Oczekiwania od firmy:

1. dobra znajomość języka Java(17);
2. doświadczenie z zakresu frameworka Spring, Spring Boot;
3. znajomość Hibernate, JPA;
4. doświadczenie w pracy z relacyjną bazą SQL;
5. doświadczenie z narzędziami Maven, Git;
6. podstawowa znajomość Dockera

Spis treści

Zagadnienia na taką rozmowę rekrutacyjną

1. Java

JVM, heap, old, young

Co to jest JVM (Java Virtual Machine)?
Old i Young generation oraz struktura pamięci JVM'a - mieć pojęcie, że takie coś jest, nie trzeba dokładnie wiedzieć jak wygląda
Link do opracowania

Jak działa hashmapa w Javie?

Link do opracowania

Kolekcja Java jakie znasz

Kolekcje

map vs flatmap, peek

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class MapVsFlatMapDemo {
    public static void main(String[] args) {
        List<String> words1 = Arrays.asList("Hello", "World");
        List<String> words2 = Arrays.asList("Java", "Stream", "API");

        // Using map to concatenate each word with "!" and return a Stream<String[]>
        Stream<String[]> mapResult = words1.stream()
                .map(word -> word.split(""))
                .peek(array -> System.out.println("Map: " + Arrays.toString(array)));

        // Using flatMap to concatenate each word with "!" and return a Stream<String>
        Stream<String> flatMapResult = words2.stream()
                .flatMap(word -> Arrays.stream(word.split("")))
                .peek(character -> System.out.println("FlatMap: " + character));

        // Collect the results into a List
        List<String[]> mapList = mapResult.collect(Collectors.toList());
        List<String> flatMapList = flatMapResult.collect(Collectors.toList());
    }
}
```

Wyjątki – Exception i Error, checked i unchecked

https://www.baeldung.com/java-catch-throwable-bad-practice

https://kobietydokodu.pl/niezbednik-juniora-wyjatki-i-ich-obsluga/

Zademonstruj jak działa Sealed class w Javie 17

Zademonstruj jak działa Pattern Matching w Javie 17

How Pattern Matching for instanceof works in Java 17:

```
public class PatternMatchingExample {
    public static void main(String[] args) {
        Object object = "Hello, World";

        if (object instanceof String str) {
            // Pattern matching allows direct assignment of the casted value
            System.out.println("Length of the string: " + str.length());

            // You can use 'str' as the casted variable in this block
            // without an additional explicit cast
        } else {
            System.out.println("Not a string");
        }

        // Now, 'str' is not in scope here
        // If 'object' were not a String, the 'else' block would execute
    }
}
```

In this example:

We have an object of type Object.
We use Pattern Matching for instanceof with the variable str to check if object is an instance of String. If it is, the casted value is assigned to str, and we can use it directly within the if block without additional casting.
If object is not an instance of String, the else block is executed.
Note that the scope of the str variable is limited to the if block, so it's not accessible outside of that block.
Pattern Matching for instanceof simplifies code by eliminating the need for explicit type casting and provides more concise and readable code, especially when working with complex conditional logic.

2. Dobre praktyki, wielowątkowość, testy

Singleton

Niemutowalność (Immutability) - czym jest i dlaczego warto stosować?
Link do opracowania

zasady SOLID na pamięć, każda z literek! w teorii i praktyce

Wielowątkowość (Atomic, Pula wątków, metoda .join(), kolekcje Concurrent Collections, volatile)

Testy (zasada F.I.R.S.T)

3. Spring

Aspekty

REST w Springu GET POST DELETE - REST API Basics

Jak można stworzyć beana? 3 sposoby i kontekst Springa

Wstrzykiwanie zależności Dependency Injection przez konstruktor

4. JPA

czym jest JPA i jakie adnotacje znasz?
"@Entity, @Table, @Column, @Id
Link

Jak ignorować mapowanie wybranych pól klasy?

Ignorowanie robimy przez użycie: @Transient
Link

5. SQL

Zadanie praktyczne: LEFT JOIN

```
-- Create a table for Departments
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50)
);

-- Insert sample data into the Departments table
INSERT INTO Departments (DepartmentID, DepartmentName)
VALUES
    (1, 'HR'),
    (2, 'IT'),
    (3, 'Finance');

-- Create a table for Employees
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);

-- Insert sample data into the Employees table
INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID)
VALUES
    (1, 'John', 'Doe', 1),
    (2, 'Jane', 'Smith', 2),
    (3, 'Bob', 'Johnson', NULL),
    (4, 'Alice', 'Brown', 1),
    (5, 'Eve', 'Davis', 2);
```

Write a LEFT JOIN query to retrieve information about employees and their corresponding departments, taking advantage of the foreign key relationship:

Odpowiedź:
The result will include all employees, even if they don't have a department assigned (thanks to the LEFT JOIN), and it will display the department name if it exists for the employee.

```
-- Retrieve a list of all employees and their corresponding department names using a LEFT JOIN
SELECT
    E.EmployeeID,
    E.FirstName,
    E.LastName,
    D.DepartmentName
FROM
    Employees E
LEFT JOIN
    Departments D
ON
    E.DepartmentID = D.DepartmentID;
```

DDL, DQL, DML, TCL

Indeksy

Transakcje - izolacje

Transakcje - propagacja

Transakcje - podstawy

Joiny

To też

6. Mikroserwisy i HTTP

Zalety i wady Mikroserwisów

Link do opracowania

HTTP jest bezstanowy - co to znaczy?

GET, Post, put, delete - jakie znasz metody i jak to zakodzić w Springu?

Link do opracowania

Docker vs Virtual Machine

https://aws.amazon.com/compare/the-difference-between-docker-vm/

https://www.youtube.com/watch?v=TvnZTi_gaNc