МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

**О Т Ч Ё Т**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

**«Объектно-ориентированное программирование на языке C#. Абстракция.»**

**по курсу:**

***"Конструирование программного обеспечения"***

|  |  |
| --- | --- |
| Выполнили, студенты гр. 10701121: | Качан И. В.  Длусский К. Ю. |
|  |  |
| Проверил: | Станкевич С. Н. |

Минск – 2023

**Цель работы:**

Научиться грамотно анализировать предметную область и с помощью абстракции выделять существенные детали, на базе которых в дальнейшем проектируются классы и объекты будущей программной системы согласно методологии ООП, а также практически закрепить данные навыки при решении  
соответствующих задач (бизнес проблем).

**Требования**:

1) Необходимо спроектировать и реализовать UML-диаграмму взаимодействия  
классов и объектов разрабатываемой программной системы с отображением  
всех связей (отношений) между классами и объектами.  
2) При проектировании и разработке системы необходимо полностью использовать своё объектно-ориентированное воображение и по максимум использовать возможности, которые предоставляет язык программирования Java для  
реализации ООП-методологии.  
3) Основные классы системы должны быть самодостаточными, т.е. не зависеть, к  
примеру, от консоли! Любые типы отношений между классами должны применяться обосновано и лишь тогда, когда это имеет смысл.  
4) При выполнения задания необходимо по максимуму пытаться разрабатывать  
универсальный, масштабируемый, легко поддерживаемый и читаемый код.  
5) Также рекомендуется придерживаться Single Responsibility Principle, SRP  
(принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только  
одна причина изменить в дальнейшем соответствующий блок кода.  
6) Создаваемые классы необходимо грамотно разложить по соответствующим  
пакетам, которые должны иметь «адекватные» названия и быть вложены в  
указанные стартовые пакеты: by.bntu.fitr.poisit.nameofstudent.nameofproject.  
7) В соответствующих компонентах бизнес-логики необходимо предусмотреть  
«защиту от дурака».  
8) На базе спроектированной программной системы реализуйте простейшее интерактивное консольное приложение. Используйте при реализации архитектурный шаблона проектирования Model-View-Controller, MVC.  
9) Программа должна обязательно быть снабжена комментариями, в которых  
необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчиков, название  
бригады (если есть), номер группы и дату разработки. Исходный текст классов  
и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.  
10) Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем.  
11) Интерфейс программы и комментарии должны быть на английском языке.  
12) Необходимо проверить все тестовые случаи работы основной бизнес-логики  
программы.  
13) При разработке программ придерживайтесь соглашений по написанию кода

**Основное задание**:

Необходимо решить задачу с использованием методологии ООП. Для чего  
необходимо подобрать самостоятельно соответствующую проблемную  
(предметную/доменную) область, которая базируется на объектах и событиях  
реального мира (примеры соответствующих предметных областей приведены  
ниже). Спроектировать классы (собственные пользовательские типы данных)  
в языке Java для программного представления данных объектов и основной  
логики будущей программной системы.  
Система должна решать, как минимум, два полезных действия и иметь дополнительно следующие опции:  
− не менее 3 разнообразных классов предметной области;  
− не менее 5 атрибутов (состояния) и методов (поведения) в классе-сущности;  
− не менее 3 методов, которые реализуют бизнес-логику программы, в соответствующих функциональных классах;  
− хранить глобальные характеристики системы или характеристики уровня  
отдельных классов.  
На базе спроектированной программной системы реализовать программу и  
продемонстрировать её работоспособность.

**Предметная область:**

Компания по разработки программного обеспечения, IT-компания  
(Software Company). Есть IT-компании, которые состоят из соответствующих  
сотрудников. Каждый сотрудник за свою работу получает соответствующую  
зарплату. Необходимо найти общий фонд заработной платы компании на месяц, а также определить сотрудников с максимальной (минимальной) заработной платой

**Результаты выполнения программы:**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 1. Добавление сотрудников**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 2. Вывод всех сотрудников и их суммарная зарплата за месяц.**

**Изображение выглядит как текст

Автоматически созданное описание**

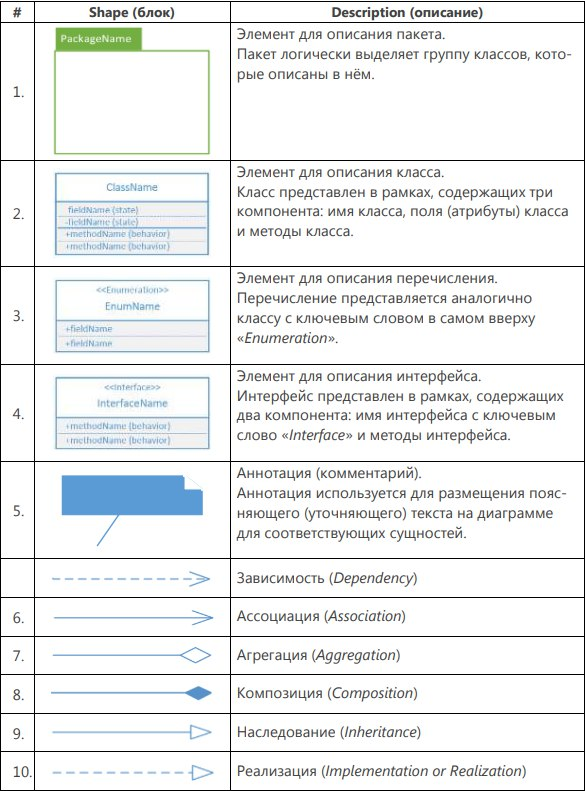
**Рисунок 3. Вывод сотрудников с минимальной и максимальной зарплатой.**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 4. Повышение зарплаты сотруднику.**

**UML-диаграмма классов:**



Изображение выглядит как диаграмма

Автоматически созданное описание

**Рисунок 5. UML-диаграмма классов.**

**Листинг исходного кода программы:**

**Класс Employee.cs**

using System;

using System.Collections.Generic;

namespace KPOlaba1

{

abstract class Employee

{

private string name;

private double salary;

private string rank;

private double salaryRatio;

protected static double startSalary = 500;

public static Dictionary<String, double> rankDict = new Dictionary<string, double>{ { "junior", 0 }, { "middle", 0 }, { "senior", 0 } };

public string Name

{

get => name;

set => name = value;

}

public virtual double Salary

{

get => salary;

set => salary = value;

}

public virtual double SalaryRatio

{

get => salaryRatio;

set => salaryRatio = value;

}

public string Rank

{

get => rank;

set => rank = value;

}

public static double StartSalary

{

get => startSalary;

set => startSalary = value;

}

public Employee(string name, string rank)

{

Name = name;

Rank = rank;

}

public Employee() { }

public abstract void PrintInfo(); // вывод информации о сотруднике

}

}

**Класс SoftwareEngineer.cs**

using System;

using System.Collections.Generic;

namespace KPOlaba1

{

internal class SoftwareEngineer : Employee

{

public SoftwareEngineer(string name, string rank) : base(name, rank)

{

rankDict = new Dictionary<String, double>() { { "junior", 2 }, { "middle", 2.5 }, { "senior", 4 } };

SalaryRatio = rankDict[Rank];

Salary = startSalary \* SalaryRatio;

}

public SoftwareEngineer() : base() { }

public override void PrintInfo() // вывод информации о сотруднике

{

Console.WriteLine("--------------------");

Console.WriteLine("Engineer info:");

Console.WriteLine("Name: {0} Rank: {1} Salary {2}", Name, Rank, Salary);

Console.WriteLine("--------------------\n");

}

}

}

**Класс QAengineer.cs**

using System;

using System.Collections.Generic;

namespace KPOlaba1

{

internal class QAengineer : Employee

{

public QAengineer(string name, string rank) : base(name,rank)

{

rankDict = new Dictionary<String, double>() { { "junior", 1 }, { "middle", 1.5 }, { "senior", 2 } };

SalaryRatio = rankDict[Rank];

Salary = startSalary \* SalaryRatio;

}

public QAengineer() : base() { }

public override void PrintInfo() // вывод информации о сотруднике

{

Console.WriteLine("--------------------");

Console.WriteLine("QA Engineer info:");

Console.WriteLine("Name: {0} Rank: {1} Salary {2}", Name, Rank, Salary);

Console.WriteLine("--------------------\n");

}

}

}

**Класс ListOfEmployeers.cs**

using System;

using System.Collections.Generic;

using System.Web.UI;

using System.Xml.Linq;

namespace KPOlaba1

{

abstract class ListOfEmployeers

{

public static List<Employee> empoyees = new List<Employee>();

public static void AddEmployee() // добавить сотрудника

{

Console.Write("Name: ");

string name = Console.ReadLine();

Console.Write("Rank(junior, middle, senior): ");

string rank = Console.ReadLine();

if (Employee.rankDict.ContainsKey(rank) == false)

{

Console.WriteLine("Invalid input");

return;

}

Console.Write("Speciality(QA, Engineer): ");

string speciality = Console.ReadLine();

switch (speciality)

{

case "Engineer":

empoyees.Add(new SoftwareEngineer(name, rank));

break;

case "QA":

empoyees.Add(new QAengineer(name, rank));

break;

default:

Console.WriteLine("Invalid input\n");

break;

}

}

public static void DeleteEmployee() // удалить сотрудника

{

Console.WriteLine("Enter a name: ");

string name = Console.ReadLine();

if (empoyees.Count != 0)

{

foreach (Employee empoyee in empoyees)

{

if (empoyee.Name == name)

{

empoyees.RemoveAll(x => x.Name == name);

Console.WriteLine("Success\n");

break;

}

else

{

Console.WriteLine("There is no employee with this name\n");

}

}

}

else

{

Console.WriteLine("List of emloyeers is empty\n");

}

}

public static Pair FindMaxMinSalary() // поиск максимальной и минимальной зарплаты

{

List<int> min = new List<int>();

List<int> max = new List<int>();

Pair pair = new Pair();

int indexMin = 0, indexMax = 0;

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

{

if (empoyees[indexMax].Salary < empoyees[i].Salary)

{

max.Clear();

indexMax = i;

max.Add(indexMax);

continue;

}

else

{

if (empoyees[indexMax].Salary == empoyees[i].Salary)

{

max.Add(i);

continue;

}

}

}

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

{

if (empoyees[indexMin].Salary > empoyees[i].Salary)

{

min.Clear();

indexMin = i;

min.Add(indexMin);

continue;

}

else

{

if (empoyees[indexMin].Salary == empoyees[i].Salary)

{

min.Add(i);

continue;

}

}

}

pair.First = min;

pair.Second = max;

return pair;

}

public static void PrintMinMax() // вывести людей с максимальной и минимальной зарплатой

{

Console.WriteLine("Info about employeers with max and min salary:");

Pair pair = FindMaxMinSalary();

List<int> min = (List<int>)pair.First;

List<int> max = (List<int>)pair.Second;

Console.WriteLine("Minimum salary: ");

foreach (int indexMin in min)

{

empoyees[indexMin].PrintInfo();

}

Console.WriteLine("---------");

Console.WriteLine("Maximum salary: ");

foreach (int indexMax in max)

{

empoyees[indexMax].PrintInfo();

}

Console.WriteLine("---------");

}

public static void RaiseSalary() // увеличить/уменьшить зарплату

{

Console.Write("Enter employee name: ");

string name = Console.ReadLine();

Console.Write("Enter amount of money: ");

double.TryParse(Console.ReadLine(), out double amount);

foreach (Employee empoyee in empoyees)

{

if (empoyee.Name == name)

{

empoyee.Salary += amount;

}

}

}

public static void PrintEmployeersInfo() // вывести всех сотрудников

{

Console.WriteLine("List of employeers:\n");

double employeeMonthSalary = 0;

foreach (Employee empoyee in empoyees)

{

employeeMonthSalary += empoyee.Salary;

empoyee.PrintInfo();

}

Console.WriteLine("\n\nEmployees salary for a month: {0}", employeeMonthSalary);

}

public static void ChangeStartSalary() // изменить стартовую зарплату

{

bool isCorrect = true;

Console.WriteLine("Enter new start salary: ");

isCorrect = double.TryParse(Console.ReadLine(), out double newStartSalary);

if (isCorrect == true)

{

foreach (Employee employee in empoyees)

{

employee.Salary = employee.Salary \* (newStartSalary / Employee.StartSalary);

}

Employee.StartSalary = newStartSalary;

}

else

{

Console.WriteLine("Invalid Input");

}

}

}

}

**Класс Program.cs**

using System;

using System.Collections.Generic;

using System.Web.UI;

/\*

\* Console application "IT Company"

\* CEO - Kachan Ivan, CTO - Dlussky Kirill. Group 10701121

\* 23.03.2023

\* version 1.0

\*/

namespace KPOlaba1

{

internal class Program

{

static void Main(string[] args)

{

bool isExit = false;

while (isExit == false)

{

Console.WriteLine("Select an action:");

Console.WriteLine("1. Add new Employee\n2. Delete employee\n3. Raise employee salary\n4. Print list of employees\n5. Find employee with MAX and MIN salary" +

"\n6. Change start salary\n7. Exit");

int.TryParse(Console.ReadLine(), out int input);

switch (input)

{

case 1:

ListOfEmployeers.AddEmployee();

break;

case 2:

ListOfEmployeers.DeleteEmployee();

break;

case 3:

ListOfEmployeers.RaiseSalary();

break;

case 4:

ListOfEmployeers.PrintEmployeersInfo();

break;

case 5:

ListOfEmployeers.PrintMinMax();

break;

case 6:

ListOfEmployeers.ChangeStartSalary();

break;

case 7:

isExit = true;

break;

default:

Console.WriteLine("\nInvalid input! Try again\n");

break;

}

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

}

}

}

}

**Контрольные вопросы**

1) Какова была главная идея создания процедурных языков программирования? На что эти языки программирования ставили акцент и что являлось  
исходным понятием методологии?  
2) К чему не были готовы процедурные языки программирования в начале  
2000 годов? Какой был кризис в сфере разработки программного обеспечения?  
3) Суть объектно-ориентированного подхода и его главный лозунг?  
4) Опишите базовые концепции, которые лежат в основе методологии ООП.  
5) Что такое объект в реальном мире? Чем он характеризуется в реальном  
мире? (*тип, состояние, поведение, уникальность, эквивалентность*)  
6) Что такое объект в программировании и для чего он вводится при моделировании будущей программной системы?  
7) Чем характеризуется объект в программном (виртуальном) мире? Как реализуются данные вещи в объектно-ориентированных языках программирования?  
8) Какая разница между уникальностью и эквивалентностью?  
9) Какую роль выполняет класс в ООП?  
10)Как и что можно описать в Java-классе?  
11)Что представляет собой абстракция в ООП?  
12)Простейшая форма абстракции в ООП?  
13)Какие разновидности классов существуют в ООП?  
14)Каким способом можно создать объект (экземпляр класса) в языке Java?  
15)Как можно обратиться к полям и методам объекта в языке Java?  
16)Какова разница между классом и объектом в языке Java?  
17)Перечислите преимущества и недостатки объектно-ориентированного  
программирования.  
18) Что такое UML?  
19) Что отображает и зачем нужна UML-диаграмма классов?  
20)Как с помощью UML можно описать классы и их взаимосвязь друг с другом (т.е.  
описать)?

**Ответы на контрольные вопросы**

1)Главная идея создания процедурных языков программирования заключалась в упрощении процесса программирования путем разбиения задач на более мелкие части, называемые процедурами или функциями. Акцент ставился на процедурном подходе к программированию, который подразумевал разделение кода на небольшие, логически связанные блоки. Исходным понятием методологии было процедурное программирование.

2)В начале 2000-х годов процедурные языки программирования не были готовы к эффективной работе с большими проектами, содержащими множество компонентов. В этот период произошел кризис в сфере разработки программного обеспечения, связанный с неспособностью процедурных языков программирования обеспечить эффективное управление сложными проектами.

3)Основная идея объектно-ориентированного подхода заключается в том, что программа строится на основе объектов, которые взаимодействуют друг с другом для выполнения задач. Главный лозунг объектно-ориентированного подхода - "все является объектом".

4)Базовые концепции, лежащие в основе методологии ООП, включают инкапсуляцию, наследование и полиморфизм. Инкапсуляция означает скрытие реализации объекта от внешнего мира, чтобы обеспечить безопасность и сокрыть детали реализации. Наследование позволяет создавать новые классы на основе уже существующих, наследуя их свойства и методы. Полиморфизм означает использование одного имени для нескольких функций, которые могут иметь различные реализации.

5)Объект в реальном мире - это конкретный экземпляр какого-либо предмета или явления, характеризующийся типом, состоянием, поведением, уникальностью и эквивалентностью.

6)Объект в программировании - это абстрактный тип данных, который используется для моделирования реального мира. Он вводится для того, чтобы представить сущности и связи между ними в программной системе.

7)В программном (виртуальном) мире объект характеризуется как экземпляр класса, имеющий определенные свойства (поля) и методы. В объектно-ориентированных языках программирования данные вещи реализуются путем создания классов, которые определяют структуру объектов, и создания экземпляров этих классов в памяти компьютера во время выполнения программы.

8)Уникальность означает, что каждый объект имеет свой уникальный идентификатор в памяти компьютера. Эквивалентность означает, что два объекта могут иметь одинаковые свойства и методы, и с точки зрения функциональности программы они могут быть эквивалентны.

9)Класс в ООП выполняет роль шаблона, определяющего свойства и методы, которые будут иметь объекты этого класса. Он также определяет правила для создания, использования и уничтожения объектов.

10)В классе можно описать свойства объекта (поля) и его методы, включая их аргументы и возвращаемые значения. Также можно определить конструкторы класса.

11)Абстракция в ООП представляет собой процесс выделения важных характеристик объекта и игнорирования менее важных.

12)Простейшая форма абстракции в ООП - это создание класса, который может иметь только свойства, но не методы.

13)Разновидности классов в ООП включают абстрактные классы, интерфейсы и анонимные классы.

14)Объект (экземпляр класса) в языке C# можно создать с помощью оператора "new", например: MyClass obj = new MyClass();

15)Для обращения к полям и методам объекта в языке C# используется оператор точки, например: obj.myMethod(); obj.myField = 42;

16)Класс в Java - это шаблон, определяющий структуру объектов, а объект - это конкретный экземпляр класса, созданный в памяти компьютера во время выполнения программы.

17)Преимущества объектно-ориентированного программирования:

Легкость в поддержке кода и добавлении новых функций

Высокий уровень абстракции и модульности, что способствует повторному использованию кода и сокращению времени разработки

Удобство работы с большими и сложными проектами, поскольку объекты взаимодействуют друг с другом в рамках четко определенных интерфейсов

Более естественное отображение реального мира, что упрощает понимание и разработку программы

Недостатки объектно-ориентированного программирования:

Сложность для начинающих программистов, поскольку требуется понимание многих концепций ООП

Увеличение сложности кода при использовании наследования и полиморфизма

Увеличение объема памяти и времени выполнения программы в связи с созданием объектов и их хранением в памяти

Сложность отладки программы в связи с большим количеством взаимосвязанных объектов

18)UML (Unified Modeling Language) - это стандартный язык графического моделирования, используемый для создания и визуализации моделей программного обеспечения. UML предоставляет нотации и методы для описания различных аспектов системы, таких как структура, поведение, взаимодействие и архитектура.

19)UML-диаграмма классов отображает структуру классов в системе и связи между ними. Она помогает программистам лучше понимать архитектуру системы и ее компонентов, а также может использоваться для создания новых компонентов, документации и рефакторинга кода.

20)С помощью UML можно описать классы и их взаимосвязи с помощью диаграмм классов. Диаграмма классов представляет собой графическое изображение классов, их атрибутов и методов, а также связей между классами.

В диаграмме классов классы изображаются в виде прямоугольников, в которых указывается название класса. Атрибуты класса отображаются в виде списка, а методы - в виде списка или таблицы. Связи между классами изображаются в виде стрелок, которые указывают направление связи и ее тип.

**Вывод:**

Научились грамотно анализировать предметную область и с помощью абстракции выделять существенные детали, на базе которых в дальнейшем проектируются классы и объекты будущей программной системы согласно методологии ООП, а также практически закрепили навыки при решении соответствующих задач (бизнес проблем).