OOP
Lab 11.

🎯 Objectives

- Efficient search

Storage

A text file (`data.txt`) contains product data. Each line contains the following information: identifier (`int`), name (`String`), amount (`int`), price (`int`). Data in a given line is separated by whitespaces. Each identifier is unique.

Create a `Product` class which stores the data and has the following behaviour:

- constructor (4 parameters)
- getters (for each attribute)
- `increaseAmount(int newAmount)` method which increases the amount by newAmount
- `toString()`

Complete the class by adding a **natural ordering**, allowing ordering by `id` (increasing order).

Create a `Storage` class allowing the storage of multiple products. Besides the storage, the class has to allow quick updates of the stored products, therefore you should provide a quick search based on product ID.
Add the following behaviour to the class:

- A `constructor` which has a filename as a parameter, and stores the products read from the file.
- An `update` method which has a filename as a parameter. The method reads the data from the file (`update.txt`) and updates all products read from the file. The structure of the update file is as follows: `id` and `newAmount` separated by whitespaces. This file also contains identifiers that are not in the storage. The method **returns** the number of products that were successfully updated.

Download the input files, run your program for each pair of input files, and measure execution time! Complete the following table:

Products	Updates	Number of products updated	Execution time
`data1000.txt`	`update1000.txt`		
`data1000.txt`	`update1000000.txt`		
`data1000000.txt`	`update1000.txt`		
`data1000000.txt`	`update1000000.txt`		