UNIVERSIDADE DA CORUÑA

Software Design

Practicum Design (2024-2025)

INSTRUCTIONS:
Deadline: December 13, 2024 (until 23:59).

- **The exercises must be solved applying design principles and patterns. A solution that does not use them will be considered incorrect.**
- **Report**: For each exercise you must produce a report that includes the following:
  - Explanation of the **design principles** you have used (particularly the **SOLID** principles) and exactly where they have been used (name the specific classes from your code).
  - Explanation of the **design pattern(s)** you have used. For each pattern you will include the following:
    - **Brief explanation of the chosen pattern** and the rationale for its use.
    - **Class diagram** where the classes involved in the pattern are shown. It is important to indicate the role played by each class in the diagram itself, with UML annotations.
    - **Dynamic diagrams** (sequence, communication or state-machine) showing the dynamic behavior of fundamental aspects of the code. You must decide what type of diagram is more suitable for each exercise.
- **Code and submission**:
  - The exercises will be submitted using *GitHub Classroom*. Invitation link: `https://classroom.github.com/a/c8La-RoB`.
  - You must upload a single IntelliJ IDEA project (including configuration files) with the name of the repository of the *classroom* and two packages (`e1` y `e2`) for the corresponding exercises.
  - You must include unit tests and check the code coverage.
  - The documentation with the design and the principles and patterns will be submitted as a `PDF` file in a `doc` folder inside the IntelliJ IDEA project.
- **Evaluation**:
  - This assignment represents 40% of the final practicum grade. The documentation and the code will be evaluated based on the following criteria:
  - **Quality of documentation**: suitableness of the chosen patterns and principles, clarity of the explanations, quantity and clarity of the submitted diagrams, match between design and code, etc.

- **Quality of code**: correct implementation of design patterns and principles, correct application of the object-oriented philosophy, match between design and code, validity of the tests, etc.

1. **Keeping track of a naval fleet**

   In a world of naval warfare, you are in charge of keeping track of the situation of the fleet, its funds, and each of its vessels, something necessary for the proper functioning of the base.

   Each ship has a specific name and belongs to a specific type of vessel:

   - **Ultra-light ships**: DE (Destroyer Escorts) and DD (Destroyers)
   - **Light ships**: CL (Light Cruise Ships) and AV (Hydroplane Carriers)
   - **Heavy ships**: CA (Heavy Cruise Ships) and CV (Aircraft Carriers)
   - **Ultra-heavy ships**: BB (Battleships)

   At any time, a new vessel may join the base fleet. Each vessel may find itself in different situations within its activities at the naval base, as follows:

   - A vessel may not be at the base, but participating in a naval exercise at a certain location. When a vessel completes an exercise and returns to base, the base receives additional funds as a reward (heavier ships receive higher rewards).
   - If a vessel has been damaged in the exercise, a repair may be requested for the vessel, which will be waiting for the repair. If this is the case, the vessel may not participate in naval exercises until repaired. If the damage is massive, the vessel may sink during the exercise, failing to return to base.
   - The process of repairing the vessel might be adjudicated to the person tasked with it, which has a cost proportional to the weight of the ship. The repair cannot be undertaken if the necessary funds are not available.
   - In accordance with the admiral's decision, it is possible to cancel a repair to give priority to other ships or due to the impossibility of repairing it; or confirm the end of the repair and that the ship is ready to sail. A newly repaired vessel may require further repairs if they have not been carried out correctly.
   - It is possible to request an *express* repair for ultra-light ships. These repairs can be undertaken even if there are no funds available at the base, as long as no other vessel is under repair at the same time. Moreover, these repairs are so quick and easy that we consider them to be "instantaneous": they do not need to be confirmed and cannot be cancelled.
   - When a vessel has reached the end of its useful life, its technology is obsolete, or it does not pay or it is not possible to repair it, the admiral may request its scrapping at the base.

   When a ship's situation is changed in the system, a message should be printed on the screen explaining the change, in order to be viewed by the admiral.

   For example, when repair is requested for a vessel:

```
A repair request has been filed for IJN Musashi (BB) | Expected repair cost: 590000
```

In the same way, several operations are required to see the current situation of the fleet:

- List the active vessels of the fleet, their information and their current situation.
- Know the available funds for the fleet, as well as the expected income and expenses of naval exercises and pending repairs.
- List the inactive vessels of the fleet, as well as their cause of inactivity and the total of completed exercises

An example output to the inactive vessel listing operation is shown below:

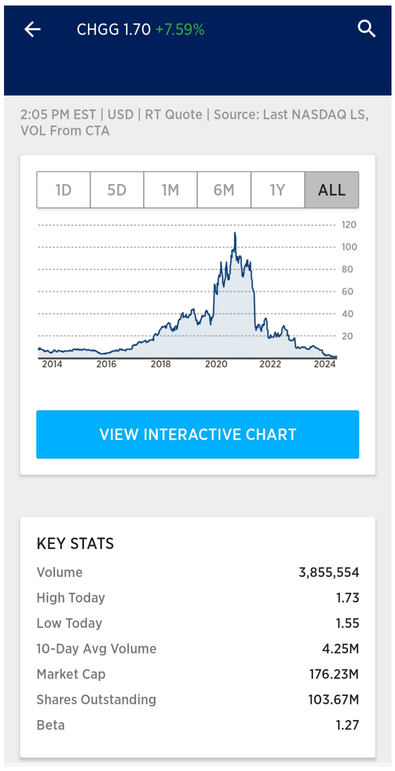
```
INACTIVE VESSELS
---------------------------------
Name: USS Enterprise (CV) | Reasons: Decommissioned | Missions: 20
Name: IJN Shinano (CV) | Reasons: Sunk | Missions: 0
Name: KM UC-48 (SS) | Reasons: Sunk | Missions: 42
Name: KM Prinz Eugen (CA) | Reasons: Decommissioned | Missions: 33
```

You should treat fleet tracking management as an open process, i.e. one that easily accepts to be extended with new functionalities in the future (e.g. vessels captured by the enemy, under refit..., etc). Therefore, your solution must adhere to design patterns and principles.

2. **Valuations of shares in the stock market**

The stock market consists in buying and selling shares (also called stocks). Essentially, shares represent ownership claims on businesses, and their valuations fluctuate during the day. We are interested in the following information about a given stock:

- **Ticker symbol**: Up to four letters representing a stock. For example, Apple's ticker symbol is `AAPL`, Microsoft's is `MSFT`, etc.
- **Closing price**: Price when the stock market closes.
- **Day high**: Maximum intraday price.
- **Day low**: Minimum intraday price.
- **Volume**: Number of shares traded during that day.



CHGG 1.70 +7.59%

2:05 PM EST | USD | RT Quote | Source: Last NASDAQ LS, VOL From CTA

KEY STATS

Volume	3,855,554
High Today	1.73
Low Today	1.55
10-Day Avg Volume	4.25M
Market Cap	176.23M
Shares Outstanding	103.67M
Beta	1.27

Many investors and traders are interested in obtaining data about different shares, but not everyone is interested in the same kind of information. For example, some **simple stakeholders** are only interested in the closing price in order to show it on a display like the ones in the news.

On the other hand, **complex stakeholders** may need all the information about a given stock. For example, in order to display it on trading applications like the one shown above.

Other stakeholders **may be interested only in a particular share or set of shares** and not in all the stock market data. New types of stakeholders with different requirements might also appear in the future.

Design a solution based on design principles and patterns that will let you correctly represent and manage the different types of stakeholders. It must also let you add new types of stakeholders. Each time new stock market data is available, all the interested stakeholders will be updated. When a given pattern has different valid implementations and any of them is suitable for solving the problem, you must choose one and clearly specify which one and why.