# OOP Advanced Exam – Minedraft

You ever heard about the Rick and Morty’s Foundation for mining Plumbus Ore. Naaa, probably not. Well let’s just say there is this company that mines things, and they hired you to write them a supervising software. A draft which will be used to analyze the data of the mining – a … Minedraft.

### Overview

You have a task to write a software program, which represent a system for future mining industry. You already have some of the functionalities implemented, although ... the guy that started the project … well it’s obvious he is **quite a rookie** so you need to **refactor** his code and finish the project. The **only** thing he has **done right** is the **class** ProviderController and **ALL** the **INTERFACES**. You must leave these as they are and **don’t modify anything**.

## Task 1: Structure

The main structure of the program should include the following elements:

* **Engine**
* **CommandInterpreter**
* **HarvesterController**
* **ProviderController**
* **EnergyRepository**

Everywhere you need to inject all dependencies by constructor injection.

The Structure also consists of Harvesters and Providers.

### Harvesters

A basic Harvester has the following properties:

id - a **int**.

oreOutput - a **floating-point number**.

energyRequirement - a **floating-point number**.

durability - a **floating-point number**.

All harvesters start with **1000** **durability**. There are generally **4** types of Harversters:

#### StandartHarvester

This type of harvester doesn’t have anything special, so all stats are standart for harvester.

#### SonicHarvester

Really fast and of course weaker than standart ones. Upon initialization, their **energy requirement** is divided by 2, but their **durability** is also decreased by **300**.

#### HammerHarvester

Heavy harvesters are bigger than standart ones. Upon initialization, their **energy requirement** is **multiplied** by **2**, but their ore output is **multiplied** by **4**.

#### InfinityHarvester

Infinity harvesters are very special, they have special ability to repair themselves over nights. Upon initialization, their **ore ourput** is **divided** by **10**, but they **cannot be broken.**

### Providers

A basic Provider has the following properties:

id - a **int**.

energyOutput - a **floating-point number**.

durability - a **floating-point number.**

All providers start with **1000** durability. There are generally **3** types of Providers:

#### StandartProvider

This type of provider doesn’t have anything special, so all stats are standart for provider.

#### SolarProvider

Extracts energy from the Sun. Upon initialization, their **durability is increased by 500**.

#### PressureProvider

Extracts energy from deep beneath the earth. Upon initialization, their **durability is decreased by 300**, but their **energy output is multiplied by 2**.

## Task 2: Business Logic

### The Command Pattern

The business logic of the program should be concentrated around several commands, so this program is perfect place to implement Command pattern. Implement a class called Engine, which will read commands and will pass them to the CommandInterpreter.

### Functionality

The whole system never stops working, but for easy counting how much work is done, process is separated to days. Every day all the Providers produce energy and the Harvesters consume energy and mine Plumbus Ore. In your program a **day passes** when you have been given the **corresponding command**.

The Providers produce energy which is being stored on the system. When there is **ENOUGH** **energy** for a whole day, **power up** **ALL** Harvesters, the Harvesters **consume** it and return the ore.

Because of terrible sound, which Harvester produce, they work in different modes, which are 3 types. Each mode change their **energy requirement** and **ore output** to percent of their initial values.

Energy - **20%**

Half - **50%**

Full - **100%**

For now it seems so easy and perfect, but like you know from real world, everything can be broke. Harvesters and Providers can be broke too. Actually they get broken at different events:

Harvesters - lose 100 of their durability each time, when **mode is changed.**

Providers - lose 100 of their durability each **day.**

When any entity durability is **reach less than 0**, it is broken and should be remove from system. Here is difference between providers and harvesters. Providers **can** be repaired, but harvesters **CAN** **NOT** be repaired.

### Commands

There are several commands that control the business logic of the application you are supposed to build.   
They are stated below.

#### Register Command

Creates a Harvester or Provider, and registers it into the system, so they can start working.

type of entity - a **string**, equal to either Harvester or Provider.

type - a **string,** that correspond with types of Harvester or Provider.

id – a **int**.

If you receive Provider, command end with:

energy output - a **double**.

If you receive Harvester, command end with:

energy requirement - a **double**.

ore output - a **double**.

#### Day Command

When you receive this command, a day passes. This is the moment, where real work start. You need to **calculate** all provided **energy** and **STORE** it on the system. Then you need to **check** if there is **enough** **energy** for harvesters to start mining. If energy requirement of **ALL** harvesters is more than **stored energy** then **NOTHING** happen. But if there is **enough energy**, **ALL** harvesters **start mining** and they **consume** **energy** from store **EQUAL** to their energy requirement.

**NOTE**: The summed up energyRequirement might be **less** or **more** depending on the current **working** **Mode**.

#### Mode Command

Changes the **mode** of the system, to the **given one**.

mode - a **string**, equal to either Full, Half or Energy.

#### Repair Command

Repair all providers with value

value - **double.**

#### Inspect Command

**Checks** the Provider or the Harvester with the **given id**, returning a **string representation** of it. The system should check if there is an **element** with the **given id** among the Providers or the Harvesters. The **ids** are **unique** so there should be only **one** with that **id**.

id - **int**

#### Shutdown Command

**Ends** the program and **prints the total ore mined.**

## Task 3: Tasks

### Reflection

You need to refactor the given Engine class by implementing a Command pattern. You are given an ICommand interface, which you need to implement in all your commands. You should implement all commands.

RegisterCommand

DayCommand

InspectCommand

ModeCommand

RepairCommand

ShutdownCommand

### Unit Testing

Like you see at the beginning there is а class, which does not need refactoring - ProviderController**.** This is the class, against which you need to **write unit tests**. For easy testing, there are some stuff that are not high quality (setters are public, etc.), but you can use them in your unit tests. In your skeleton, the ProviderController isworking **perfect**, but it still needs to be tested, because in **Judge** we have prepared some **bugs** and you need to catch them in your unit tests.

Do **NOT** use **Mocking** in your unit tests!

### Input

Below, you can see the **format** in which **each command** will be given from the input:

* Register Harvester Sonic {id} {oreOutput} {energyRequirement}
* Register Provider Hammer {id} {energyOutput}
* Day
* Mode {mode}
* Inspect {id}
* Repair {id} {rapairValue}
* Shutdown

### Output

#### Register Command

If registration is successful, print message:

“Successfully registered {Class Name}”.

#### **Day Command**

After each day command, you need to print:

“Produced {energy produced this day} energy today!

Produced {ore produced this day} ore today!”.

#### Mode Command

On mode changing you need to print:

“Mode changed to {new mode}!”.

#### Repair Command

Repair a provider with a specific id. The provided id will always be valid and will be a provider’s id, so you don’t need to check it.

“Providers are repaired by {value}”

#### Inspect Command

Should return a **string representation** of the element with the **given id**. If there is no such entity, the command should output a message:

“No entity found with id – {id}”.

If there is a entity with this id, you need to print:

“{full type of entity}

Durability: {entity durability}”.

#### Shutdown Command

Should output a message

“System Shutdown

Total Energy Produced: {totalEnergyProduced}

Total Mined Plumbus Ore: {totalMinedOre}”.

The totalEnergyStored and totalMinedOre are the total values that have been gathered throughout the program’s execution.

### Constraints

* All members in our mining industry program **(harvesters and providers)** have **unique id's**.
* All **floating-point numbers** will be in **range [-100.000, 100.000]**.
* There will be **NO invalid** input data.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Register Harvester Hammer 20 100 100  Register Provider Pressure 40 100  Day  Mode Half  Day  Inspect 60  Shutdown | Successfully registered HammerHarvester  Successfully registered PressureProvider  Produced 200 energy today!  Produced 400 ore today!  Mode changed to Half!  Produced 200 energy today!  Produced 200 ore today!  No entity found with id - 60  System Shutdown  Total Energy Produced: 400  Total Mined Plumbus Ore: 600 |
| Register Harvester Standart 20 100 100  Register Harvester Standart 40 100 100  Inspect 20  Day  Mode Energy  Day  Shutdown | Successfully registered StandartHarvester  Successfully registered StandartHarvester  StandartHarvester  Durability: 1000  Produced 0 energy today!  Produced 0 ore today!  Mode changed to Energy!  Produced 0 energy today!  Produced 0 ore today!  System Shutdown  Total Energy Produced: 0  Total Mined Plumbus Ore: 0 |
| Register Harvester Hammer 20 100 100  Register Harvester Sonic 30 100 50  Register Provider Pressure 40 100  Register Provider Solar 80 100  Day  Day  Day  Mode Full  Mode Half  Inspect 30  Repair 20  Day  Day  Mode Half  Mode Energy  Mode Full  Mode Half  Mode Energy  Mode Half  Mode Energy  Mode Half  Mode Energy  Inspect 30  Day  Shutdown | Successfully registered HammerHarvester  Successfully registered SonicHarvester  Successfully registered PressureProvider  Successfully registered SolarProvider  Produced 300 energy today!  Produced 500 ore today!  Produced 300 energy today!  Produced 500 ore today!  Produced 300 energy today!  Produced 500 ore today!  Mode changed to Full!  Mode changed to Half!  SonicHarvester  Durability: 500  Providers are repaired by 20  Produced 300 energy today!  Produced 250 ore today!  Produced 300 energy today!  Produced 250 ore today!  Mode changed to Half!  Mode changed to Energy!  Mode changed to Full!  Mode changed to Half!  Mode changed to Energy!  Mode changed to Half!  Mode changed to Energy!  Mode changed to Half!  Mode changed to Energy!  No entity found with id - 30  Produced 300 energy today!  Produced 0 ore today!  System Shutdown  Total Energy Produced: 1800  Total Mined Plumbus Ore: 2000 |