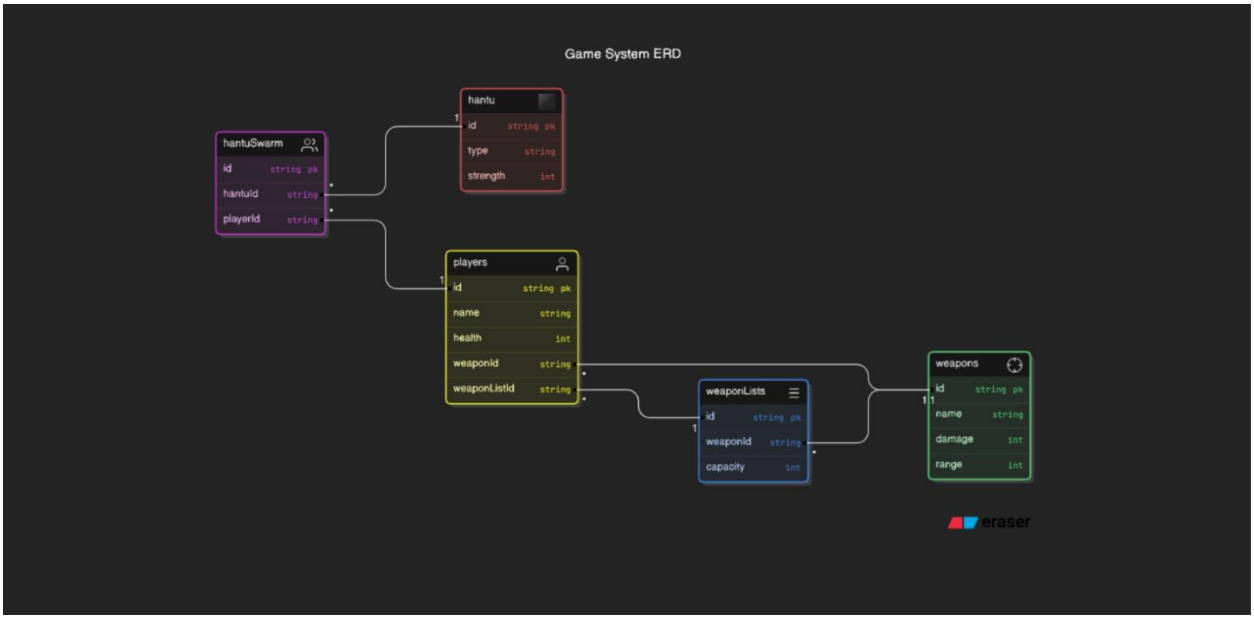
OOP MIKHAIL_22003781@utp.edu.my

CLASS DIAGRAM FOR HANTU, HANTUSWARM,PLAYER,WEAPONLIST AND WEAPONS.



HANTU CLASS CODE

```
using System;

5 references
public class Hantu
{
    private string _id;
    4 references
    public string Id
    {
        get { return _id; }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Id cannot be null or empty.");
            _id = value;
        }
    }

    private string _type;
    5 references
    public string Type
    {
        get { return _type; }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Type cannot be null or empty.");
            _type = value;
        }
    }

    private int _strength;
    10 references
    public int Strength
    {
        get { return _strength; }
        set
```

```
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Type cannot be null or empty.");
            _type = value;
        }
    }

    private int _strength;
    10 references
    public int Strength
    {
        get { return _strength; }
        set
        {
            if (value < 0)
                throw new ArgumentException("Strength cannot be negative.");
            _strength = value;
        }
    }

    0 references
    public Hantu(string id, string type, int strength)
    {
        Id = id;
        Type = type;
        Strength = strength;
    }

    0 references
    public void DisplayInfo()
    {
        Console.WriteLine($"Hantu ID: {Id}, Type: {Type}, Strength: {Strength}");
    }

    1 reference
    public void Attack(Player player)
    {
        if (Strength > 0)
```

```
public class Player
{
    class Hantu.Player
    3 references
    public string Name { get; set; }
    3 references
    public int Health { get; set; }

    0 references
    public Player(string name, int health)
    {
        Name = name;
        Health = health;
    }

    1 reference
    public void TakeDamage(int damage)
    {
        Health -= damage;
        Console.WriteLine($"{Name} took {damage} damage. Remaining health: {Health}");
    }
}
```

PLAYER CLASS CODE

```
public class Player
{

    private string _id;
    public string Id
    {
        get { return _id; }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Id cannot be null or empty.");
            _id = value;
        }
    }

    public string Name { get; set; }
    public int Health { get; set; }
    public Weapon CurrentWeapon { get; set; }
    public WeaponList PlayerWeapons { get; set; }


    public Player(string id, string name, int health)
    {
        Id = id;
        Name = name;
        Health = health;
        PlayerWeapons = new WeaponList();
    }


    public void Attack(Hantu hantu)
    {
        if (CurrentWeapon != null)
        {
            Console.WriteLine($"{Name} attacks Hantu with {CurrentWeapon.Name}");
            CurrentWeapon.Fire();
```

```
    public void Attack(Hantu hantu)
    {
        if (CurrentWeapon != null)
        {
            Console.WriteLine($"{Name} attacks Hantu with {CurrentWeapon.Name}");
            CurrentWeapon.Fire();
            hantu.TakeDamage(CurrentWeapon.Damage);
        }
        else
        {
            Console.WriteLine($"{Name} has no weapon equipped to attack.");
        }
    }


    public void TakeDamage(int damage)
    {
        Health -= damage;
        if (Health < 0)
            Health = 0;
        Console.WriteLine($"{Name} took {damage} damage. Remaining health: {Health}");
    }
```

Weapon Class code

```
using System;

5 references
public class Weapon
{

    1 reference
    public string Id { get; set; }
    3 references
    public string Name { get; set; }
    2 references
    public int Damage { get; set; }
    2 references
    public int Range { get; set; }


    0 references
    public Weapon(string id, string name, int damage, int range)
    {
        Id = id;
        Name = name;
        Damage = damage;
        Range = range;
    }

    0 references
    public void Fire()
    {
        Console.WriteLine($"{Name} fired with {Damage} damage and {Range} range.");
    }
}
```

WeaponList Class code

```
using System;

1 reference
public class WeaponList
{

    private List<Weapon> _weapons;

    0 references
    public WeaponList()
    {
        _weapons = new List<Weapon>();
    }

    0 references
    public void AddWeapon(Weapon weapon)
    {
        _weapons.Add(weapon);
        Console.WriteLine($"Weapon {weapon.Name} added to weapon list.");
    }

    0 references
    public Weapon SwitchWeapon(int index)
    {
        if (index < 0 || index >= _weapons.Count)
        {
            Console.WriteLine("Invalid weapon index.");
            return null;
        }

        return _weapons[index];
    }
}
```

HANTU SWARM CLASS CODE

```
public class HantuSwarm
{
    private List<Hantu> _hantuSwarm;

    0 references
    public HantuSwarm()
    {
        _hantuSwarm = new List<Hantu>();
    }


    0 references
    public void AddHantu(Hantu hantu)
    {
        _hantuSwarm.Add(hantu);
        Console.WriteLine($"Hantu {hantu.Id} added to the swarm.");
    }


    0 references
    public void AttackPlayer(Player player)
    {
        foreach (var hantu in _hantuSwarm)
        {
            if (hantu.IsAlive())
            {
                hantu.Attack(player);
            }
        }
    }

    0 references
    public void RemoveHantu(Hantu hantu)
    {
        _hantuSwarm.Remove(hantu);
        Console.WriteLine($"Hantu {hantu.Id} removed from the swarm.");
    }
}
```