T03: REFLECTION

TOOL DEVELOPMENT

OVERVIEW

```
******* CHARACTERS IN THE GAME ********
- NINJA CAT:
This character has high speed and agility, and can perform a ranged attack with ninja stars. The number of ninja stars is random.
- ALIEN OCTOPUS:
This character can shoot ink at enemies to blind them, and can cling to walls and ceilings. When it dies, it teleports away in a beam of light.
- GIANT TEDDY BEAR:
This character can pick up and throw enemies, or give hugs to heal one if it decides it is a friend. When it dies, it deflates into a pile of stuffing.
- BANANA SLUG:
This character is very slow but has a lot of health, and can leave a trail of slime that decreases the attack power of enemies. When it dies, it turns into a pile of goo.
- JELLYBEAN:
This character has a low health pool but deals a lot of damage with its sticky attack. When it dies, it explodes into a rainbow of colors.

******* FIGHT! FIGHT! FIGHT!! ********
What's that sticking to the wall there? It's the Alien Octopus!
Ninja Cat gets hit by it's ink and is temporarely blinded. It decreases its attack and deals damage.
NINJA CAT [Health: 4, Attack: 1]

******* FIGHT! FIGHT! FIGHT!! ********
JellyBean uses its sticky atack on Ninja Cat!
NINJA CAT [Health: -6, Attack: 1]

++++++++++ DEFEAT ++++++++++
Ninja Cat took the last damage and disappeared in a cloud of smoke.
=> 4 characters are left in the game!

******* FIGHT! FIGHT! FIGHT!! ********
A trail of slime was left by Banana Slug, decreasing Giant Teddy Bear's attack power from 3 to 1!
GIANT TEDDY BEAR [Health: 6, Attack: 1]
```

```
JellyBean uses its sticky atack on Banana Slug!
BANANA SLUG [Health: 1, Attack: 0]

******* FIGHT! FIGHT! FIGHT!! ********
JellyBean uses its sticky atack on Banana Slug!
BANANA SLUG [Health: 0, Attack: 0]

++++++++++ DEFEAT ++++++++++
Does it smell like milkshake in here? Banana Slug died and turned into a pile of goo.
=> 1 characters are left in the game!

******* WINNER!! ********
JellyBean
```

0. Project description and setup

> *The main goal of this exercise is to have a game that can **dynamically (= at runtime!) load all possible game characters** into the game and make them fight.*
>
> *On the one hand, we have the game called **PluginLoader**. It just loads all .dll files from a folder, and checks if it contains one or more characters. If so, it turns them into a game character and lets it join the game.*
>
> *On the other hand, we have the **plugins**. They are dll files that contain one or more characters. You receive a few ready-to-use dll files, but you will also create your own plugins. If you do things right, they should be loaded into the game without changing any code in the PluginLoader.*

- Find and extract the resource rar file that contains the **PluginLoader** and **plugins** folder on Leho.

ILDASM

We will explore the given **ToolDevPluginLIB.dll** first.

- Open Visual Studio and go to **Tools → ILDasm**
  - **error or not there?** Then go to Tools → External Tools... and:
  - select (or Add) "ILDasm"
  - The command should contain the path to ildasm.exe, but often it is wrong. Search it via windows explorer (usually in program files x86) and copy-paste the full path including exe.

- In ILDasm, choose “Open” and browse to the given **ToolDevPluginLIB.dll** to open it.
- **Explore the library:**
    - What is in there? Classes, properties, methods,... ?
    - Take your time to explore everything! Make sure you also **double-click to open** them.
    - What do you think the code in PrintStats() is doing? Why does it contain so much more code than the Die() method?
- Now also explore **the given plugin dll** files. *Sidenote: you can also do this with .exe files!*

PluginLoader

Since we do not know which characters are available through the plugins beforehand, we cannot create instances at compile time.
*Somehow the **plugins** should be and processed **loaded dynamically / at runtime**. By adding / removing a dll from the folder, we can very easily add/remove available characters. This is a typical case where we can use **REFLECTION**.*

*This project will **process** the available plugins in a folder **at runtime** to search for character types, using **reflection**. Furthermore, it will use reflection to **create an instance of these objects at runtime.** These instances are then added to a list of characters so they can participate in the game.*

- Open the given project **PluginLoader**
- A lot of code is already given. It is your job to add the missing parts by finding and resolving all the **TODO’s** marked in the project.
    - If you did it correctly, the project should start by giving an overview of all characters (see screenshot).
    - Then they should start fighting each other with an interval of 2 seconds, until finally one character is the winner. *Warning: this quick test program is not properly testing a bunch of things; it might get stuck but you do not have to worry about that.*
    - If you move one of the dll’s from the folder and restart the project, the characters in this dll will no longer be in the game.

Create/load your own plugin

- Create a new **Class Library (.NET framework)** project
- Try to create one or more characters that would be recognized by the plugin loader
    - Note sure how? Use ILDasm to explore the plugin dll files again as an example.
- Build the project, get the generated .dll file from the bin folder and place it with the others in the plugin folder
- Run the PluginLoader project again; your new character should appear and participate in the fight!
- ***Warning:** this quick test program is not properly testing a bunch of things; it might get stuck based on the health and attack you chose in combination with the others for example, but you do not have to worry about that.*

Reflect

Take a moment to **reflect** on what you just did and how **reflection** (😉) was part in all this!

(No need to upload to GitHub; this is just a small theory thing.)