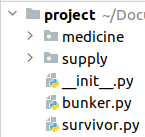
# Bunker Games

*You and your friends gathered together to play a new game called "Bunker Games" in your basement!  
Your job is to help your teammates ration the supplies and medicine they have, so you can win the game!*

You will be provided with a **skeleton** which includes all the folders and files that you will need.

***Note: You are not allowed to change the folder and file structure and change their names!***



# Judge Upload

For the **first 2 problems**, create a **zip** file with the **project** **folder** and **upload it** to the judge system

For the **last problem**, create a **zip** file with the **tests** **folder** and **upload it** to the judge system

# Structure (Problem 1) and Functionality (Problem 2)

Our first task is to implement the **structure and functionality** of all the classes (properties, methods, inheritance, etc.)

## Class Supply

In the file **supply.py** the class **Supply** should be implemented:

### Structure

The class should be **abstract**, and should have the following attributes:

* **needs\_increase: int** – passed upon **initialization** (if it is a **negative number** raise **ValueError** with message **"Needs increase cannot be less than zero."**)

### Methods

**\_\_init\_\_(needs\_increase: int)**

The **\_\_init\_\_** method should receive a **needs\_increase: int**

**apply(survivor: Survivor)**

Method should **increase** the **needs property** of the given survivor with **the supply's** **needs\_increase** value

## Class FoodSupply

In the **food\_supply.py** file the class **FoodSupply** should be implemented

### Structure

The class should **inherit** from the **Supply** class

### Methods

#### \_\_init\_\_()

An instance of the **FoodSupply** class will have **needs\_increase** of **20**

## Class WaterSupply

In the **water\_supply.py** file the class **WaterSupply** should be implemented

### Structure

The class should **inherit** from the **Supply** class

### Methods

#### \_\_init\_\_()

An instance of the **WaterSupply** class will have **needs\_increase** of **40**

## Class Medicine

In the **medicine.py** file the class **Medicine** should be implemented

### Structure

The class should be **abstract**, and should have the following attributes:

* **health\_increase: int** – passed upon initialization (if it gets **less than zero**, raise **ValueError** with message **"Health increase cannot be less than zero."**)

### Methods

**\_\_init\_\_(health\_increase: int)**

The **\_\_init\_\_** method should receive a **health\_increase: int**

**apply(survivor: Survivor)**

Method should **increase** the **health property** of the given survivor with **the medicine's** **health\_increase** value

## Class Painkiller

In the **painkiller.py** file the class **Painkiller** should be implemented

### Structure

The class should **inherit** from the **Medicine** class

### Methods

#### \_\_init\_\_()

An instance of the **Painkiller** class will have **health\_increase** of **20**

## Class Salve

In the **salve.py** file the class **Salve** should be implemented

### Structure

The class should **inherit** from the **Medicine** class

### Methods

#### \_\_init\_\_()

An instance of the **Salve** class will have **health\_increase** of **50**

## Class Survivor

The Survivor class will store the info of each survivor

### Structure

The class should have the following attributes:

* **name: str** – passed upon initialization (if it's set to an **empty string**, raise **ValueError** with message **"Name not valid!"**)
* **age: int** – passed upon initialization (if it's set to a number **less than zero**, raise **ValueError** with message **"Age not valid!"**)
* **health: int** – **100** upon initialization (if its set to a number **less than zero**, raise **ValueError** with message **"Health not valid!"**, its **max value** is **100**)
* **needs: int** – **100** upon initialization (if its set to a number **less than zero**, raise **ValueError** with message **"Needs not valid!"**, its **max value** is **100**)
* **needs\_sustenance** – bool **property** that returns if the survivors **needs** are **less than 100**
* **needs\_healing** – bool **property** that returns if the survivors **health** is **less than 100**

### Methods

**\_\_init\_\_(name: str, age: int)**

Upon initialization all the needed attributes must be set.

## Class Bunker

The **Bunker** class will contain **all the functionality** of our project

### Structure

The Bunker class will have the following **attributes**:

* **survivors: list** – **empty** list upon initialization that will contain **all the survivors** (objects)
* **supplies: list** – **empty** list upon initialization that will contain **all the supplies** (objects)
* **medicine: list** – **empty** list upon initialization that will contain **all the medicine** (objects)
* **food: list** – **property** that returns **only the food** objects from the **supplies** (if there are **no food** supplies, raise **IndexError** with message **"There are no food supplies left!"**)
* **water: list** – **property** that returns **only** **the water objects** from the **supplies** (if there are **no water** supplies, raise **IndexError** with message **"There are no water supplies left!"**)
* **painkillers: list** – **property** that returns **only the painkiller objects** from the **medicine** (if there are **no painkillers**, raise **IndexError** with message **"There are no painkillers left!"**)
* **salves: list** – **property** that returns **only the salve objects** from the **medicine** (if there are **no salves**, raise **IndexError** with message **"There are no salves left!"**)

### Methods

**add\_survivor(survivor: Survivor)**

If a survivor **already exists**, raise **ValueError** with the message **"Survivor with name {name} already exists."**. Otherwise **add the survivor** to the survivors list

**add\_supply(supply: Supply)**

**Adds** the supply to the **supplies list**

**add\_medicine(medicine: Medicine)**

**Adds** the medicine to the **medicine list**

**heal(survivor: Survivor, medicine\_type: str)**

Remove the **last medicine** added to the medicine list from the given type (if the survivor needs it), **apply** it to him/her and return a message **"{survivor\_name} healed successfully with {medicine\_type}"**

**sustain(survivor: Survivor, sustenance\_type: str)**

Remove the **last supply** added to the supplies list from the given type (if the survivor needs it), apply it to him/her and return a message **"{survivor\_name} sustained successfully with {sustenance\_type}"**

**next\_day()**

* First, the **needs** of each survivor **get reduced** by the result of **multiplying** his/her **age** by **2**
* Then we need to **sustain** each survivor by **giving** him/her **one food** and **one water** supply

# Problem 3. Unit Tests

You will **be provided with another skeleton** for this problem. **Open** the **new skeleton** as a **new project** and write tests for the **PaintFactory** class. The class will have some methods, fields and one constructor, which are working properly. You are **NOT ALLOWED** to change any class. Cover the whole class with unit tests to make sure that the class is working as intended. Submit **only the test** folder.