```java
import java.util.HashMap;
import java.util.Map; // Карта

/*
 пример реализации классов Department и Employee с использованием поля
 positions в классе Department для установления связи между отделом и
сотрудниками
 */

// Класс отдела
class Department {
    private String name;
    private Map<String, Double> positions; // Карта должностей и ставок
    private Map<String, Employee> employees; // Карта сотрудников по имени (class
Employee)

    public Department(String name) {
        this.name = name;
        this.positions = new HashMap<>(); // Инициализация карты должностей и
ставок
        this.employees = new HashMap<>(); // Инициализация карты сотрудников
    }

    // Метод для добавления должности и ставки
    public void addPosition(String position, double salary) {
        positions.put(position, salary);
    }

    // Метод для добавления сотрудника
    public void addEmployee(String name, String position) {
        employees.put(name, new Employee(name, position));
    }

    // Метод для поиска сотрудника по имени
    /* true, если сотрудник найден, или false, если не найден */
    public boolean hasEmployee(String name) {
        return employees.containsKey(name); // Проверка наличия сотрудника по
имени
    }

    // Метод для получения ставки по должности
    public double getSalaryForPosition(String position) {
        return positions.getOrDefault(position, 0.0); // Возвращает ставку или
0.0, если должность не найдена
    }

    // Метод для вывода всех сотрудников отдела
    public void printEmployees() {
        System.out.println("Employees in " + name + ":");
        for (Map.Entry<String, Employee> entry : employees.entrySet()) {
            Employee employee = entry.getValue();
```

```
            System.out.println(employee.getName() + " - " +
employee.getPosition());
        }
    }

    // Метод для поиска сотрудников по должности
    /* true, если должности найдена, или false, если не найдена */
    public void searchByPosition(String position) {
        System.out.println("Employees with position " + position + " in " + name
+ ":");
        for (Map.Entry<String, Employee> entry : employees.entrySet()) {
            Employee employee = entry.getValue();
            if (employee.getPosition().equals(position)) {
                System.out.println(employee.getName() + " - " +
employee.getPosition());
            }
        }
    }
}

// Класс сотрудника
class Employee {
    // Класс Employee хранит информацию
    // о имени сотрудника и его должности
    private String name;
    private String position;

    public Employee(String name, String position) {
        this.name = name;
        this.position = position;
    }

    public String getName() {
        return name;
    }

    public String getPosition() {
        return position;
    }
}

class Main {
    public static void main(String[] args) {
        /*
         * Для каждого сотрудника получается ставка в соответствии с его
должностью в
         * отделе, и выводится информация о сотруднике и его зарплате
         */

        // Создание отдела IT
        Department department = new Department("IT Department");
```

```
        department.addPosition("Developer", 50000.0); // Добавление должности и
ставки
        department.addPosition("Manager", 60000.0);

        // Добавление сотрудников (объявление переменной employees в классе
Department)
        department.addEmployee("Alice", "Developer");
        department.addEmployee("Bob", "Manager");
        department.addEmployee("Bob1", "Manager");
        department.addEmployee("Bob2", "Manager");

        // Вывод всех сотрудников
        department.printEmployees();

        // Поиск сотрудника по имени и проверка наличия данных
        String searchName = "Alice";
        if (department.hasEmployee(searchName)) {
            System.out.println(searchName + " is found in the department.");
        } else {
            System.out.println(searchName + " is not found in the department.");
        }

        // Поиск сотрудников по должности
        department.searchByPosition("Manager");
        department.searchByPosition("Manager2");
    }
}
```

PS C:\javasing\l6\l6> c:; cd 'c:\javasing\l6\l6'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:56101' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\USER\AppData\Roaming\Code\User\workspaceStorage\744c66603949d31d97d6532cc2d75fe6\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'Main'

Employees in IT Department:

Bob2 - Manager

Bob1 - Manager

Bob - Manager

Alice - Developer

Alice is found in the department.

Employees with position Manager in IT Department:

Bob2 - Manager

Bob1 - Manager

Bob - Manager

Employees with position Manager2 in IT Department:

PS C:\javasing\l6\l6>

```
PS C:\javasing\l6\l6>  c:; cd 'c:\javasing\l6\l6'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-agentlib:jdwp=transport=dt_so
cket,server=n,suspend=y,address=localhost:56101' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\US
ER\AppData\Roaming\Code\User\workspaceStorage\744c66603949d31d97d6532cc2d75fe6\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'Main
'
Employees in IT Department:
Bob2 - Manager
Bob1 - Manager
Bob - Manager
Alice - Developer
Alice is found in the department.
Employees with position Manager in IT Department:
Bob2 - Manager
Bob1 - Manager
Bob - Manager
Employees with position Manager2 in IT Department:
PS C:\javasing\l6\l6>
```