**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

**Отчет по лабораторной работе №1**

по дисциплине «Информационные технологии и программирование» на тему:

Освоение принципов ООП в Java

Выполнил:

студент группы БВТ2302

Боданюк Алексей Павлович

Проверил:

Харрасов К. Р.

Москва 2024

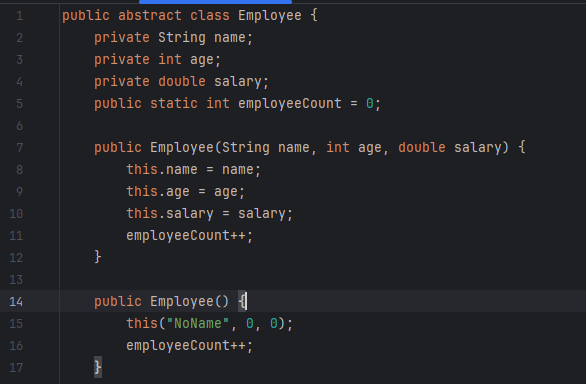
Цель работы: изучить теорию и выполнить задания, основанные на принципах работы с ООП.

Задание: Создайте иерархию классов в соответствии с вариантом. Ваша иерархия должна содержать: - абстрактный класс - 2 уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведение объекта) - демонстрацию реализации всех принципов ООП (абстракция, модификаторы доступа, перегрузка, переопределение) - наличие конструкторов (в том числе по умолчанию) - наличие геттеров и сеттеров - ввод/вывод информации о создаваемых объектах - предусмотрите в одном из классов создание счетчика созданных объектов с использованием статической переменной, продемонстрируйте работу

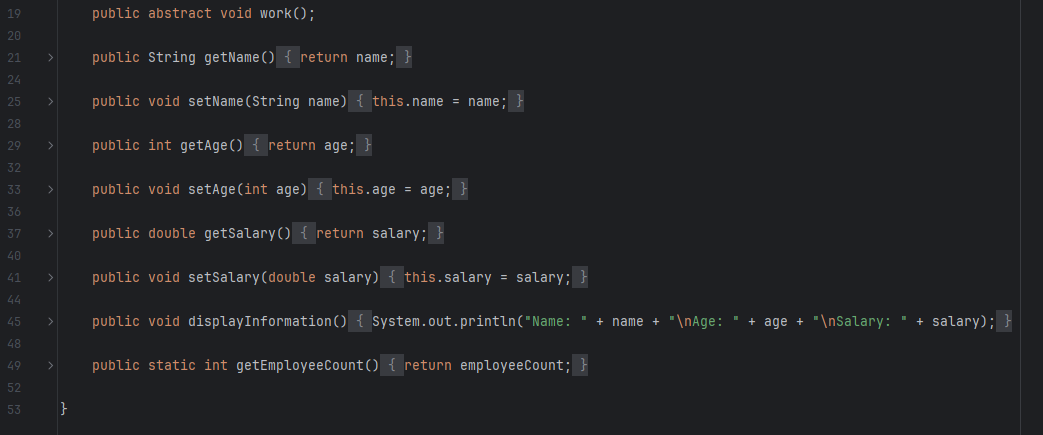
Ход работы:  
Я второй по списку, следовательно выполнять буду второй вариант. Его схема такая:

Базовый класс: Сотрудник Дочерние классы: Администратор, Программист, Менеджер

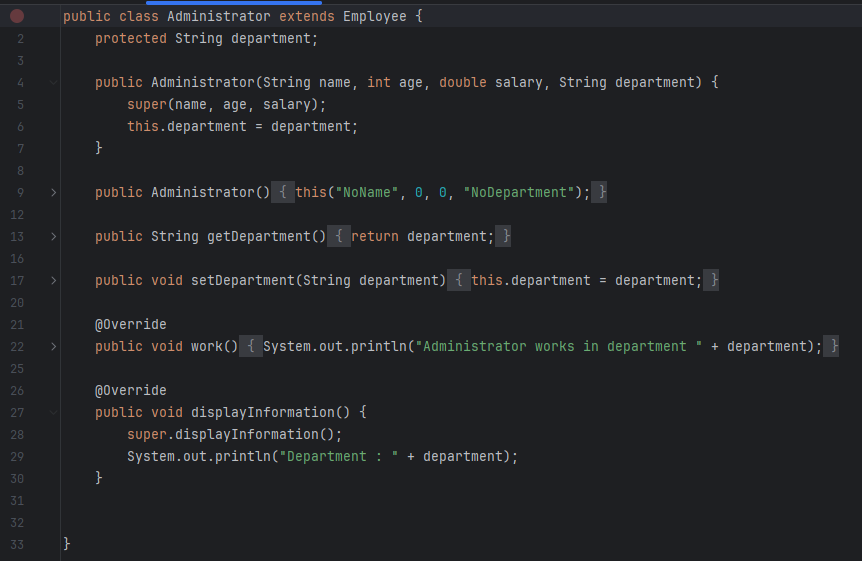
Так как требуется 2 уровня вложенности, в моей схеме класс Manager будет наследовать от класса Administrator.

Для начала создадим абстрактный класс и добавим в него 2 конструктора. На их примере как раз и отображается использование перегрузки:  


Далее в нем объявим поля для объектов, зададим геттеры и сеттеры, и 2 метода, которые будут использоваться в дочерних классах:

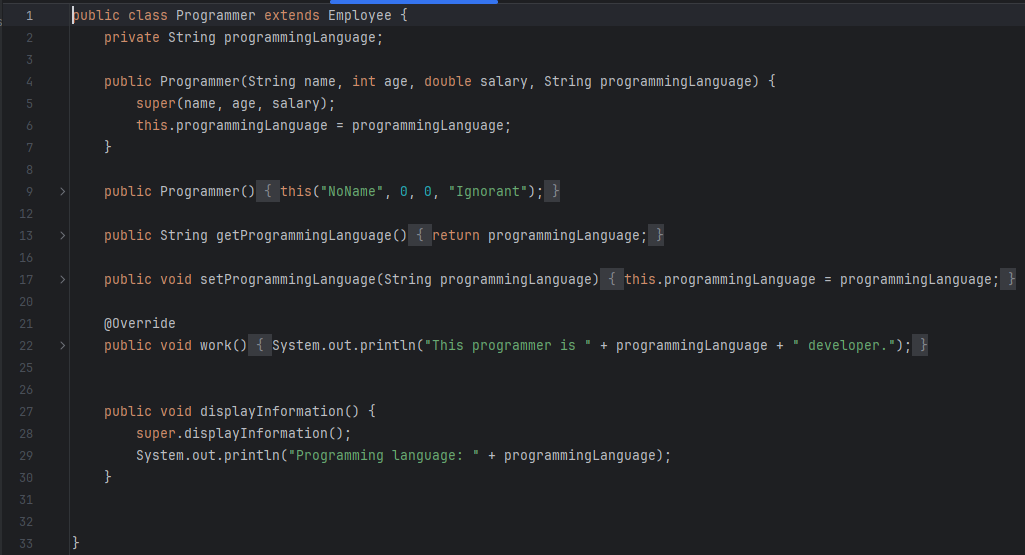


Перейдем к написанию класса Administrator:

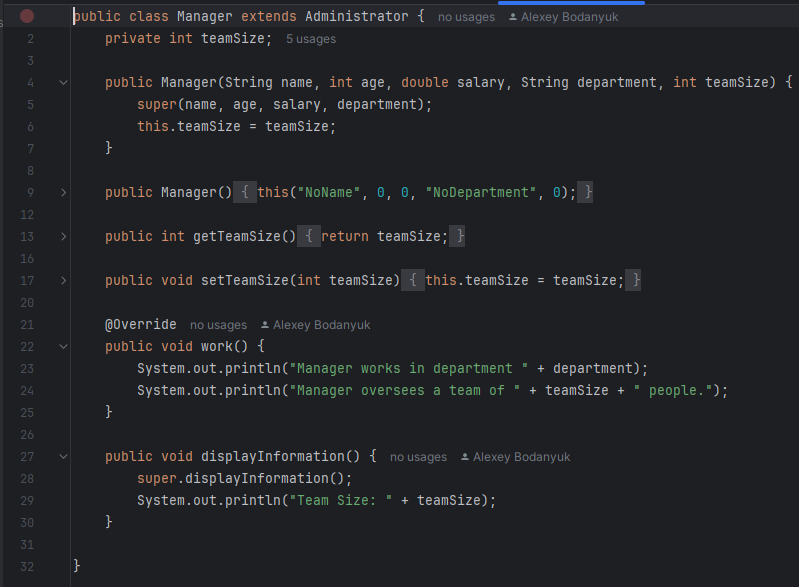


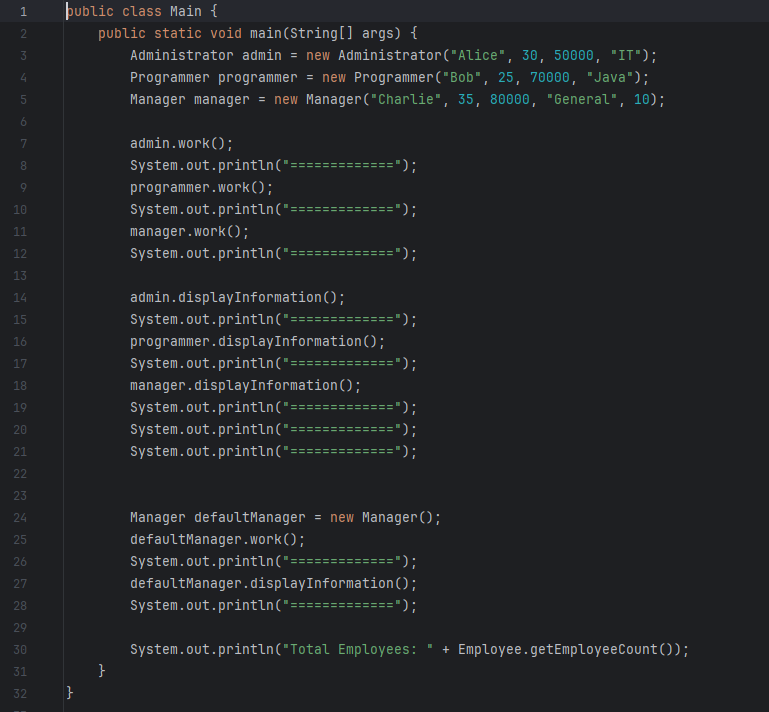
В нем мы добавляем ещё одно поле, помимо возраста, имени и зарплаты. Добавляем для него геттеры и сеттеры, а также переписываем поведение двух методов, которые служат для выдачи информации об экземпляре.

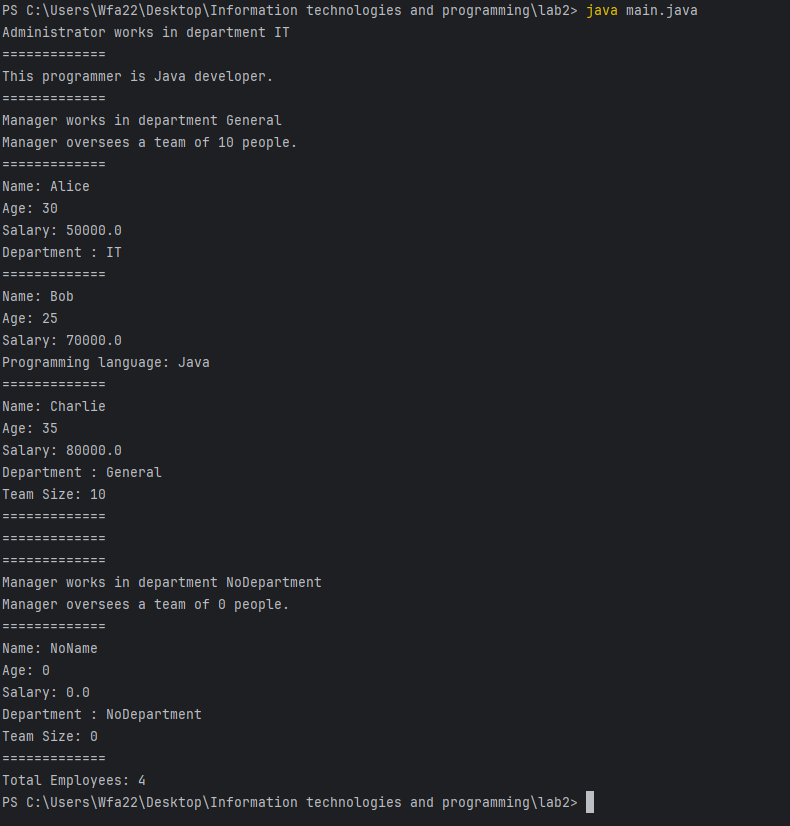
Аналогичные действия совершаем для класса Programmer:



Остался класс Manager. Для него сделаем почти все то же самое, за исключением того, что у него будет 5 полей, а не 4, как у двух прошлых классов:



Создадим файл Main.java, в котором мы создадим экземпляры классов и посмотрим на корректность написанного кода. Также продемонстрируем полное выполнение всех требований в задании:  




Вывод: создали иерархию классов, в которой реализовали абстрактный класс, 2 уровня вложенности, демонстрацию реализации всех принципов ООП (абстракция, модификаторы доступа, перегрузка, переопределение), наличие конструкторов, геттеров и сеттеров, вывода информации об объектах и наличие счетчика.