**Task-6-1**

**Программа учета сотрудников.**

**Цели задания:**

1. Познакомиться с ООП
2. Получить первые знания в написании классов и создании объектов

**Описание:**

Цель текущей задачи – познакомиться с Объектно-ориентированным программированием (ООП) и написать свой первый Java класс. Вторая цель – понять в первом приближении, что такое инкапсуляция в ООП.

Суть задания состоит в том, что вам необходимо спроектировать класс «Сотрудник» – **Employee**.

Класс **Employee** должен содержать следующие поля:

1. firstName – имя – строка.
2. lastName – фамилия – строка.
3. position – должность – строка.
4. salary – зарплата – double.

Класс Employee должен содержать следующие методы:

1. Конструктор с параметром **firstName** – позволяет создать сотрудника только с именем.
2. Конструктор с параметрами **firstName** и **lastName** – позволяет создать сотрудника с именем и фамилией.
3. **void setPosition(String p)** – позволяет установить должность сотрудника. В качестве видов должностей можете использовать следующие примеры:
   1. director
   2. programmer
   3. qa
4. **void setSalary(double s)** – позволяет установить зарплату сотрудника.
5. **String getFullName()** – возвращает «полное» имя сотрудника. Полное имя – это:
   1. **Имя + Фамилия -** если фамилия сотрудника присутствует
   2. **Только имя** – если фамилия не была введена (см. пункты 1 и 2)
6. **String getPosition()** – возвращает должность сотрудника.
7. **double getSalary()** – возвращает заработную плату сотрудника.
8. **String getEmployeeInfo() –** возвращает полную информацию о сотруднике в формате **«полное имя – должность - зарплата»**

Для тестирования класса Employee создайте отдельный класс с методом main(). В методе main() объявите массив сотрудников. Так как мы работаем пока только со статическими массивами – заранее определите размер массива. Например, 5 сотрудников. И реализуйте следующие операции в методе main():

1. Добавить нового сотрудника, у которого есть только имя.
2. Добавить нового сотрудника, у которого есть имя и фамилия.
3. Распечатать полную информацию обо всех сотрудниках.
4. Распечатать полную информацию о директоре.
5. Распечатать полную информацию обо всех сотрудниках, кроме директора.
6. Распечатать информацию о сотруднике, у которого самая маленькая заработная плата.
7. Распечатать информацию о сотруднике, у которого самая большая заработная плата.

**Внимание, дополнительные условия на добавление сотрудников:**

1. Нельзя добавлять сотрудников, если уже заполнен весь массив. В этом случае нужно вывести сообщение, что добавление сотрудника невозможно.

2. Нельзя добавить директора, если он уже был добавлен! То есть сотрудник с позицией «director» в списке должен быть только один!

После написания и проверки этой программы, вы можете восхититься, как круто мы инкапсулировали данные о сотруднике:

1. Никто не может изменить имя и фамилию сотрудника – потому что это не входит в список задач того офиса, где трудится человек.
2. Метод **getFullName()** выдает ту информацию, которая интересует руководство компании. Если формат информации потребуется изменить, то программисты будут править единственное место во всем проекте – метод **getFullName()**.

Это действительно круто!

Но сразу после этого, необходимо задуматься – а что мы еще не инкапсулировали? Какие проблемы еще остались в нашей программе? И какой новый класс (или классы) помогут нам усилить инкапсуляцию и избежать текущих проблем?