|  |  |
| --- | --- |
| **SOFTWARE DESIGN PATTERNS**  Diploma in IT  Year 2 (2024/25) Semester 4 | Week **8** |
| **1** hour |
| **Tutorial 8 – Composite** | |

**OBJECTIVES**

* Understand the Composite design pattern

**ACTIVITY**

Recall Week 8 Tutorial:

In a multiplayer squad-based first-person shooter (FPS) game, different teams of players characters (PCs) or non-player characters (NPCs) battle against each other in a combat arena. PC professions include Warrior and Wizard, and NPC professions include Orcs and Goblins. Each team consists of one or more players PCs or NPCs or both; for example, a team may consist of a Warrior, a Wizard and 2 Goblins. Furthermore, multiple teams may form an alliance to form a combined team.

The following are a set of classes for the game. For simplicity, we represent the class of a PC or NPC using a string (e.g., “Warrior”, “Orc”, etc.).

A white box with black text

Description automatically generated A black and white rectangular sign with black text

Description automatically generated A white box with black text

Description automatically generated A close-up of a label

Description automatically generated A white rectangular object with black text

Description automatically generated

A diagram of a computer network

Description automatically generated

Implement the Composite design pattern such that running the following program will produce the desired output.

**Program:**

GameEntity hero1 = new PC("Warren", "Warrior");

GameEntity hero2 = new PC("Wendy", "Wizard");

GameEntity mob1 = new NPC("Oscar", "Orc");

GameEntity mob2 = new NPC("Gary", "Goblin");

GameEntity arenaTeams = new Team("Arena");

GameEntity team1 = new Team("Awesome");

team1.add(hero1);

GameEntity team2 = new Team("Killers");

team2.add(hero2);

team2.add(mob1);

GameEntity team3 = new Team("Lonely");

team3.add(mob2);

arenaTeams.add(team1);

arenaTeams.add(team2);

arenaTeams.add(team3);

Arena arena = new Arena(arenaTeams);

Console.WriteLine("--- Current Arena Teams ---");

arena.print();

Console.WriteLine("---------------------------\n");

arena.removeEntity(team2);

team3.add(team2);

Console.WriteLine("--- New Arena Teams ---");

arena.print();

Console.WriteLine("-----------------------");

**Output:**

Warren the Warrior joins Awesome!

Wendy the Wizard joins Killers!

Oscar the Orc joins Killers!

Gary the Goblin joins Lonely!

Team Awesome joins Arena!

Team Killers joins Arena!

Team Lonely joins Arena!

--- Current Arena Teams ---

Arena:

Awesome:

Warren the Warrior

Killers:

Wendy the Wizard

Oscar the Orc

Lonely:

Gary the Goblin

---------------------------

Team Killers leaves Arena!

Team Killers joins Lonely!

--- New Arena Teams ---

Arena:

Awesome:

Warren the Warrior

Lonely:

Gary the Goblin

Killers:

Wendy the Wizard

Oscar the Orc

-----------------------

Notes:

* In the output, it is not clear that Team Lonely contains Team Killers. One way to solve this issue is to add a new attribute that reflects the “depth” of the GameEntity such that you can add a number of spaces proportional to the depth when printing. You can try to implement this on your own.
* So far, we have not yet used the getChild() method. This is used when you need to retrieve a specific Component.
* We have not implemented a composite iterator in this practical. You may wish to try doing this on your own as well.