**Фирми**

**Общ преглед**

Във вашата фирма постъпва проект за създаване на приложение, обслужващо големи корпорации.

Вашият софтуер трябва да описва **компания** **(**Company**)** и **служител (**Employee**).**

Tрябва да реализирате функционалност, която да позволява създаване на компании и служители, а в последствие наемане/уволнение и справки за тях – всичко това ще работи чрез **команди**, които вие ще получавате. Поредицата от команди приключва с „**End**”. За ваше удобство ще получите готов Program.cs файл, и ще трябва да реализирате само необходимите класове Company.cs и Employee.cs

**Подзадача 1: Структура на Служител – 15 точки**

**Employee**

Всички служители имат id, име, презиме, години и заплата:

id – низ, съставен от малки и големи латински букви

firstName – низ, съставен от малки и/или големи латински букви

lastName – низ, съставен от малки и/или големи латински букви

age – цяло число

salary – число с плаваща запетая. Ако не е подадено – стойност по подразбиране - 500.0;

Реализирайте два конструктора:

* public Employee(string id, string firstName, string lastName, int age) –  този конструктор трябва да приема id, име, фамилия и години на служителя.
* public Employee(string id, string firstName, string lastName, int age, double salary) –  този конструктор трябва да приема id, име, фамилия, години и **заплата** на служителя.

За справка вижте следната схема за Employee.cs:

|  |
| --- |
| Employee.cs |
| public Employee(string id, string firstName, string lastName, int age)  {      //TODO: Добавете вашия код тук …  }    public Employee(string id, string firstName, string lastName, int age, double salary)  {      //TODO: Добавете вашия код тук …  } |

**Командa за добавяне на служител**

Вашето приложение трябва да обслужва следната команда за добавяне на служители:

* **CreateEmployee**<**id**>  <**име**>  <**фамилия**> <**възраст**>  - тази команда има за цел да добави служител с неговите id, име, фамилия, възраст.
* **CreateEmployee**<**id**>  <**име**>  <**фамилия**> <**възраст**> <**заплата**>  - тази команда има за цел да добави служител с неговите id, име, фамилия, възраст и **заплата.**

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за служител. Командата за това е следната:

* **PrintEmployeeInfo**<**id**> - отпечатва информация за служител във формат:  
  Employee: **<име>**  **<фамилия>** with id: **<id>**  and salary: **<заплата>**
* Тази команда ще получава винаги валидни и съществуващи id-та на служители. За успешна реализация трябва да реализирате ваша версия на **ToString()**метода за класа **Employee.**

**\*Заплатата трябва да бъде форматирана до втория знак след десетичния разделител.**

**Подзадача 2: Структура на Компания – 15 точки**

**Company**

Всяка компания има: име, списък от служители, град.

name – низ, съставен от малки и големи латински букви

employees – списък от **Employee**

city – низ, съставен от малки и големи латински букви

Трябва да реализирате констуктор:

* public Company(string name, string city)– трябва да приема име и град.

За справка вижте следната схема за Company.cs:

|  |
| --- |
| Company.cs |
| public Company(string name, string city)  {        //TODO: Добавете вашия код тук …  } |

**Команда за добавяне на компания**

Вашето приложение трябва да обслужва следната команда за добавяне на компании:

* **CreateCompany <companyName> <city>** - тази команда има за цел да създаде компания с дадените име и град

**Уверете се, че всички полета имат валидни състояния!**

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за компания. Командата за това е следната:

* **PrintCompanyInfo <name> -**отпечатва информация за компания във формат:

Company **<name>** from **<city>** has the following employees:

--->Employee: **<име>**  **<фамилия>** with id: **<id>**  and salary: **<заплата>**

--->Employee: **<име>**  **<фамилия>** with id: **<id>**  and salary: **<заплата>**

…/всички други служите, в реда, в който са постъпили в компанията/

* Тази команда ще получава винаги валидни и съществуващи имена на компании. За успешна реализация трябва да реализирате ваша версия на **ToString()**метода за класа **Company.**Всички служители трябва да бъдат изпечатани по дадения формат /ToString() на всеки от тях/, в реда, в който са постъпили в компания.

**Всички компании ще имат поне един служител!**

**Подзадача 3: Логика – 40 точки**

**Kоманди**

Вашето приложение трябва да реализира следните команди:

* **CreateEmployee <employeeID> <firstName> <lastName> <age>**
* **CreateEmployee <employeeID> <firstName> <lastName> <age> <salary> -**Командите създават нов Служител. При НЕподадена заплата – сетвате дефолтната стойност – 500.
* **CreateCompany <companyName> <city> -**Командata създава новa Компания.
* **PrintCompanyInfo <companyName>**- Изпечатва се информация в посочения формат за компания с даденото име
* **PrintEmployeeInfo <ID> -**Изпечатва се информация в посочения формат за служител с даденото име
* **HireEmployee <companyName> <employeeID> -**Служител с даденото **ID**следва да бъде **добавен** към компанията със съответното име.
* **FireEmployee <companyName> <employeeID> -**Служител с даденото **ID**следва да бъде **премахнат** от компанията със съответното име.
* **IncreaseSalaries <companyName> <double amount> -**Всички служители в дадената компания получават определеното повишение към заплатите си.
* **DecreaseSalaries <companyName> <double amount> -**На служителите в дадената компания трябва да бъдат намалени заплатите. **ВАЖНО –**Ако на стойността на подаденото намаление е по-голяма от стойността на заплатата на даден служител, то следва да не намалявате заплатата му, а да продължите към следващите служители.
* **GetMostHighPaidEmployee <companyName> -**На конзолата се изпечатва информация за най-високо платения служител от компанията. **ВАЖНО –** Методът в Company.cs следва да връща обект от тип Employee.

**Подзадача 4: Бонус логика – 20 точки**

Трябва да реализирате още няколко команди:

* **GetTopThreeMostHighPaidEmployees <companyName> -**Командата изпълянява логика за извеждане на информация за 3-мата служители с най-високи заплати. Очаква се да имплементирате метод в Company.cs, който да връща списък от 3 елемента – служители, подредени по заплати в намаляващ ред.
* **CheckEmployeeIsPresent <companyName> <employeeID> -**Проверява се дали служител с даденото ID е добавен към компания със съответното име. Очаква се да имплементирате метод в Company.cs, който да връща true/false спрямо резултата от проверката.
* **GetAverageEmployeeSalary <companyName> -**Очаква се да имплементирате метод в Company.cs, който да връща стойността на средната работна заплата на всички работници.

**За безпроблемната работа на всички изброени команди от задачи 3 и 4 подзадача трябва да реализирате Employee.cs и Company.cs, по аналогичен начин на показаното по-долу:**

|  |
| --- |
| Employee.cs |
| public Employee(string id, string firstName, string lastName, int age)  {     //TODO: Добавете вашия код тук …  }    public Employee(string id, string firstName, string lastName, int age, double salary)  {     //TODO: Добавете вашия код тук …  }    public override string ToString()  {     //TODO: Добавете вашия код тук …  } |

|  |
| --- |
| Company.cs |
| public Company(string name, string city)  {  //TODO: Добавете вашия код тук …  }    public void HireEmployee(Employee emp)  {  //TODO: Добавете вашия код тук …  }    public void FireEmployee(string employeeId)  {  //TODO: Добавете вашия код тук …  }    public void IncreaseSalaries(double value)  {  //TODO: Добавете вашия код тук …  }    public void DecreaseSalaries(double value)  {  //TODO: Добавете вашия код тук …  }    public Employee GetMostHighPaidEmployee()  {  //TODO: Добавете вашия код тук …  }  public List<Employee> GetTopThreeMostHighPaidEmployees()  {  //TODO: Добавете вашия код тук …  }    public bool CheckEmployeeIsPresent(string employeeId)  {  //TODO: Добавете вашия код тук …  }    public double GetAverageEmployeeSalary()  {  //TODO: Добавете вашия код тук …  }    public override string ToString()  {  //TODO: Добавете вашия код тук …  } |

**Забележка:**Освен горепосочените методи трябва да реализирате и необходимите свойства за всеки от класовете. Възможно е да е удачно да реализирате допълнителни полета, свойства и методи, по ваша преценка.

**Подзадача 5: Валидация – 10 точки**

Освен всичко останало вие трябва да направите и валидация!

Не допускайте създаването на:

* FirstName на Employee следва да бъде низ с дължина над 2 символа и под 8. Ако не - Invalid employee name
* Age на Employee следва да бъде положително число. Ако не - Invalid employee age
* CompanyName следва да бъде низ с дължина над 2 символа. Ако не - Invalid company name
* CompanyCity следва да бъде низ с дължина над 4 символа, на който първата буква е главна. Ако не - Invalid city

При невалидни данни хвърлете грешка от тип **ArgumentException(“message”)**

**Допълнително информация свързана с валидацията**

Командите **CreateEmployee, CreateCompany**могат да съдържат информация, която да наруши правилата от по-горе. Въпреки това, имената на служителите и компаниите в тези команди **винаги** ще бъдат низове, съдържащи малки и/или големи латински букви.

На всички останали методи, подадената информация ще бъде валидна.

**Ако получите команда, която се опитва да извърши операция, която нарушава валидацията, хвърлете изключение с текст, посочен за съответната валидация!**

**Вход / Изход**

**Вход**

* Програмата ще получава множество редове с информация. Всеки ред представлява команда. Самият вход се обработва изцяло от примерния Program.cs.
* Всички команди приключват с въвеждането на End

**Изход**

За някои от командите не е нужно да извеждате нищо. За други е необходимо форматиране на изход – напр. Company.ToString(), Employee.ToString()

**Ограничения**

* Всички цели числа ще бъдат в диапазона **–10000** до **+10000**
* Всички числа с плаваща запетая ще бъдат въвеждани с до два знака след запетаята.
* Всички имена няма да съдържат интервал

**Примери**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| CreateEmployee peshoID Pesho Peshov 20  CreateEmployee goshoId Gosho Goshov 35 725  PrintEmployeeInfo peshoID  PrintEmployeeInfo goshoId  End | Employee: Pesho Peshov with id: peshoID and salary: 500.00  Employee: Gosho Goshov with id: goshoId and salary: 725.00 |
| CreateCompany itguruOOD Sofia  CreateCompany itguruOOD2 Pernik  CreateEmployee goshoId Gosho Goshov 35 725  CreateEmployee peshoID Pesho Peshov 20  HireEmployee itguruOOD goshoId  PrintCompanyInfo itguruOOD  HireEmployee itguruOOD peshoID  PrintCompanyInfo itguruOOD  End | Company itguruOOD from Sofia has the following employees:  -->Employee: Gosho Goshov with id: goshoId and salary: 725.00  Company itguruOOD from Sofia has the following employees:  -->Employee: Gosho Goshov with id: goshoId and salary: 725.00  -->Employee: Pesho Peshov with id: peshoID and salary: 500.00 |
| CreateEmployee peshoId Pesho Peshov 20 678.99  CreateEmployee goshoId Gosho Goshov 18  CreateCompany companyName Sofia  PrintEmployeeInfo peshoId  HireEmployee companyName peshoId  HireEmployee companyName goshoId  PrintCompanyInfo companyName  FireEmployee companyName goshoId  PrintCompanyInfo companyName  HireEmployee companyName goshoId  IncreaseSalaries companyName 120  PrintCompanyInfo companyName  DecreaseSalaries companyName 50  PrintCompanyInfo companyName  GetMostHighPaidEmployee companyName  CreateEmployee strahilId Strahil Peshov 20 20000  CreateEmployee mishoId Misho Mishov 15 120  HireEmployee companyName strahilId  HireEmployee companyName mishoId  GetTopThreeMostHighPaidEmployees companyName  CheckEmployeeIsPresent companyName mishoId  GetAverageEmployeeSalary companyName  CreateEmployee mishoId M NotValidFirstName 15 120  CreateEmployee mishoId NotValidAge Mishov -200  CreateCompany c NotValidCompanyName  CreateCompany NotValidCity a  End | Employee: Pesho Peshov with id: peshoId and salary: 678.99  Company companyName from Sofia has the following employees:  -->Employee: Pesho Peshov with id: peshoId and salary: 678.99  -->Employee: Gosho Goshov with id: goshoId and salary: 500.00  Company companyName from Sofia has the following employees:  -->Employee: Pesho Peshov with id: peshoId and salary: 678.99  PrintCompanyInfo companyName  Company companyName from Sofia has the following employees:  -->Employee: Pesho Peshov with id: peshoId and salary: 798.99  -->Employee: Gosho Goshov with id: goshoId and salary: 620.00  Company companyName from Sofia has the following employees:  -->Employee: Pesho Peshov with id: peshoId and salary: 748.99  -->Employee: Gosho Goshov with id: goshoId and salary: 570.00  The most high paid worker is - Employee: Pesho Peshov with id: peshoId and salary: 748.99  Top three highest paid employees at companyName:  ---#1 - Employee: Strahil Peshov with id: strahilId and salary: 20000.00  ---#2 - Employee: Pesho Peshov with id: peshoId and salary: 748.99  ---#3 - Employee: Gosho Goshov with id: goshoId and salary: 570.00  mishoId is present at companyName  The average salary at companyName is: 5359.75  Invalid employee name  Invalid employee name  Invalid company name  Invalid city |

**Точки**

Разбивката по подзадачи е следната:

1. **15** точки, като трябва да имате и задължително реализиран ToString() метод
2. **15** точки, като трябва да имате и задължително реализиран ToString() метод
3. **40** точки
4. **20** точки
5. **10** точки

Общ брой точки: **100**