**Task 1**

**Class: Agency**

Attributes:

* employeesList
* contractsList

Operations:

* addEmployee
* addContract
* getEmployeesList
* getContractsList
* getEmployeeWithLeastNumberOfContracts
* getSumOfContractsCosts

**Class: Contract**

Attribiutes:

* cost
* maxNumberOfDesigners
* designersTeam

Operations:

* getCost
* getDesignersTeam
* addDesigner

**Class: Employee**

Attributes:

* employeeName
* numberOfContracts

Operations:

* getEmployeeName
* getNumberOfContracts
* increaseNumberOfContracts

**Task 2**

**Class: Agency**

addEmployee(Employee employee): adds *Employee* object to *employeesList.*

addContract(Contract contract): adds *Contract* object to *contractsList.*

getEmployeesList(): returns *employeesList.*

getContractsList(): returns *contractsList.*

getEmployeeWithLeastNumberOfContracts(List<Employee> exceptThem): Makes a copy of actual list of employees and then removes from it employees from *exceptThem* list. This list is provided by contract, there are employees which are currently working on a contract. It prevents from returning employee that is currently working on this contract. Then it finds an employee with the least number of contracts working on by using streams and the method *getNumberOfContracts()*. If there are two employees with same number, last one from the list is returned. If list is empty, *NoSuchElementException* is thrown.

getSumOfContractsCosts(): This method iterates through every contract object in the *contractsList* and adds the cost to the sum. Cost is taken out from objects by *getCost()* method. Then sum is returned as float.

**Class: Contract**

getCost(): returns the cost of contract as float.

getDesignersTeam(): returns list of designers which are working on actual contract.

addDesigner(Agency agency): Adds a designer (employee) to *designersTeam* list. It uses a method *getEmployeeWithLeastNumberOfContracts(List<Employee> exceptThem)* providing *designersTeam* list as *exceptThem* variable, to get a designer (employee) with the least number of contracts working on. Operation is invoked only if size of *designersTeam* list is less than *maxNumberOfDesigners*. After receiving a designer, it is added to designersTeam and its *numberOfContracts* is increased by one using *increaseNumberOfContracts()* method. Method returns true if operation finish successfully, otherwise false. Also, method requires to point to the *Agency* object as I have decided to make an *Agency* as non-static class.

**Class: Employee**

getEmployeeName(): returns employee’s name as String.

getNumberOfContracts(): returns numberOfContracts as int.

increaseNumberOfContracts(): increases number of contracts by one, used by method *addDesigner(Agency agency)*.

**Task 3**

1..maxNumberOfDesigners

1

\*

1

|  |
| --- |
| Agency |
| -employeesList: List<Employee>  -contractsList: List<Contract> |
| +addEmployee(employee: Employee)  +addContract(contract: Contract)  +getEmployeesList(): List<Employee>  +getContractsList(): List<Contract>  +getEmployeeWithLeastNumberOfContracts  (exceptThem: List<Employee>): Employee  +getSumOfContractsCosts(): float |
| Exception  NoSuchElementException |

|  |
| --- |
| Employee |
| -employeeName: String  -numberOfContracts: int |
| +getEmployeeName(): String  +getNumberOfContracts(): int  +increaseNumberOfContracts() |

|  |
| --- |
| Contract |
| -cost: float  -maxNumberOfDesigners: int  -designersTeam: List<Employee> |
| +getCost(): float  +getDesignersTeam(): List<Employee>  +addDesigner(agency: Agency): boolean |
| Exception  IllegalArgumentException |

\*

\*

**Task 4**

getCost()

getDesignersTeam()

addDesigner(agency: Agency)

produceAdvert()

produceAdvert()

HasDesignersSpace

new Contract/

cost=1000;

maxNumberOfDesigners=3;

HasMaximumDesigners

getCost()

getDesignersTeam()

**Task 5**

|  |
| --- |
| designer.increaseNumberOfContracts()  j = getNumberOfContracts()  employee: Employee  n = designer: Employee  designersTeam.add(designer)  agency: Agency  (n) = getEmployeeWithLeastNumberOfContracts(designersTeam)  contract: Contract  sd addDesigner(Agency agency) |

**Task 6**

**Sets:**

DT – set of employees working on a contract identified by positive numbers 0, 1, 2 …

DT = {x|x ≥ 0 and x ≤ maxNumberOfDesigners}

**Functions:**

Contract: CostType x maxNumberOfDesignersType > ContractType

**Types:**

cost – Float: CostType

maxNumberOfDesigners – Integer: maxNumberOfDesignersType constant

contract – Contract: ContractType

**Variables:**

cost: float\_0…\*

maxNumberOfDesigners: Integer\_1…\* constant

**Invariants:**

0 < card(DT) ≤ maxNumberOfDesigners

0 ≤ cost

**Function addDesigner:**

addDesigner: card(DT) x maxNumberOfDesignersType x DT > EmployeeType