# C# OOP Advanced Exam – The Last Army

Not too long ago, the protection forces in Bulgaria almost ended. There was almost nobody who can protect our country from the bad guys and the evil which was spreading faster that any time before in history. Street fights, dangerous criminals walking all around and terrorism in not a few places. The last army we left was composed from 3 special forces – **Ranker**, **Corporal**, **SpecialForce.**

**Each** of these forces have **special skills** on which they were trained to protect the nation.

# Overview

You have a task to write a software program, which will generate army, warehouse and will perform different missions. 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** **MissionController** and **ALL** the **INTERFACES**. You must leave these as they are and **don’t modify anything**.

# Task I : Structure

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

* **Engine**
* **GameController**
* **MissionController**

## Soldiers

An abstract **soldier** has the following characteristics:

* string  **name**
* Int32  **age**
* Double  **experience, endurance**

You need to create three **classes** for each of the three **types** of soldiers we have:

* **Ranker**
* **Corporal**
* **SpecialForce**

An Army is nothing without weapons. That’s why the president of the country gave our special forces the last warehouse which is left. Although it is the last one, it has all the weapons and military equipment which the army needs. In other words, a **soldier does contain a warehouse of weapons**.

## Warehouse (Ammunitions)

You should create an **abstract** class **Ammunition** with the following parameters:

* string  **name**
* Double  **weight**

The following classes, inherit **Ammunition**:

* **Class** **Gun**
* **Class** **AutomaticMachine**
* **Class** **MachineGun**
* **Class** **RPG**
* **Class** **Helmet**
* **Class** **NightVision**
* **Class** **Knife**

## Missions

An **abstract** **class Mission** will have the following **parameters**:

* Double  **enduranceRequired**, **scoreToComplete**

The following classes will inherit **Mission** and each of them has a constant value for **enduranceRequired**:

* **Class Easy - 20**
* **Class Medium - 50**
* **Class Hard - 80**

# Task II: Business Logic

## Ammunitions Rules:

**Each type** of ammunition has a **specific** constant **weight**, which is the following:

* AutomaticMachine (**6.3**)
* Gun: (**1.4**)
* Helmet: (**2.3**)
* Knife: (**0.4**)
* MachineGun: (**10.6**)
* NightVision: (**0.8**)
* RPG: (**17.1**)

Every ammunition has also **wear level**, which decreases after each successful mission. Upon creation, each ammunition has a **wear** **level** equal to its **weight \* 100**.

## Soldier Rules

Each type of soldier can wear a different list of ammunitions:

* **Ranker:**
  + **Gun**
  + **AutomaticMachine**
  + **Helmet**
* **Corporal:**
* **Gun**
* **AutomaticMachine**
* **MachineGun**
* **Helmet**
* **Knife**
* **SpecialForce -** they are а different story, **they have to wear all the types of ammunitions**, because they are “Special”.
* **Gun**
* **AutomaticMachine**
* **MachineGun**
* **RPG**
* **Helmet**
* **Knife**
* **NightVision**

When you **create a soldier**, you need to check in the **Warehouse**, if **all the ammunitions** for this type of soldier **are available**. If **any** ammunition **is missing**, the soldier **doesn’t get** **added** **to the army** and you should print - **"There is no weapon for {type} {name}!"**

There is also one very **IMPORTANT** statistic about each soldier **-** his **OverallSkill**, which is the **sum** of a soldiers’ **age** and **experience**. After adding these numbers together, each **concrete type** of soldier must multiply the result by:

* **Ranker - 1.5**
* **Corporal - 2.5**
* **SpecialForces - 3.5**

From time to time the soldiers in the army experience **REGENERATION,** which represents an **increase** of their **endurance**. This happens only upon the **Regenerate** **command** and affects only the type of soldiers specified in the command parameters.

* **Ranker’s** and **Corporal’s** endurance gets increased by **10** **plus the age** of the soldier
* **SpecialForce** again are special and their endurance gets increased by **30** **plus the age** of the soldier

A soldier’s **endurance** should **never exceed the amount of** **100**! The unwanted endurance just disappears, without any error generated.

**Before** **each** mission, you need to **check** which soldiers in the army **are ready for the mission**. There are three conditions for a soldier to participate in a mission:

1. He needs to have **enough endurance** for the current mission ;

2. He has to be **equipped with all needed ammunitions** for this type of solder;

**3. All his ammunitions** need to have **at least positive value of wear level**.

**Before** you send soldier on mission, **you have to check in the Warehouse for ammunitions**. They might be created later. Each soldier replaces his worn out ammunitions with the currently available in the Warehouse, if they are available. If a soldier **doesn’t get fully equipped** (gathers all the needed ammunitions) he **keeps the ammunitions** **he has found** but still does not participate in the mission.

In order to participate in a mission, a soldier **MUST** be equipped with **all the ammunitions**, which **he can wear** corresponding to his type.

## Missions Rules

**Performing** missions is **done immediately after its creation** and can be successful … or not. If a **mission is unsuccessful it is put on hold** while there is a team that is capable of executing it. Although the **maximum number of missions on hold is 3**. If there is a **forth** mission called, the **oldest one is declined**.

### Easy

The **first** **mission (Easy)** is named “**Suppression of civil rebellion**” is the **easiest** one. Here is some more specific information about that mission:

* The **endurance** of the soldiers **goes down** by **20 after completing the mission**
* The **wear** **level** of **EACH** ammunition of the soldiers **decreases by 30 after completing the mission**
* Only soldiers who have **AT LEAST** **20** **endurance** can **participate in it**

### Medium

The **Second** **mission (Medium)** is named “**Capturing dangerous criminals**”. Here is more information about the mission:

* The **endurance** of the soldiers **goes down** by **50 after completing the mission**
* The **wear level** of **EACH** ammunition of the soldiers **decreases by 50 after completing the mission**
* Only soldiers who have **AT LEAST** **50** **endurance** can **participate in it**

### Hard

The **last** **mission** **(Hard)** which is the hardest one and most dangerous one - “**Disposal of terrorists**”. Here is more information about the mission:

* The **endurance** of the soldiers **goes down** by **80 after completing the mission**
* The **wear level** of **EACH** ammunition of the soldiers **decreases by 70** after completing the mission
* Only soldiers who have **AT LEAST** **80** **endurance** can **participate in it**

А mission gets completed if the **sum** of **all participating** soldiers **overall skill** is **equal or bigger** than the missions’ **score to complete**.

At the end of а **successfully completed** mission all soldiers that took part in the mission **gain experience** **equal to endurance** **required for the mission**.

After **the end of the input** all missions which are **left on hold** must be counted as **FAILED**!

## Commands

1. **Adding** a soldier to the army

Soldier [Type] [Name] [Age] [Experience] [Endurance]

1. **Adding** ammunitions to the warehouse

Warehouse [Name] [Count]

1. **Missions** command:

Mission [Type] [Score to Complete]

1. **Regenerating** soldiers command:

Soldier Regenerate [Soldier Type]

# Tasks III:

## Reflection

You need to refactor given factories and implement new ones. Factories have to **use reflection**, so it will be easy for us to follow **Open/Closed Principe**. You need to have **three factories**:

* **Class** SoldierFactory
* **Class** AmmunitionFactory
* **Class** MissionFactory

Your task is to implement these factories in such a way that it will be **easy to extend the number of concrete types of each entity**. With simple words, your factories need to work the same way without modifications, if someone **adds a new class**, which **implements** ISoldier**,** IAmmunition **and** IMission

**No static** factoriesareallowed**!**

## Unit Testing

Like you see at the beginning there is а class, which does not need refactoring - MissionController**.** This is the class, against which you need to **write unit tests**. For easy testing, there is some stuff that are not high quality (setters are public), but you can use them in your unit tests. In your skeleton, the MissionControllerisworking **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

* The input will come from the console in the form of commands, in the format specified above - each command on new line
* The input sequence ends when you receive the command “**Enough! Pull back!**”
* **Any** type of command, except the “**Enough! Pull back!**” can be given at any time.

## Output

Upon creating Soldier command, if it is **not possible to equip the soldier with ammunitions** and thus add him to the army, print:

* “**There is no weapon for {Soldier Type} {Name}!**”

Upon **each** Mission **command**, you should print on the console:

* If there is **no room in the mission queue** for the incoming mission: “**Mission declined - {Name}**”

For each mission, **currently in the queue**, print:

* If the mission is successful: “**Mission completed - {Name}**”
* If the mission is unsuccessful: “**Mission on hold - {Name}**”

The next lines of the output have to be printed at the end, after receiving the command "**Enough! Pull back!**".

First, you need to print summary for all the missions in the format:

**Results:**

**Successful missions - {number}**

**Failed missions - {number}**

Then you need to print all soldiers in the army, in **descending order by their overall skill** in the format:

**Soldiers:**

**{Name} - {OverallSkill}**

## Constrains

* **All ints** in the input will be in the **range [0, 1000]**
* **All Doubles** will be in the **range [0, 1000]**
* All input lines will be **VALID** command
* There will be **NO MORE** than **500** commands
* **All** rules **specified above** will be **strictly** **followed**, there will be **NO** unexpected input or conditions

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| WareHouse AutomaticMachine 4  WareHouse Gun 6  WareHouse Helmet 20  Soldier Ranker Ivan 28 55 100  Mission Easy 150  Enough! Pull back! | Mission on hold - Suppression of civil rebellion  Results:  Successful missions - 0  Failed missions - 1  Soldiers:  Ivan - 124.5 |
| WareHouse AutomaticMachine 30  WareHouse Gun 2  WareHouse Helmet 9  Mission Easy 1500  Soldier Ranker Ivan 47 23 100  WareHouse Knife 8  WareHouse MachineGun 4  Soldier Corporal Ivaylo 21 78 100  Mission Medium 180  Enough! Pull back! | Mission on hold - Suppression of civil rebellion  Mission on hold - Suppression of civil rebellion  Mission completed - Capturing dangerous criminals  Results:  Successful missions - 1  Failed missions - 1  Soldiers:  Ivaylo - 372.5  Ivan - 180 |
| Mission Hard 5000  Mission Hard 12  Mission Hard 12  Mission Hard 12  WareHouse AutomaticMachine 5  WareHouse RPG 2  WareHouse NightVision 4  WareHouse AutomaticMachine 30  WareHouse Gun 2  WareHouse Helmet 9  Soldier Ranker Ryan 47 23 100  Mission Hard 12  WareHouse Knife 8  WareHouse MachineGun 4  Soldier Corporal Gosho 21 10 100  Soldier Regenerate Corporal  Soldier Ranker Pijo 79 19 50  Mission Medium 79  Enough! Pull back! | Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission declined - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission declined - Disposal of terrorists  Mission completed - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Disposal of terrorists  There is no weapon for Ranker Pijo!  Mission completed - Disposal of terrorists  Mission on hold - Disposal of terrorists  Mission on hold - Capturing dangerous criminals  Results:  Successful missions - 2  Failed missions - 4  Soldiers:  Gosho - 277.5  Ryan - 225 |