# Программирование на Python

## Домашнее задание №2.

Задача состоит в создании классов «Инженер» и «Менеджер», которые будут наследоваться от класса «Сотрудник».

1. Создайте класс Employee, который будет описывать любого сотрудника компании.
2. В качестве начальных атрибутов класса задайте позиционные параметры «name» - имя, «surname» - фамилия, «position» - должность, «age» - возраст, и именованные параметры «salary» - зарплата, по умолчанию равный 50000, и «currency» - валюта, по умолчанию равный «rub». Атрибуты salary, age и currency сделайте локальными (private), для обращения к ним пропишите геттеры (@property), при необходимости можно также добавить сеттеры.
3. Определите в классе статический метод convert\_currency(), который принимает в качестве параметров денежную сумму, текущую валюту и требуемую валюту и возвращает значение денежной суммы в требуемой валюте. Для пересчета валюты определите внутри метода словарь exchange\_rate, где ключом будет являться валюта, а значением – коэффициент перевода. В качестве основной валюты (которая будет иметь значение коэффициента перевода = 1) выберите «rub» и определите еще несколько других валют, например {‘rub’: 1, ‘usd’: 90, ‘eur’:100}.
4. Определите в классе метод change\_currency(), который будет менять валюту, в которой сотрудник получает зарплату. Метод должен работать с использованием другого метода – convert\_currency(). В случае отсутствия курса новой валюты в словаре exchange\_rate, выводите в консоль предупреждение о невозможности изменения валюты и не меняйте атрибуты объекта. В противном случае, измените зарплату и валюту объекта на новые значения.
5. Определите приватный (private) атрибут класса AGE\_LIMITS, равный 75, для ограничения верхней границы возраста и защищенный (protected) метод класса is\_meeting\_standards(), который будет возвращать True, если сотрудник младше верхней границы, и False в противном случае. Добавьте в инициализатор класса (\_\_init\_\_) проверку параметра age и в случае, если возраст сотрудника выходит за пределы диапазона, выведите в консоль предупреждение о том, что возраст сотрудника не удовлетворяет корпоративным стандартам, но объект создайте.
6. Добавьте проверку соответствия типов данных и значений для атрибутов age и salary: age должен быть целочисленным значением в диапазоне от 18 до 120, а salary – любым неотрицательным числом. Проверка должна производиться как при создании объекта класса, так и при попытке изменить атрибуты уже созданного объекта. При попытке создания нового объекта с некорректными данными объект создаваться не должен, а при изменении атрибутов существующего объекта на некорректные значения изменение не должно произойти. В обоих случаях выводите в консоль сообщение о некорректных параметрах возраста или зарплаты.
7. Добавьте магический (dunder) метод, который определит возвращаемое значение при преобразовании объекта в строку, чтобы можно было пользоваться функцией print(<объект класса>) и видеть только имя и фамилию объекта.
8. Добавьте магический метод, который позволит обратиться к объекту класса как к функции, выводя при обращении в консоль «Сотрудник <имя сотрудника> уже бежит к вам!»
9. Добавьте магический метод, который позволит сравнивать объекты одного класса. Объекты будут считаться равными, если их имя, фамилия, возраст и должность равны.
10. Создайте классы Engineer и Manager, унаследованные от класса Employee.
11. В классе Engineer переопределите атрибут базового класса AGE\_LIMITS на 65.
12. В классе Manager переопределите значение по умолчанию параметра salary на 70000.
13. В классе Manager определите метод change\_salary(), добавив проверку на процент увеличения зарплаты. Если зарплата увеличилась более чем на 50%, нужно выводить в консоль предупреждение о подозрении на коррупционные схемы.
14. Добавьте в класс Engineer атрибут manager – ссылка на объект класса Manager.
15. Добавьте в класс Manager атрибут engineers – список объектов класса Engineer.
16. Добавьте в класс Manager метод add\_engineer(), и атрибут engineers (список). Метод будет принимать на вход ссылку на объект Engineer и добавлять ее в engineers, если данного объекта еще нет в списке.
17. Добавьте в класс Engineer атрибут manager, в котором будет храниться ссылка на объект Manager, к которому приписан инженер. Изначально атрибут равен None. Ссылка должна прописываться при вызове метода add\_engineer() объекта класса Manager.
18. Определите метод give\_premium() в классах Engineer и Manager. Метод должен выводить в консоль сообщение типа <position> <name> <surname> получил премию в размере <размер премии> и возвращать размер этой премии. Размер премии для объекта класса Engineer должен равняться 10% от его зарплаты, а для объекта класса Manager – 8% от суммы всех зарплат инженеров, находящихся в его подчинении.
19. Создайте 5 объектов класса Engineer и 2 объекта класса Manager и распределите инженеров по менеджерам. Выведите в консоль атрибут engineers объектов класса Manager и раздайте всем премии.

# Пример работы:

