**Salesforce Developers**

**Basic Knowledge Test: 1**

**Name: Anton Kaliev**

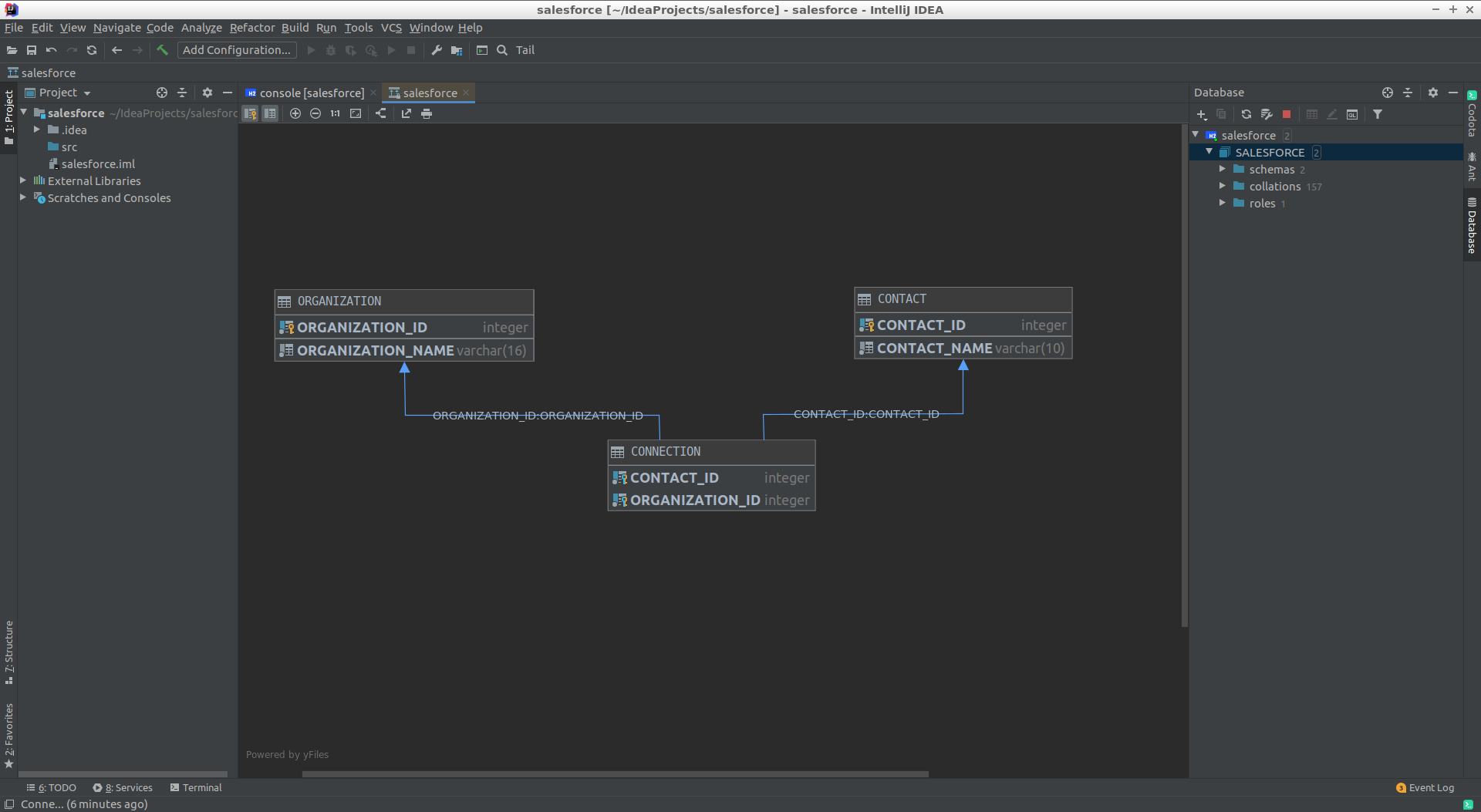
**Skype:**

**Phone:093 7 888 99 5**

**Note:** All test tasks should be completed in well-structured code/ scripts applying Java, Vanilla JS, SQL best practices and Code Conventions.

1. Your DB needs to hold two main objects – **Contacts** and **Organizations**. Each Contact can work in several organizations and each Organization has a lot of contacts.   
     
   - What type of relationship would you use?   
   - Draw the scheme of all tables with relationships.

Relationship Name:   
Many-to-many. We use a table of relationships to bring it to a one-to-many relationship, because many-to-many relationships are not possible in relational databases  
  
Schema:



create table Contacts

(

contact\_ID integer not null auto\_increment,

contact\_name varchar(10) not null,

primary key (contact\_ID)

);

create table Organization

(

organization\_ID integer not null auto\_increment,

organization\_name varchar(16) not null,

primary key (organization\_ID)

);

create table Connection

(

contact\_ID integer not null,

organization\_ID integer not null,

primary key (contact\_ID, organization\_ID),

foreign key (contact\_ID) references Contacts,

foreign key (organization\_ID) references Organization,

);

1. Now we have DB with each **Contact** has the only one related **Organization**.   
     
   - How can you find all contacts with **Name** starts with *“A”* in organization named *“Cats”*?   
   - How can you count number of contacts for each organization?   
   - Write SQL query for each question above.

1:

select Contacts.contact\_name

from Connection

inner join Contacts on Connection.contact\_ID = Contacts.contact\_ID

inner join Organization on Connection.organization\_ID = Organization.organization\_ID

where organization\_name = 'Cat'

and contact\_name like 'A%';

2:

select Organization.organization\_name, *count*(Organization.organization\_name)

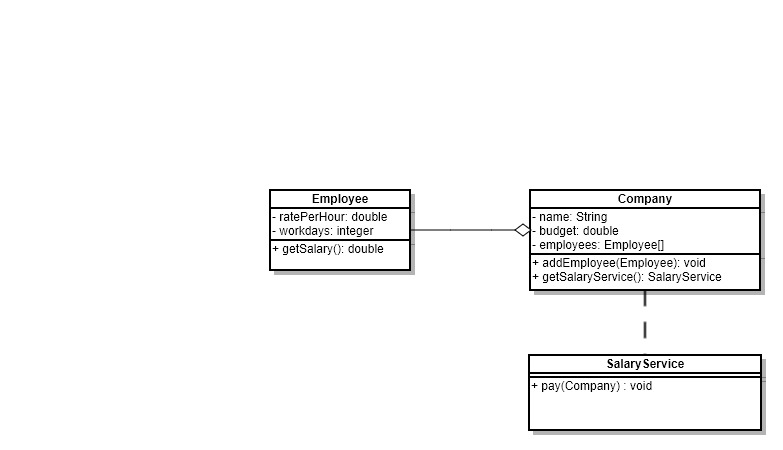
from Connection

join Organization on Connection.organization\_ID = Organization.organization\_ID

group by Organization.organization\_name;

1. As a part of project, you need to implement functionality of calculating salaries for employees in different companies. Each company have its own salary budget. Every employee has his own hourly rate and work hours. Basic formula for salary calculation is ratePerHour \* workdays \* 8

Each company should have ability to pay salaries to its workers. We should take into consideration that total salary payments can’t be greater than company wage budget. UML diagram listed below.



1) Write classes that implement this diagram.   
2) You should have at least 3 types of employees  
e.g.

Manager

Developer

Scrum master

Developer may have bonuses that should be add to the basic salary, it should be taken into consideration during calculation of his wage.

package Main;

import Main.Employees.Developer;

import Main.Employees.Employee;

import lombok.Getter;

import lombok.NonNull;

import lombok.Setter;

import java.util.ArrayList;

import java.util.List;

import static Main.Main.*ANSI\_RED*;

import static Main.Main.*ANSI\_RESET*;

public class Company {

@Getter

private String name;

@Setter

private double budget;

private List<Employee> employees;

public Company(String name, double budget) {

this.name = name;

this.budget = budget;

employees = new ArrayList<>();

}

*/\*\**

*\* Try to add new employee to company.*

*\* Throw exception if employee already exist in company,*

*\* or company's free budget less than current employee salary.*

*\**

*\** ***@param*** *employee to add, cannot be a null value*

*\*/*

public void addEmployee(@NonNull Employee employee) throws IllegalArgumentException {

try {

//may be current employee already exist?

if (employees.stream().anyMatch(x -> x.getName().equals(employee.getName())))

throw new IllegalArgumentException(String.*format*(

"current employer, %s, already exist\n", employee.getName()));

//do we have enough budget?

double totalSalary = calcTotalSalary();

totalSalary += employee.getSalary();

if (budget < totalSalary)

throw new IllegalArgumentException(String.*format*(

"%s company have only %.2f available budget, but %s need %.2f\n",

this.getName(), this.availableBudget(), employee.getName(), employee.getSalary()));

//if all tests pass, add it

employees.add(employee);

System.*out*.printf("company %s hire %s and still have %.2f available budget\n",

this.getName(), employee.getName(), this.availableBudget());

} catch (IllegalArgumentException e) {

System.*out*.println(*ANSI\_RED* + e.getMessage() + *ANSI\_RESET*);

}

}

*/\*\**

*\* Give bonuses to employee*

*\**

*\** ***@param*** *employee*

*\*/*

public void giveBonus(Employee employee, double bonus) {

try {

synchronized (employee) {

if (!(employee instanceof Developer))

throw new IllegalArgumentException(String.*format*("The current employee %s is not in the bonus program\n", employee.getName()));

if (this.availableBudget() < bonus)

throw new IllegalArgumentException(String.*format*("The current employee %s bonus %.2f more then available budget(%.2f)\n",

employee.getName(), bonus, this.availableBudget()));

((Developer) employee).setBonuses(bonus);

System.*out*.printf("employee %s have %.2f bonus\n", employee.getName(), bonus);

}

} catch (IllegalArgumentException e) {

System.*out*.println(*ANSI\_RED* + e.getMessage() + *ANSI\_RESET*);

}

}

*/\*\**

*\* calculate total available salary for current company*

*\** ***@return***

*\*/*

private double calcTotalSalary() {

double totalSalary = 0;

for (Employee employee : employees)

totalSalary += employee.getSalary();

return totalSalary;

}

public boolean checksAbilityToPay() {

return budget > calcTotalSalary();

}

private double availableBudget() {

return budget - calcTotalSalary();

}

}

package Main;

import static Main.Main.\*;

public class SalaryService {

private Company company;

private SalaryService(Company company) {

this.company = company;

}

public static SalaryService builder(Company company) {

return new SalaryService(company);

}

public static void pay(Company company) throws Exception {

if (!company.checksAbilityToPay())

throw new Exception(String.*format*("company %s inability to pay salary", company.getName()));

System.*out*.printf(*ANSI\_GREEN* + "company %s salary paid successfully\n" + *ANSI\_RESET*, company.getName());

}

}

package Main.Employees;

import lombok.Getter;

import lombok.Setter;

public abstract class Employee {

@Getter

protected double ratePerHour;

@Getter

protected int workdays;

@Getter

protected String name;

// Now it is not used, but will protect against collisions.

// @Getter

// @Setter

// protected int ID;

public double getSalary() {

return ratePerHour \* workdays \* 8;

}

}

package Main.Employees;

public class Manager extends Employee {

public Manager(String name, int rate, int workdays) {

this.name = name;

this.ratePerHour = rate;

this.workdays = workdays;

}

}

package Main.Employees;

import lombok.Getter;

import lombok.Setter;

public class Developer extends Employee {

@Getter

@Setter

private double bonuses;

public Developer(String name, int rate, int workdays) {

this.name = name;

this.ratePerHour = rate;

this.workdays = workdays;

}

@Override

public double getSalary() {

return ratePerHour \* workdays \* 8 + bonuses;

}

}

package Main.Employees;

public class ScrumMaster extends Employee {

public ScrumMaster(String name, int rate, int workdays) {

this.name = name;

this.ratePerHour = rate;

this.workdays = workdays;

}

}

1. You have a list of numbers to fill the table of prices in the shown way

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Jan** | **Feb** | **Mar** | **Apr** | **May** | **Jun** | **Jul** | **Aug** | **Sep** | **Oct** | **Nov** | **Dec** |
| Prod. 1 | Num1 | Num2 | Num3 | Num4 | Num5 | Num6 | Num7 | Num8 | Num9 | Num10 | Num11 | Num12 |
| Prod. 2 | Num13 | Num14 | … |  |  |  |  |  |  |  |  |  |
| Prod. 3 |  |  |  |  |  |  |  |  |  |  |  |  |

- Implement Java method which receive this list and return the object which will be convenient to find price using Month and Product keys.

package Main;

import java.time.Month;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

public class ProductList {

//variables

private List<Product> list = new <Product>ArrayList();

private class Product {

String productName;

Map<Month, Double> price = new <Month, Double>LinkedHashMap();

public Product(String productName, Month month, double value) {

this.productName = productName;

this.price = new LinkedHashMap<>();

this.price.put(month, value);

}

public double getPrice(Month month) {

return price.get(month);

}

public void setPrice(Month month, double value) {

price.put(month, value);

}

}

*/\*\**

*\* add Product to the ProductList, if it not already there, or set the price of the currently added product*

*\**

*\** ***@param*** *productName product Name*

*\** ***@param*** *month month*

*\** ***@param*** *price price*

*\*/*

public void addValue(String productName, Month month, double price) {

Product product = list.stream().filter(x -> x.productName.equals(productName)).findAny().orElse(null);

if (null == product) {

product = new Product(productName, month, price);

list.add(product);

return;

}

product.setPrice(month, price);

}

*/\*\**

*\* get Price value for selected product.It will throw exceptions if something wrong*

*\**

*\** ***@param*** *productName product Name*

*\** ***@param*** *month month*

*\** ***@return*** *price of the product*

*\** ***@throws*** *NullPointerException if there is no expected product. or its value for a given month is not set*

*\*/*

public double getValue(String productName, Month month) throws NullPointerException {

Product product = list.stream().filter(x -> x.productName.equals(productName)).findAny().orElse(null);

if (null == product)

throw new NullPointerException(String.*format*("no %s product in current list", productName));

try {

return product.getPrice(month);

} catch (NullPointerException e) {

// e.printStackTrace();

throw new NullPointerException(String.*format*("no price for %s product at %s", productName, month));

}

}

public static void main(String[] args) {

ProductList productList = new ProductList();

// ProductList.Product product = productList.new Product("beer", Month.NOVEMBER, 20.2);

productList.addValue("beer", Month.*NOVEMBER*, 20.2);

System.*out*.println(productList.getValue("beer", Month.*NOVEMBER*));

try {

System.*out*.println(productList.getValue("beer", Month.*DECEMBER*));

} catch (NullPointerException e) {

System.*out*.println(e.getMessage());

}

}

}

1. Given todo list in file ‘Test\_Backend\_4’.



Implement following tasks using javaSript:

* 1. creation of the toDo list item.

<div class="todo">

<div class="todo-canvas">

<div class="todo-header">

<button class="todo-button todo-button-add" id="todo-button-add">+</button>

</div>

<div class="todo-list">

<div class="todo-card" id="card-1">

<label class="label">Content</label>

<input class="content" type="text" id="task-value"/>

</div>

<div id="task-card">

<input class="content" type="text" readonly value="111111111"/>

<input class="content" type="text" readonly value="222222222"/>

<input class="content" type="text" readonly value="333333333"/>

<input class="content" type="text" readonly value="444444444"/>

<input class="content" type="text" value="555555555"/>

</div>

</div>

</div>

</div>

<script>

var ***addButton*** = ***document***.getElementById("todo-button-add");

var ***taskValue*** = ***document***.getElementById("task-value");

***addButton***.addEventListener("click", addTask);

***taskValue***.addEventListener("keypress", function (ev) {

if (ev.key === 'Enter') {

addTask();

}

});

//add task

function addTask() {

var task = ***document***.createElement("input");

task.className = "content";

task.setAttribute("type", "text");

task.readOnly = true;

task.value = ***taskValue***.value;

if (***taskValue***.value === '') {

alert("no task text");

} else {

***document***.getElementById("task-card").append(task);

}

***taskValue***.value = "";

}

</script>

1. Why have you chosen to be developer? Describe your main objectives and hopes about your future carrier.