ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ  
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

навчальна дисципліна «Об'єктно-орієнтоване програмування»

ЗВІТ

про виконання лабораторної роботи № 2

на тему: «Проєктування класів з використанням мови UML»

Варіант № 11

Виконав:

студент 2 курсу

групи 6.04.121.010.23.2

інститу ІТ

Місюра О.А

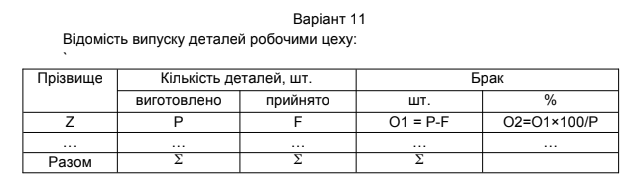
Перевірив:

доцент кафедри ІС

Бредіхін В.М.

м. Харків – 2024 рік

Завдання 1.



Умова завдання:

* Розробити об'єктну модель, яка включає дані про студентів: прізвище, пропущені години, кількість пропущених годин у відсотках.
* Описати відповідні класи та їхні методи.
* Зробити UML-діаграму, яка відображатиме структуру класів та їх зв'язки.

Код:

using System;

using System.Collections.Generic;

namespace WorkshopManagement

{

public class Worker

{

public string Name { get; set; }

public int MissedHours { get; set; }

public int ProducedDetails { get; set; }

public int AcceptedDetails { get; set; }

public int CalculateDefectCount()

{

return ProducedDetails - AcceptedDetails;

}

public double CalculateDefectPercentage()

{

if (ProducedDetails == 0)

return 0;

return (double)CalculateDefectCount() \* 100 / ProducedDetails;

}

}

public class Workshop

{

public List<Worker> Workers { get; set; } = new List<Worker>();

public void AddWorker(Worker worker)

{

Workers.Add(worker);

}

public int CalculateTotalProduced()

{

int totalProduced = 0;

foreach (var worker in Workers)

{

totalProduced += worker.ProducedDetails;

}

return totalProduced;

}

public int CalculateTotalAccepted()

{

int totalAccepted = 0;

foreach (var worker in Workers)

{

totalAccepted += worker.AcceptedDetails;

}

return totalAccepted;

}

public int CalculateTotalDefects()

{

int totalDefects = 0;

foreach (var worker in Workers)

{

totalDefects += worker.CalculateDefectCount();

}

return totalDefects;

}

public double CalculateTotalDefectPercentage()

{

int totalProduced = CalculateTotalProduced();

if (totalProduced == 0)

return 0;

return (double)CalculateTotalDefects() \* 100 / totalProduced;

}

}

class Program

{

static void Main(string[] args)

{

Workshop workshop = new Workshop();

workshop.AddWorker(new Worker { Name = "Іванов", MissedHours = 2, ProducedDetails = 100, AcceptedDetails = 90 });

workshop.AddWorker(new Worker { Name = "Петров", MissedHours = 1, ProducedDetails = 120, AcceptedDetails = 115 });

workshop.AddWorker(new Worker { Name = "Сидоров", MissedHours = 0, ProducedDetails = 80, AcceptedDetails = 75 });

Console.WriteLine("Загальна кількість виготовлених деталей: " + workshop.CalculateTotalProduced());

Console.WriteLine("Загальна кількість прийнятих деталей: " + workshop.CalculateTotalAccepted());

Console.WriteLine("Загальна кількість браку: " + workshop.CalculateTotalDefects());

Console.WriteLine("Середній відсоток браку: " + workshop.CalculateTotalDefectPercentage() + "%");

foreach (var worker in workshop.Workers)

{

Console.WriteLine($"\nРобітник: {worker.Name}");

Console.WriteLine("Кількість пропущених годин: " + worker.MissedHours);

Console.WriteLine("Відсоток браку: " + worker.CalculateDefectPercentage() + "%");

}

}

}

}

Опис класів та їхні методи:

**Клас Worker**

Цей клас представляє одного робітника, який виробляє деталі. Він зберігає основні дані про робітника та дозволяє обчислювати кількість браку і відсоток браку для кожного робітника.

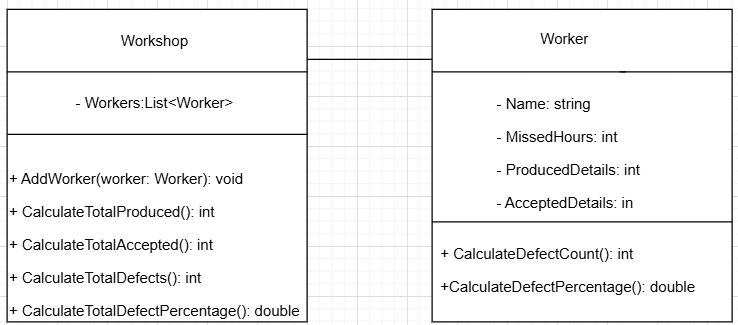
* Поля:
  + Name — рядок, який зберігає ім'я або прізвище робітника.
  + MissedHours — кількість пропущених годин (ціле число).
  + ProducedDetails — кількість виготовлених деталей (ціле число).
  + AcceptedDetails — кількість прийнятих деталей (ціле число).
* Методи:
  + CalculateDefectCount() — обчислює кількість браку (кількість виготовлених мінус кількість прийнятих деталей). Повертає ціле число, яке представляє кількість дефектних деталей.
  + CalculateDefectPercentage() — обчислює відсоток браку, використовуючи формулу: (кількість браку \* 100) / кількість виготовлених деталей. Якщо ProducedDetails дорівнює 0, повертає 0. Результат повертається у вигляді числа з плаваючою комою (double).

**Клас Workshop**

Клас представляє цех і містить інформацію про всіх робітників, а також методи для підрахунку загальних показників.

* Поля:
  + Workers — список об'єктів Worker. Зберігає всіх робітників у цеху.
* Методи:
  + AddWorker(Worker worker) — додає робітника до списку Workers.
  + CalculateTotalProduced() — обчислює загальну кількість виготовлених деталей всіма робітниками в цеху. Повертає ціле число.
  + CalculateTotalAccepted() — обчислює загальну кількість прийнятих деталей всіма робітниками. Повертає ціле число.
  + CalculateTotalDefects() — обчислює загальну кількість дефектів для всіх робітників у цеху. Повертає ціле число.
  + CalculateTotalDefectPercentage() — обчислює середній відсоток браку для всього цеху. Використовує загальну кількість браку та загальну кількість виготовлених деталей. Якщо загальна кількість виготовлених деталей дорівнює 0, повертає 0. Результат повертається як double

UML діаграма:



Завдання 2

using System;

using System.Collections.Generic;

namespace WorkshopManagement

{

public class Worker

{

public string Name { get; set; }

public int MissedHours { get; set; }

public int ProducedDetails { get; set; }

public int AcceptedDetails { get; set; }

public Worker(string name, int missedHours, int producedDetails, int acceptedDetails)

{

Name = name;

MissedHours = missedHours;

ProducedDetails = producedDetails;

AcceptedDetails = acceptedDetails;

}

public int CalculateDefectCount()

{

return ProducedDetails - AcceptedDetails;

}

public double CalculateDefectPercentage()

{

if (ProducedDetails == 0)

return 0;

return (double)CalculateDefectCount() \* 100 / ProducedDetails;

}

}

public class Workshop

{

private List<Worker> Workers { get; } = new List<Worker>();

public void AddWorker(string name, int missedHours, int producedDetails, int acceptedDetails)

{

Workers.Add(new Worker(name, missedHours, producedDetails, acceptedDetails));

}

public int CalculateTotalProduced()

{

int totalProduced = 0;

foreach (var worker in Workers)

{

totalProduced += worker.ProducedDetails;

}

return totalProduced;

}

public int CalculateTotalAccepted()

{

int totalAccepted = 0;

foreach (var worker in Workers)

{

totalAccepted += worker.AcceptedDetails;

}

return totalAccepted;

}

public int CalculateTotalDefects()

{

int totalDefects = 0;

foreach (var worker in Workers)

{

totalDefects += worker.CalculateDefectCount();

}

return totalDefects;

}

public double CalculateTotalDefectPercentage()

{

int totalProduced = CalculateTotalProduced();

if (totalProduced == 0)

return 0;

return (double)CalculateTotalDefects() \* 100 / totalProduced;

}

}

class Program

{

static void Main(string[] args)

{

Workshop workshop = new Workshop();

workshop.AddWorker("Іванов", 2, 100, 90);

workshop.AddWorker("Петров", 1, 120, 115);

workshop.AddWorker("Сидоров", 0, 80, 75);

Console.WriteLine("Загальна кількість виготовлених деталей: " + workshop.CalculateTotalProduced());

Console.WriteLine("Загальна кількість прийнятих деталей: " + workshop.CalculateTotalAccepted());

Console.WriteLine("Загальна кількість браку: " + workshop.CalculateTotalDefects());

Console.WriteLine("Середній відсоток браку: " + workshop.CalculateTotalDefectPercentage() + "%");

}

}

}

**Висновок:** У цій роботі було розроблено об'єктну модель для системи обліку виготовлення деталей у робочому цеху. Створено класи Workshop (цех) та Worker (робітник), між якими реалізовано композиційний зв'язок: робітники існують тільки в межах конкретного цеху. Це забезпечило високу інкапсуляцію та цілісність, дозволивши централізовано обробляти дані про робітників, підраховувати виготовлені та браковані деталі, а також їхній відсоток. Модель має чітку структуру, легко масштабована і дозволяє ефективно контролювати якість виробництва.