Описание задач

**Задача 1:**

По управлению количеством продукта с функционалом по типу добавить количество, убрать количество, показать текущее количество и выход. Первым делом после запуска программы у нас появляется меню "Настройка продукта" где первым пунктом будет указание название продукта, после нам нужно ввести начальное количество нашего продукта, после чего мы можем начинать работу с нашим продуктом.

**Задача 2:**

По системе управления задачами с функционалом добавить задачу, показать все задачи, изменить статус задача, удалить задачу, сортировать по приоритету и выход. В этой задаче нас встречает менюшка с выбором действия после добавления задачи мы можем менять ее статус на один из трех доступных NEW, IN\_PROGRESS и DONE. Так же можно удалять задачу если она выполнена или не актуальна. и сортировать их по заданному в них приоритету с самого начала.

**Задача 3:**

Она заключается в том, чтобы создать таблицу с продуктами где будет указанно id, name, price и quntyti для продуктов. Эти параметры задаются через указание массивов. По мимо табличной(пользовательской) формы зписи есть еще и вид для разработчиков. Есть два вида табличного вывода с полем id и без него.

Вызов таблицы с полем id через методы "InventoryManager.displayInventory()" - если вызывать через "InventoryManager", а если через класс "Product" тогда через метод "Product.table()".

Вызов таблицы без поля id выполняется через метод "InventoryManager.getAllProducts()".

Отображени для разработчика выводится путем вызова метода "toString()" из класса Product, первоначально создав экземпляр класса Product.

Присутствуют еще четыре метода:

1. InventoryManager.removeProduct() - этот метод удаляет продукт по его id из таблицы;
2. InventoryManager.getProductById() - этот метод выводит данные по конкретному продукту;
3. InventoryManager.updateQuantity() - этот метод обновляет колличество указанного продукта (поиск совершается по его id);
4. InventoryManager.addProduct() - этот метод добавляет новые продукты.

Условия задач

**Название:** Управление запасами одного продукта(задача 1)

**Описание:** Необходимо создать простую консольную программу для управления запасами одного конкретного продукта на складе. Программа должна позволять:

1. Установить начальное количество продукта.
2. Добавить определенное количество продукта.
3. Удалить определенное количество продукта (если достаточно на складе).
4. Показать текущее количество продукта.

Все действия должны быть интерактивными через консоль.

**Требования:**

* Создать класс Product с полями name (String) и quantity (int).
* Создать класс InventoryManager с методом main, который будет обрабатывать ввод пользователя и вызывать соответствующие методы класса Product или напрямую манипулировать его полями.
* Использовать Scanner для чтения ввода пользователя.
* Обеспечьте базовую обработку некорректного ввода (например, если пользователь вводит не число).

**Название:** Система управления задачами(задача 2)

**Описание:** Разработать консольную систему управления списком задач. Каждая задача должна иметь:

1. Название (String)
2. Описание (String)
3. Приоритет (от 1 до 5, где 1 - самый высокий)
4. Статус (перечисление enum: NEW, IN\_PROGRESS, DONE)

Программа должна позволять:

1. Добавлять новую задачу.
2. Показывать все задачи.
3. Изменять статус задачи по названию.
4. Удалять задачу по названию.
5. Сортировать задачи по приоритету (от самого высокого к самому низкому).
6. Выход.

**Требования:**

* Создать класс Task с полями, конструктором, геттерами и сеттерами.
* Использовать enum для статуса задачи.
* Создать класс TaskManager для управления списком задач.
* Реализовать методы для добавления, удаления, изменения статуса и отображения задач.
* Использовать Collections.sort() с пользовательским Comparator для сортировки задач по приоритету.
* Обеспечить обработку случаев, когда задача с указанным названием не найдена.

**Название:** Управление инвентарем магазина(задача 3)

**Описание:** Необходимо разработать систему для управления инвентарем простого магазина. Система должна позволять добавлять товары, обновлять их количество, удалять товары и выводить список всех товаров с их текущим количеством.

**Требования:**

Класс Product:

* Создайть класс Product с массивами:
* id (уникальный идентификатор товара, например, String или int)
* name (название товара, String)
* price (цена товара, double)
* quantity (количество товара на складе, int)
* hashMap (словарь который будет хранить все данные под ключем принимается поле id а под значением List<Object> в котором будут поля name, price и quntity)
* Реализовать конструктор для инициализации всех массивов.
* Предоставить публичные геттеры для всех массивов.
* Реализовать сеттер только для quantity.
* Переопределить методы equals() и hashCode() на основе поля id, чтобы товары можно было однозначно идентифицировать.
* Создать метод table(int num) который будет принимать количество элементов массива и создавать в отдельном окне таблицу с разбивкой элементов по четырем колонкам(id, name, price и quantity).
* Переопределить метод toString() для удобного представления информации о товаре.

Класс InventoryManager:

* Создайть класс InventoryManager, который будет управлять коллекцией товаров.
* Класс InventoryManager должен наследоваться от класса Product.
* Реализовать следующие методы:
* addProduct(Product product): Добавляет новый товар в инвентарь.
  + Обработка исключений: Если товар с таким id не найден, выбросить исключение ProductNotFoundException сообщением "Имя не найдено."
* updateQuantity(String productId, int newQuantity): Обновляет количество товара по его id.
* getProductById(String productId): Возвращает товар по его id.
* removeProduct(String productId): Удаляет товар из инвентаря по его id.
* getAllProducts(): Возвращает таблицу в новом окне с элементами по найденному id(id в таблице не присутствует).
* displayInventory(): Выводит таблицу элементов из класса Product.
* Создайте приватный конструктор для InventoryManager, чтобы нельзя было создавать экземпляры напрямую.

Класс ProductNotFoundException:

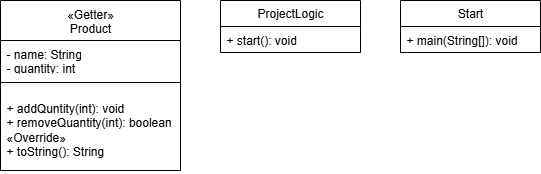
* Создать собственный checked exception ProductNotFoundException, наследующийся от Exception.
* Он должен иметь конструкторы, принимающие сообщение об ошибке.

Класс Main:

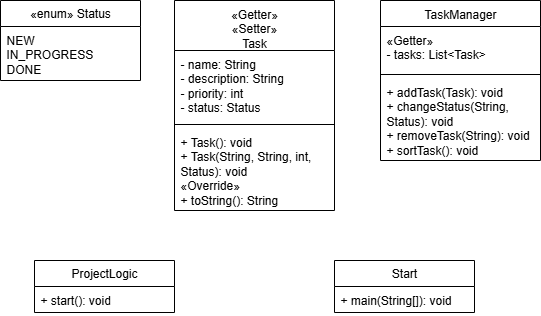
* Создать класс с методом main().
* Создать один объект класса Product.
* Использовать InventoryManager для добавления, обновления, удаления товаров.
* Вывести результаты работы программы.

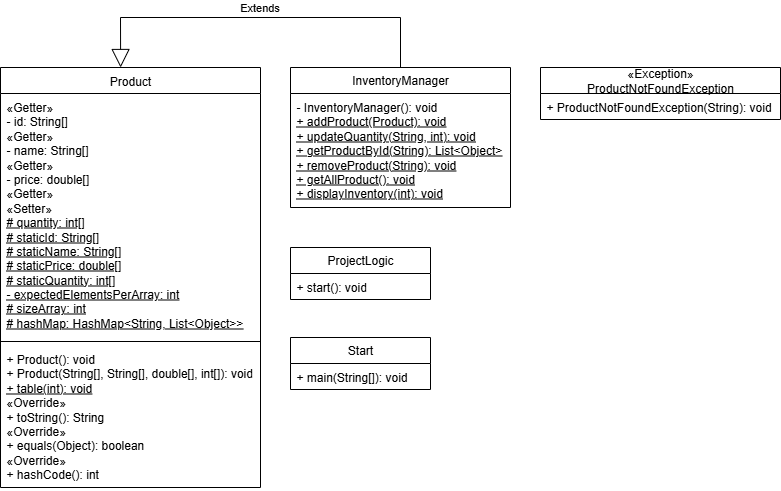
UML-таблицы

**Задача 1:**



**Задача 2:**



**Задача 3:**