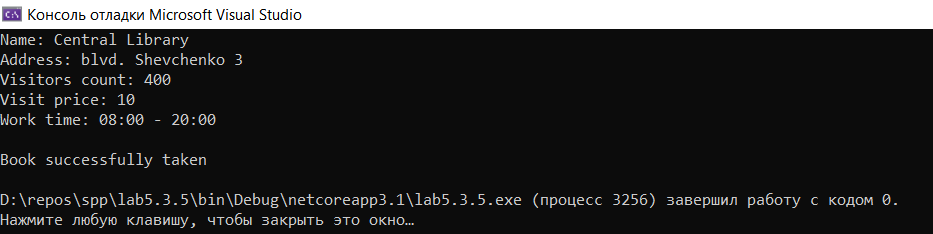
{

isWorking = false;

}

}

}



Вывод: приобрел практические навыки в области объектно-ориентированного проектирования.