**Homework\_class 7, 8**

**Start 10.18.2023**

**Deadline 10.26.2023**

**!!! c 3 по 6 включительно!!! Extra по желанию**

3 задачу закончить и тоже прислать как домашку

Для каждой задачи создать тестовый класс с методом main и выполнить методы которые реализовывали. Первое что буду учитывать это то что метод main запустился и методы выполнили свою функциональность. Учитываем крайние кейсы допустим когда массив переполнен и мы не можем добавить новую сущность. (Если будут какие то несогласованные моменты в задачах пишите в чат и будем корректировать)

1) Добавление записей и поиск в телефонной книге:  
 Создайте класс PhoneBook на Java, который позволит пользователям добавлять и искать class Record в телефонной книге.  
  
 Реализуйте следующие функции:  
 1. Добавление нового контакта с полями, такими как name, phoneNumber и email.  
 2. Поиск контакта по имени и отображение (sout) его phoneNumber и email, если он найден.  
 3. Предоставление возможности отображать (sout) все контакты в телефонной книге.  
 4. Используйте array для хранения записей в телефонной книге.  
  
2) Создайте два класса: Car и CarFactory, чтобы моделировать процесс создания автомобилей на автомобильном заводе.  
  
 Car (Класс Автомобиля):  
  
 1. У класса Car должны быть следующие атрибуты: brand, model, year.  
 2. Реализуйте конструктор, который принимает параметры для установки значений атрибутов при создании экземпляра.  
 3. Создайте методы start(), который выводит сообщение о том, что автомобиль начал движение,  
 и stop(), который выводит сообщение о том, что автомобиль остановился.  
 4. Реализуйте метод getDetails(), который выводит информацию о марке, модели, годе выпуска.  
  
 CarFactory (Класс Автозавода):  
  
 1. Класс CarFactory представляет автомобильный завод и ответственен за производство автомобилей.  
 2. Создайте метод manufactureCar(), который создает новый экземпляр класса Car.  
 Метод должен принимать параметры для установки марки, модели, года выпуска создаваемого автомобиля.  
 3. Реализуйте метод listCars(), который отображает информацию о всех созданных автомобилях,  
 включая их марки, модели и годы выпуска.  
 4. В автозаводе может быть ограничение на количество созданных автомобилей 50 шт,  
 учтите это при разработке метода manufactureCar().  
  
3) Создайте два класса: Product и Store, чтобы моделировать управление продуктами в магазине.  
  
 Product (Класс Продукта):  
  
 1. У класса Product должны быть следующие атрибуты: name (название), price (цена),  
 quantity in stock (количество на складе), и description (описание).  
 2. Реализуйте конструктор, который принимает параметры для установки значений атрибутов при создании экземпляра.  
 3. Создайте методы sell(int quantity) (продать), который уменьшает количество продуктов на складе на заданное количество,  
 и getDetails() (получить информацию), который выводит информацию о продукте, включая название, цену, количество на складе и описание.  
  
 Store (Класс Магазина):  
  
 1. Класс Store представляет магазин и ответственен за управление продуктами.  
 2. Создайте метод addProduct(Product product) (добавить продукт), который добавляет продукт в ассортимент магазина.  
 3. Реализуйте метод listProducts() (список продуктов), который отображает информацию о всех продуктах в магазине,  
 включая их названия, цены и количество на складе.  
 4. Реализуйте метод sellProduct(String productName, int quantity) (продать продукт),  
 который продает указанное количество продуктов с учетом остатка на складе.  
 Обновите количество продуктов на складе после продажи. Учтите проверку того что продуктов достаточно,  
 а если не достаточно то вывести фразу "Не достаточною Введите количество еще раз"  
  
4) Задача: Управление книгами в библиотеке  
  
 Создайте два класса: Book (Книга) и LibraryBook (Книжка в библиотеке) и Library (библиотека), чтобы моделировать управление книгами в библиотеке.  
  
 Book (Класс Книги):  
  
 1. У класса Book должны быть следующие атрибуты: title (название), author (автор), year (год издания) и description (описание).  
 2. Реализуйте конструктор, который принимает параметры для установки значений атрибутов при создании экземпляра.  
 3. Создайте метод getDetails() (получить информацию), который выводит информацию о книге, включая название, автора, год издания и описание.  
  
 LibraryBook (Класс Книжки в библиотеке):  
  
 1. Класс LibraryBook представляет книгу в библиотеке и ответственен за управление книгами.  
 2. Создайте метод checkOutBook() (взять книгу), который регистрирует, что книга была взята из библиотеки.  
 3. Реализуйте метод returnBook() (вернуть книгу), который регистрирует, что книга была возвращена в библиотеку.  
 4. Создайте метод isAvailable() (доступна ли), который проверяет, доступна ли книга для выдачи в данный момент.  
  
 Library (библиотека):  
  
 1. Должна содержать массив LibraryBook.  
 2. Добавлять книгу.  
 3. Получать книгу если доступна  
 4. Получать все книги.  
 5. Удалять книгу  
  
5) Создайте три класса: Course (Курс), Student (Студент) и College (Колледж), чтобы моделировать управление курсами и студентами в колледже.  
  
 Course (Курс):  
  
 1. Класс Course должен иметь следующие атрибуты: name (название) и students (список студентов, записанных на курс).  
 2. Реализуйте конструктор, который принимает параметры для установки значений атрибутов при создании экземпляра.  
 3. Создайте методы addStudent(Student student) (добавить студента) для записи студента на курс (студент должен быть старше 16 лет)  
 4. listStudents() (список студентов) для отображения списка записанных студентов.  
 5. Реализуйте метод getDetails() (получить информацию), который выводит информацию о курсе, включая название и список записанных студентов.  
  
 Student (Студент):  
 Student представляет студента и имеет атрибуты name (имя), yearOfBirth (год рождения) и courses (список курсов, на которые записан студент).  
  
 1. Создайте конструктор для студентов, который принимает имя и yearOfBirth студента.  
 2. Реализуйте методы enrollInCourse(Course course) (записаться на курс) для записи студента на курс и  
 listEnrolledCourses() (список записанных курсов) для отображения курсов, на которые записан студент.  
 3. Реализуйте метод getDetails() (получить информацию), который выводит информацию о студенте, включая имя, yearOfBirth студента и список записанных курсов.  
  
 College (Колледж):  
  
 1. Класс College представляет колледж и отвечает за управление курсами и студентами.  
 2. Создайте методы addCourse(Course course) (добавить курс) для добавления курса в колледж и  
 listCourses() (список курсов) для отображения списка доступных курсов.  
 3. Реализуйте метод addStudent(Student student) (добавить студента) для добавления студента в колледж и  
 listStudents() (список студентов) для отображения списка студентов.  
 4. Реализуйте метод enrollStudentInCourse(String studentName, String courseName) (записать студента на курс) для записи студента на курс и  
 listCoursesForStudent(String studentName) (список курсов для студента) для отображения курсов, на которых учится студент.  
  
6) Создайте три класса: Employee (Сотрудник), Company (Компания) и Department (Отдел), чтобы моделировать управление сотрудниками в компании.  
  
 Employee (Сотрудник):  
  
 1. Класс Employee должен содержать атрибуты: name (имя), id (идентификатор), position (должность) и salary (заработная плата).  
 2. Реализуйте конструктор, который принимает параметры для установки значений атрибутов при создании экземпляра.  
 3. Создайте метод getDetails() (получить информацию), который выводит информацию о сотруднике, включая имя, идентификатор, должность и заработную плату.  
  
 Company (Компания):  
  
 1. Класс Company представляет компанию и отвечает за управление сотрудниками.  
 2. Создайте методы addEmployee(Employee employee) (добавить сотрудника) для добавления сотрудника в компанию и  
 listEmployees() (список сотрудников) для отображения списка всех сотрудников в компании.  
  
 Department (Отдел):  
  
 1. Класс Department представляет отдел в компании и содержит сотрудников, относящихся к этому отделу.  
 2. Создайте методы addEmployee(Employee employee) (добавить сотрудника) для добавления сотрудника в отдел и  
 listEmployees() (список сотрудников) для отображения списка сотрудников, работающих в этом отделе.  
  
 1 - В методе main создать Company с 3 Department  
 2 - В каждом Department должно быть несколько сотрудников с разными должностями и зарплатами  
 3 - Найти общую зарплату всех сотрудников  
 4 - Найти суммы зарплат отдельно по каждому Department  
 5 - Если в каком то департаменте зарплата превышает сумму зарплат оставшихся Department то вывести на экран "Требуется оптимизация"  
  
Extra:  
Самостоятельно продумать управление задачами в календаре и в методе main  
в цикле while(true) запустить имитацию программы Calendar  
(Создание нового события для пользователя, удаление события, возможность провести событие(boolean isFinished), добавить описание)  
  
CalendarEvent (Событие в календаре) - класс, представляющий событие с атрибутами, такими как дата, время и описание.  
Calendar (Календарь) - класс, моделирующий календарь и управляющий событиями.  
User (Пользователь) - класс, представляющий пользователя, создающего события в календаре.  
  
Как правило для управления объектами используются так называемые id т.е. мы можем ввести уникальное поле int id;  
и по нему находить наши объекты в массиве