# Workshop

## Part1

Да се напише ОО програма за персонал во некоја компанија. Компанијата располага со вработени (Employee) од следниве профили: технички персонал (TechnicalStuff), сметководители (Accountant), роботи (Robot), менаџери (Manager), и директори (Director).

За секој вработен се чуваат следните податоци: ID, Name, Surname, Title, BirthDate, HireDate.

**Прв дел**

Да се напише скелетот од класи, методи и пропертиа за задачата.

|  |  |
| --- | --- |
| **Класа** | **Метод** |
| TechnicalStuff | void MaintainSystems() |
| Accountant | void MaintainAccounts() |
| Robot | void MaintaingDatawareHouse() |
| Manager | void DoEmployeeApprisal(Employee e)  void PromoteEmployee(Employee e) |
| Director | void DoEmployeeApprisal(Employee e)  void PromoteEmployee(Employee e) |

Заради заштеда на време, методите доволно е да печатат некакво инфо за ентитетите.

1. Да се напишаат класите за горенаведените ентитети.
2. Да се оверрајдува витуелниот метод ToString() така што ќе печати детали за објектите од наведените ентитети.
3. Да се напишаат методите.
4. Да се инстанцираат објекти од сите класи и да се повикаат методите.

**Втор дел**

Методите *void DoEmployeeApprisal(Employee e)* и *void PromoteEmployee(Employee e)* кои треба да ги имате веќе во Manager и Director, да се префрлат во нова класа ManagementStuff. Во истата класа да се додаде нов метод *void SendCommunication()* кој ќе го имплементираат класите Manager и Director. Класите Manager и Director треба да ги задржат пропертијата и методите што претходно ги имале.

**Трет дел**

Да се креира фолдер Interfaces и таму да се додадат интерфејси. Да се додадат следните интерфејси со методите и потоа да се имплементираат кон соодветните класи:

|  |  |  |
| --- | --- | --- |
| Интерфејс | Метод | Класи |
| IWorkable | Void Work() | TechnicalStuff, Robot |
| IEatable | Void Eat() | TechnicalStuff, Accountant |
| IManagable | Void Manage() | ManagementStaff, Accountant |

Напомена: додавањето на методи кон ManagementStuff класата значи достапност на методите и во Manager, Director класите. Заради едноставност, методите печатат информативни пораки.

## Part2

**Четврти дел**:

А) Да се додадат *Email* и *Salary* пропертиа во класата *Employee* и да се прошири *ToString*(). Сите објекти во Main да се прошират со податок за *Email* и *Salary*.

Б) Да се прошири класата *ManagementStuff* со листа од вработени кои се наоѓаат под некој менаџер/директор. Листата да биде приватна и додавањето на вработен да биде преку метод *AddSubEmployee(Employee e).* Потоа да се додаде метод *GetEmployees*() кој ќе враќа листа од сите вработени кои репортираат на мендаџер/директор -от.

В) Да се додаде string проперти *EmailList* во класата *ManagementStuff* кое само ќе ја враќа листата од emails (сепаратор “;”) на вработените кои репортираат на мендаџер/директор -от. Исто така да се оверајдува *ToString*() во оваа класа така што дополнително ќе враќа *EmailList* на крај.

Г) Во Main, да се употребат ново-додадените работи.

**Петти дел:**

Да се разгледа класата *Logger*.cs. Да се дискутира ако има прашања.

Во сите методи каде што се употребува *Console.WriteLine*() да се замени со *Log*() методот од *Logger* класата со истата порака. Потоа да се изврши програмата и да се види што е запишано во лог фајлот на Desktop.

Да се креира метод *ArchiveLog()* кој ако постои Log.txt фајл на Desktop ќе го премести (Move) во фолдер *Archive* на Desktop под име Log\_YYYYMMDDHHMMSS.txt. Ако не постои фолдер *Archive*, методот да го креира истиот еднократно.

**Шести дел**:

Да се напише метод *PromoteSubEmployees*(*int totalPromotionAmount*) во *ManagementStaff* класата кој дадена сума ќе ја подели подеднакво на сите вработени под менаџер/директор –от.

Ако менаџерот/директорот немаат вработени под нив, програмата треба да излогира порака од тип Error (енум *LogType*) во фајл со користење на *Logger* класата.

**Седми дел:**

Методите *SendCommunication*() во класите Manager и Director да се трансфоримраат така што ќе работат со *async/await* (може да се употреби помошен метод). Да се осигури дека кодот се извршува без грешка.

Да се проверат времињата на запишување во Log.txt.

**Осми дел:**

Да се употреби publisher subscriber моделот. Менаџерот/Директорот можат да промовираат EmployeeOfTheMonth. Сите вработени се заинтересирани да дознаат кој е вработен на месецот. Секој вработен кој е subscribe-иран на event-от EmployeeOfTheMonth треба да провери дали тој е избран за работник на месецот и доколку е да го испечати property-to GratitudeText, а доколку не е да го испечати property-то CongratulationText.

* Во **publisher** класата **ManagegementStuff** треба да се декларира делегат

public delegate void EventHandler(object sender, EventArgs e);

* Во **publisher** класата **ManagegementStuff** треба да се декларира event

public event EventHandler EmployeeOfTheMonth;

* Во **subscriber** класата **Employee** треба да се направи метод **Subscribe** кој што ќе направи subscription на event-от **EmployeeOfTheMonth**

EmployeeOfTheMonth += new ManagementStuff.EventHandler(manageableStuff\_EmployeeOfTheMonth);

* Во **subscriber** класата **Employee** во

public void manageableStuff\_EmployeeOfTheMonth(object sender, EventArgs e)

треба да се провери дали вработениот е избран за работник на месецот или не. Доколку е избран да се испечати property-то **GratitudeText**, во спротивно property-то **CongratulationText**.

* Во **publisher** класата **ManagegementStuff** во методот

public void PromoteEmployeeOfTheMonth(Employee e)

треба да се провери дали има **subscribe**-ри на **event**-от

(EmployeeOfTheMonth != null)

и доколку има да се креира објект од класата EmployeeEventArgs : EventArgs

EmployeeEventArgs employeeInfo = new EmployeeEventArgs() { Name = e.Name };

и да се стартува **event**-от

EmployeeOfTheMonth(this, employeeInfo);