Универсальный склад

Вы работаете над системой логистики. Товары на складе могут измеряться по-разному: целыми коробками (Integer), точным весом (Double) или долями паллеты (ваш собственный класс Fraction, который мы делали на паре). Нам нужно универсальное хранилище, которое не только держит объект, но и умеет проводить ревизию

1. Реализация Generic-класса ItemStorage<T>

Создайте параметризованный класс, который будет выступать контейнером для любого типа данных:

- **Поле:** private T value — само значение (товар на складе)
- **Конструктор и геттеры/сеттеры** для доступа к данным
- **Метод void compareWith(T otherValue):**
  - Прежде всего проверьте value и otherValue на null. Если хотя бы один из них пуст, выведите: «Опачки! Ревизия невозможна, объект не найден», чтобы избежать NullPointerException
  - Если значения равны, выведите: «Значения идентичны. Контроль качества пройден»

2. Проверка в классе Main

Продемонстрируйте работу вашего склада в трех сценариях. В комментариях к коду **обязательно** объясните результат (почему вывелось именно это)

Сценарий А:

1. Создайте ItemStorage<Integer> со значением **100.**
2. Вызовите compareWith(100).
3. *Ожидание:* Оба сравнения должны выдать true благодаря диапазону кэширования
4. Как получилось у вас и почему?

Сценарий Б

1. Создайте ItemStorage<Integer> со значением **200**
2. Вызовите compareWith(200)
3. *Ожидание:* Сравнение по ссылке выдаст false, так как объект создается заново вне диапазона -128 до 127

Сценарий В

1. Используйте ваш класс Fraction. Создайте ItemStorage<Fraction> со значением new Fraction(1, 2)

2. Сравните его с другим new Fraction(1, 2)
3. *Важно:* Если .equals() выдает false, значит, в классной работе вы не переопределили метод equals() для дробей. Самое время это исправить!)))

Что прислать?

- Файл ItemStorage.java
- Файл Fraction.java
- Класс Main.java с тестами, демонстрирующими работу во всех трех сценариях