## Game Development Framework

Create a scenario that models a **Game Framework** using ***interfaces*** and ***abstraction***.

Guidelines:

- Create an interface defining the basic functionalities of a **game entity** – should contains property **Name** and **Update()** method.

- Then create an abstract class **implementing** common functionality for **game entity**. Add a ***constructor*** that receives ***name*** as parameter.

- Create two classes – **Player** and **Enemy** that represent a specific game entity and implement abstraction already defined. For class *Player* add **Health** property(getter and private setter) and **TakeDamage(**{int damage}**)** – reduced player health with given *damage*. For class *Enemy* add **AttackPower** property(only getter) and **Attack(**{player}**)** method that caused damage to the player(call player.TakeDamage(attackPower)).

- For Game Framework testing create a class **GameWorld** that contains a collection of game **entities**(List<?>) with possibility to add entity by **AddEntity(***{entity}***)** and updates each entity in the collection by **UpdateAllEntities()** method.

At the end code bellow should work properly:

```
class Program
{
    static void Main(string[] args)
    {
        // Creating game entities: player and enemy
        Player player = new Player("Player1");
        Enemy enemy = new Enemy("Enemy1", 10);

        // Creating a game world and adding entities to it
        GameWorld gameWorld = new GameWorld();
        gameWorld.AddEntity(player);
        gameWorld.AddEntity(enemy);

        // Updating all entities in the game world
        gameWorld.UpdateAllEntities();

        Console.ReadKey();
    }
}
```