# OOP Sample Exam - Empires

You are part of the development team of Empire - a next-generation real-time multiplayer strategy game. You are tasked to design the module for **creating** and **managing buildings**.

The game is **turn-based**, meaning each time a command is executed the game is advanced by **1 turn**. There can be many types of **buildings** in the game (Archery, Barracks, etc.). Each building can produce a **unit** and a **quantity of resources** every few game turns.

## Task 1 - Implement the Game Objects

Implement the following game objects:

**Building** - can produce **a unit** and **a resource.** Should be able to **signal other classes** **whether it can produce a unit or a resource** during the current turn.

* **Barracks** - produces **10 steel** (every **3 turns**) and a **swordsman** (every **4 turns**)
* **Archery** - produces **5 gold** (every **2 turns**) and an **archer** (every **3 turns**)

**Unit** - holds **health** and **attack damage.** A unit's **health can be modified** at some point in the future (e.g. when battles are introduced)**.**

* **Archer** - has default values of **25 health** and **7 damage**
* **Swordsman** - has default values of **40 health** and **13 damage**

**Resource** - holds **type** and **quantity.** Type can be **Gold** or **Steel**.

A building doesn't start production until the turn **after its creation**; therefore, on the turn when it was created, we assume 0 turns have passed. If a building is asked to **produce a unit or a resource** **before** the required turns have passed, it should raise an error (e.g. 2 turns have passed, barracks cannot produce steel yet - 3 turns must pass).

## Task 2 - Improve the Classes

Encapsulate all internal behavior. The production cycles of buildings should be positive; the quantity of produced resources should be non-negative; units should have non-negative damage and positive health at the time of their creation; a unit's health cannot fall below 0 after its creation. The implemented classes should not reveal any internal logic.

Avoid code repetition and promote code reusability by applying the good practices of OOP.

## Task 3 - Engine

Implement an **engine** class that continuously **reads commands** from the input and **dispatches** them. The engine should support the following four commands:

* **build <building-type>** - adds a new building of the specified type to the game
* **skip** - does nothing, skips the turn and progresses the game
* **empire-status** - prints data about the current state of the game in the following format:

|  |
| --- |
| **Treasury:**  **--Gold: {gold}**  **--Steel: {steel}**  **Buildings:**  **--{building}: {turns-passed} turns ({n} turns until {unit}, {m} turns until {resource})**  **…**  **Units:**  **--{unit-type}: {unit-count}**  **--{unit2-type}: {unit2-count}**  **…** |

The buildings and units should be listed in **order of creation**. If there are no units/buildings in the game yet, print "**N/A**". If two units are created on the same turn, print them based on the order of creation of the buildings which created them. In the example below, an archer and a swordsman are created on the same turn, but the swordsman should be printed first since the Barracks was created earlier in the game.

* **armistice** - ends the program

Each command should progress the game with **1 turn after** it is executed.

The engine should consume a produced unit / resource **as soon as it has been produced** by a building and save it.

## Task 4 - Loose Coupling

The engine should be designed to work with **any buildings**, **units** and **resources**.

## Task 5 - Input / Output Independence

The engine should be designed to work with **any input source** and **output destination**. In other words, it should **NOT** depend on the console.

## Input

The input will be read from the standard input. On each line a command will be given (one of the described above).

## Output

The output should be printed on the console. Upon receiving the empire-status command, print the current status of the empire as described above.

## Constraints

* All building and unit stats will be 32-bit integer numbers; no overflow will occur at any point during the execution of the program.
* The input will always end with the armistice command.

## Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Barracks** | | **Archery** | | **Output** |
| **Unit** | **Resource** | **Unit** | **Resource** |
| **build** **barracks** | **4** | **3** | **-** | **-** | **Treasury:**  **--Gold: 0**  **--Steel: 0**  **Buildings:**  **--Barracks: 1 turns (3 turns until** **Swordsman, 2 turns until** **Steel)**  **--Archery: 0 turns (3 turns until Archer, 2 turns until** **Gold)**  **Units:**  **N/A**  **Treasury:**  **--Gold: 5**  **--Steel: 10**  **Buildings:**  **--Barracks: 4 turns (4 turns until Swordsman, 2 turns until Steel)**  **--Archery: 3 turns (3 turns until** **Archer, 1 turns until Gold)**  **Units:**  **--Swordsman: 1**  **--Archer: 1** |
| **build** **archery** | **3** | **2** | **3** | **2** |
| **empire-status** | **2** | **1** | **2** | **1** |
| **skip** | **1** | **0  (reset to 3)  (+10 steel)** | **1** | **0  (reset to 2)  (+5 gold)** |
| **skip** | **0  (reset to 4) (+1 swordsman)** | **2** | **0  (reset to 3) (+1 archer)** | **1** |
| **empire-status** | **3** | **1** | **2** | **0  (reset to 2)**  **(+5 gold)** |
| **armistice** | - | - | - | - |
|  |  | |  | |

|  |  |
| --- | --- |
| **Input** | **Output** |
| build archery  build archery  skip  skip  build archery  empire-status  build barracks  skip  skip  skip  skip  empire-status  armistice | Treasury:  --Gold: 15  --Steel: 0  Buildings:  --Archery: 4 turns (2 turns until Archer, 2 turns until Gold)  --Archery: 3 turns (3 turns until Archer, 1 turns until Gold)  --Archery: 0 turns (3 turns until Archer, 2 turns until Gold)  Units:  --Archer: 2  Treasury:  --Gold: 60  --Steel: 10  Buildings:  --Archery: 10 turns (2 turns until Archer, 2 turns until Gold)  --Archery: 9 turns (3 turns until Archer, 1 turns until Gold)  --Archery: 6 turns (3 turns until Archer, 2 turns until Gold)  --Barracks: 4 turns (4 turns until Swordsman, 2 turns until Steel)  Units:  --Archer: 8  --Swordsman: 1 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| build barracks  empire-status  empire-status  empire-status  empire-status  empire-status  empire-status  armistice | Treasury:  --Gold: 0  --Steel: 0  Buildings:  --Barracks: 0 turns (4 turns until Swordsman, 3 turns until Steel)  Units:  N/A  Treasury:  --Gold: 0  --Steel: 0  Buildings:  --Barracks: 1 turns (3 turns until Swordsman, 2 turns until Steel)  Units:  N/A  Treasury:  --Gold: 0  --Steel: 0  Buildings:  --Barracks: 2 turns (2 turns until Swordsman, 1 turns until Steel)  Units:  N/A  Treasury:  --Gold: 0  --Steel: 10  Buildings:  --Barracks: 3 turns (1 turns until Swordsman, 3 turns until Steel)  Units:  N/A  Treasury:  --Gold: 0  --Steel: 10  Buildings:  --Barracks: 4 turns (4 turns until Swordsman, 2 turns until Steel)  Units:  --Swordsman: 1  Treasury:  --Gold: 0  --Steel: 10  Buildings:  --Barracks: 5 turns (3 turns until Swordsman, 1 turns until Steel)  Units:  --Swordsman: 1 |