Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «CПП»

Выполнил:

Студент 3 курса

Группы ПО-3

Новикевич А.А.

Проверил:

Монтик Н.С.

2020

Лабораторная работа №5

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

**Задание 1:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Здание ← abstract class Общественное Здание ← class Театр.

Код программы:

using System;

namespace ConsoleApp1

{

class Program

{

public interface IBuilding

{

public void Show();

public void Set(string name, int age, string director);

}

public abstract class PublicBuilding : IBuilding

{

public string Name { get; set; }

public int Age { get; set; }

protected PublicBuilding(string name, int age)

{

Name = name;

Age = age;

}

public virtual void Show()

{

Console.WriteLine("Название театра: " + Name + ", возраст здания: " + Age);

}

public abstract void Set(string name, int age, string director);

}

public class Theatre : PublicBuilding

{

public string Director { get; set; }

public Theatre(string name, int age, string director) : base(name, age)

{

Director = director;

}

public override void Show()

{

Console.WriteLine("Название театра: " + Name + ", возраст здания: " + Age + ", Имя директора: " + Director);

}

public override void Set(string name, int age, string director)

{

Name = name;

Age = age;

Director = director;

}

}

static void Main()

{

Theatre student1 = new Theatre("Брестский", 130, "Генадий");

student1.Show();

Console.WriteLine("Изменим название театра, возраст, директора театра");

student1.Set("Театральный", 90, "Василий");

student1.Show();

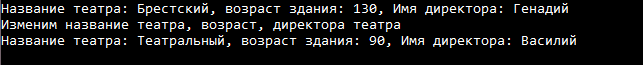
Console.ReadKey();

}

}

}

Результат выполнения:



**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 си-

туаций и объектов.

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

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

class Program

{

abstract class СompanyEmployee

{

protected int money = 0;

protected int zp;

public СompanyEmployee()

{

}

public abstract void SetZp(int zp);

public abstract void GiveZp();

public abstract void ShowMoney();

}

class Manager : СompanyEmployee

{

public override void SetZp(int zp)

{

this.zp = zp;

}

public override void GiveZp()

{

money += zp;

}

public override void ShowMoney()

{

Console.WriteLine($"Общая сумма начислений:{money}, Зарплата:{zp}");

}

}

class Analyst : СompanyEmployee

{

public override void SetZp(int zp)

{

this.zp = zp;

}

public override void GiveZp()

{

money += zp;

}

public override void ShowMoney()

{

Console.WriteLine($"Общая сумма начислений:{money}, Зарплата:{zp}");

}

}

class Programmer : СompanyEmployee

{

public override void SetZp(int zp)

{

this.zp = zp;

}

public override void GiveZp()

{

money += zp;

}

public override void ShowMoney()

{

Console.WriteLine($"Общая сумма начислений:{money}, Зарплата:{zp}");

}

}

class Tester : СompanyEmployee

{

public override void SetZp(int zp)

{

this.zp = zp;

}

public override void GiveZp()

{

money += zp;

}

public override void ShowMoney()

{

Console.WriteLine($"Общая сумма начислений:{money}, Зарплата:{zp}");

}

}

class Designer : СompanyEmployee

{

public override void SetZp(int zp)

{

this.zp = zp;

}

public override void GiveZp()

{

money += zp;

}

public override void ShowMoney()

{

Console.WriteLine($"Общая сумма начислений:{money}, Зарплата:{zp}");

}

}

class Accountant : СompanyEmployee

{

public override void SetZp(int zp)

{

this.zp = zp;

}

public override void GiveZp()

{

money += zp;

}

public override void ShowMoney()

{

Console.WriteLine($"Общая сумма начислений:{money}, Зарплата:{zp}");

}

}

static void Main(string[] args)

{

List<СompanyEmployee> list = new List<СompanyEmployee>();

СompanyEmployee tester = new Tester();

tester.SetZp(500);

list.Add(tester);

СompanyEmployee accountant = new Accountant();

accountant.SetZp(300);

list.Add(accountant);

СompanyEmployee designer = new Designer();

designer.SetZp(200);

list.Add(designer);

СompanyEmployee programmer = new Programmer();

programmer.SetZp(100);

list.Add(programmer);

Console.WriteLine("До выдачи зарплаты:");

foreach (СompanyEmployee emp in list)

{

emp.ShowMoney();

}

foreach (СompanyEmployee emp in list)

{

emp.GiveZp();

}

Console.WriteLine("После выдачи зарплаты:");

foreach (СompanyEmployee emp in list)

{

emp.ShowMoney();

}

foreach (СompanyEmployee emp in list)

{

emp.GiveZp();

}

Console.WriteLine("После ещё одной выдачи зарплаты:");

foreach (СompanyEmployee emp in list)

{

emp.ShowMoney();

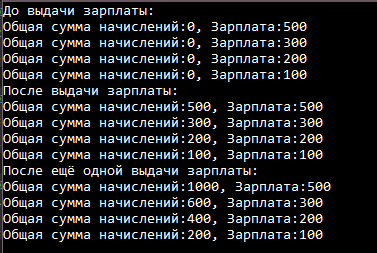
}

Console.ReadKey();

}

}

}



Задание 3: В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных

классов или интерфейсов.

using System;

using System.Collections.Generic;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

String[] bookNames = { "Dead inside", "BSTU MAP", "Dead souls" };

Library library = new Library();

//create books

Book[] books = { new Book(bookNames[0], "Bulgakov", 12),

new Book(bookNames[1], "Gogol", 3),

new Book(bookNames[2], "Tolstoy", 1) };

//create readers

Reader[] readers = { new Reader("Gena Bukin", new DateTime(2003,03,13)),

new Reader("Artyom Nov", new DateTime(2012,06,22) ),

new Reader("Artemij Novik", new DateTime(2020,04,03)) ,

new Reader("Tema Novikecih", new DateTime(2001,11,23))};

//add books in arrayList

foreach (Book book in books)

{

library.addBook(book);

}

//add readers in arrayList

foreach (Reader reader in readers)

{

library.addReader(reader);

}

library.showAll();

Console.WriteLine("\nTry to add Bulgakov to Gena Bukin\n");

library.createOrder(readers[0], books[0]);

Console.WriteLine("\nTry to add Gogol to Gena Bukin\n");

library.createOrder(readers[0], books[1]);

Console.WriteLine("\nTry to add Tolstoi to Artyom Nov\n");

library.createOrder(readers[1], books[2]);

Console.WriteLine("\nTry to add Tolstoi to Artemij Novik\n");

library.createOrder(readers[2], books[2]);

Console.WriteLine("\nTry to add Gogol to Artemij Novik\n");

library.createOrder(readers[2], books[1]);

Console.WriteLine("\nTry to add Tolstoi to Tema Novikecih\n");

library.createOrder(readers[3], books[2]);

foreach (Reader reader in readers)

{

Console.WriteLine("Reader name: " + reader.getName()

+ "\nDate: " + reader.getDate() + "\n");

}

library.showBlackList();

}

}

}

interface Show

{

void showAll();

void createOrder(Reader reader, Book book);

void showBlackList();

void removeBook(Book book);

void removeOrder(Order order);

}

class Library:Show

{

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

private List<Reader> readers = new List<Reader>();

private List<Reader> blackList = new List<Reader>();

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

public void addBook(Book book)

{

books.Add(book);

}

public void createOrder(Reader reader, Book book)

{

if (reader.checkDate(DateTime.Now, reader.getDate()) > 20)

{

blackList.Add(reader);

return;

}

foreach (Book currentBook in books)

{

if (currentBook.getId() == book.getId() && currentBook.getNumber() > 0)

{

Order order = new Order(book.getId(), reader.getId(), book.getTitle());

orders.Add(order);

removeBook(currentBook);

reader.addBook(currentBook);

//removeOrder(order);

}

}

}

public void showBlackList()

{

Console.WriteLine("\nBlack list\n");

foreach (Reader reader in blackList)

{

Console.WriteLine("\nReader name: " + reader.getName() + "\nDate: " + reader.getDate());

}

}

public void showAll()

{

Console.WriteLine("\nAll books in the library:\n");

foreach (Book book in books)

{

Console.WriteLine("\nBooks title: " + book.getTitle()

+ "\nAuthor: " + book.getAuthor()

+ "\nCount: " + book.getNumber() + "\n");

}

}

public void removeBook(Book book)

{

book.setNumber(book.getNumber() - 1);

}

public void removeOrder(Order order)

{

orders.Remove(order);

}

public void addReader(Reader reader)

{

readers.Add(reader);

}

}

class Book

{

private int id;

private String title;

private String author;

private int number;

private static int booksCount = 1;

public Book(String title, String author, int number)

{

id = booksCount++;

setTitle(title);

setAuthor(author);

setNumber(number);

}

public int getId()

{

return id;

}

public int getNumber()

{

return number;

}

public void setNumber(int number)

{

this.number = number;

}

public String getAuthor()

{

return author;

}

public void setAuthor(String author)

{

this.author = author;

}

public String getTitle()

{

return title;

}

public void setTitle(String title)

{

this.title = title;

}

}

class Reader {

private int id;

private String name;

private DateTime bookDate = DateTime.Now;

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

private static int readersCount = 1;

public Reader(String name, DateTime date)

{

id = readersCount++;

setName(name);

setDate(date);

}

public int getId()

{

return id;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public void setDate(DateTime date)

{

bookDate = date;

}

public DateTime getDate()

{

return bookDate;

}

public long checkDate(DateTime date, DateTime bookDate)

{

TimeSpan difference = date - bookDate;

long days = (long)difference.TotalDays;

return days;

}

public void addBook(Book book)

{

books.Add(book);

}

public void showAll()

{

foreach (Book book in books){

if (books.Count == 0) { Console.WriteLine("There is not any books"); }

else

{

Console.WriteLine("Book name: " + book.getTitle() + "\n" +"Author: " + book.getAuthor() + "\n");

}

}

}

}

class Order

{

private int id;

private int bookId;

private int readerId;

private String bookTitle;

private static int ordersCount = 1;

public Order(int bookId, int readerId, String bookTitle)

{

id = ordersCount++;

setBookId(bookId);

setReaderId(readerId);

setBookTitle(bookTitle);

}

public int getReaderId()

{

return readerId;

}

public void setReaderId(int readerId)

{

this.readerId = readerId;

}

public int getBookId()

{

return bookId;

}

public void setBookId(int bookId)

{

this.bookId = bookId;

}

public void setBookTitle(String bookTitle)

{

this.bookTitle = bookTitle;

}

public String getBookTitle()

{

return bookTitle;

}

}