```
using UnityEngine;
using System.Collections;

public class EnemyStats : MonoBehaviour
{
    // Public
    public NavMeshAgent agent;
    public GameObject target;
    public GameObject healthPowerup;

    public int health;
    public int damage;
    public int PowerPoints;
    public int healthDropChance;

    public float speed;
    public float attackSpeed;
    public float attackRange;
    public float hearingRange;
    public float alertPhaseTimer;
    public float blindChaseTimer;

    public enum State
    {
        Idle,
        Searching,
        Chasing,
        Alert,
        Attacking
    }

    public State curState;

    // Private
    private int _staticHealth;
    private int _healthDropRollOne;
    private int _healthDropRollTwo;

    private float m_playerDist;
    private float m_staticAlertPhaseTimer;
    private float m_staticBlindChaseTimer;
    private float m_staticAttackSpeed;

    private bool m_inRayLos = false;
    private bool m_inVisionCone = false;
    private bool m_playerVisible = false;
    private bool m_inPlayerRange = false;
    private bool m_blindChase = false;
    private bool m_recentlyChased = false;
    private bool m_alerted = false;
    private bool m_inPlayerFiring = false;

    private Vector3 m_forward;
    private Vector3 m_playerPos;
    private Vector3 m_playerDotPos;

    // Use this for initialization
    void Start ()
    {
        agent = GetComponent<NavMeshAgent> ();
        target = GameObject.Find ("Player");
        curState = State.Idle;
        _staticHealth = health;
        m_staticAlertPhaseTimer = alertPhaseTimer;
        m_staticBlindChaseTimer = blindChaseTimer;
        m_staticAttackSpeed = attackSpeed;

        agent.speed = speed;
        agent.stoppingDistance = attackRange - 0.5f;
    }

    // Update is called once per frame
    void Update ()
    {
        CheckLife ();
        CheckState ();
        EnemyBehavior ();
        if(!target.GetComponent<p_Stats>().dead)
        {
            LineOfSight ();
            ListenForPlayer ();
        }
    }

    void LineOfSight ()
    {
        m_forward = transform.TransformDirection (Vector3.forward);
        m_playerPos = target.transform.position - transform.position;
        m_playerDotPos = (target.transform.position - transform.position).normalized;
        m_playerDist = Vector3.Distance(transform.position, target.transform.position);

        RaycastHit LoSRay;

        if(Physics.Raycast (transform.position, m_playerPos, out LoSRay, 10))
        {
            if (LoSRay.collider.tag == "Player")
            {
                m_inRayLos = true;
            }
            else
            {
                m_inRayLos = false;
            }
        }

        if (Vector3.Dot (m_forward, m_playerDotPos) < 0.5f)
        {
            m_inVisionCone = false;
        }
        if (Vector3.Dot (m_forward, m_playerDotPos) >= 0.5f)
        {
            m_inVisionCone = true;
        }

        if (m_inVisionCone && m_inRayLos)
        {
            m_playerVisible = true;
            m_recentlyChased = true;
        }
        else
        {
            m_playerVisible = false;
        }
    }

    void EnemyBehavior ()
    {
        if (curState == State.Chasing)
        {
            //Debug.Log ("Chasing");
            ChasePlayer ();
        }
        if (curState == State.Attacking)
        {
            //Debug.Log ("Attacking");
            AttackPlayer();
        }
        if (curState == State.Searching)
        {
            //Debug.Log ("Searching");
            SearchForPlayer();
            BlindTimer ();
        }
        if (curState == State.Alert)
        {
            //Debug.Log ("Alert");
            AlertTimer ();
        }
        if (curState == State.Idle)
        {
            //Debug.Log ("Idle");
        }
        if (curState != State.Attacking)
        {
            attackSpeed = m_staticAttackSpeed;
        }
    }

    void CheckState ()
    {
        if (m_playerVisible && m_playerDist > attackRange)
        {
            resetTimers();
            curState = State.Chasing;
        }
        else if (m_playerVisible && m_playerDist <= attackRange)
        {
            curState = State.Attacking;
        }

        if (!m_playerVisible && m_recentlyChased)
        {
            curState = State.Searching;
            m_recentlyChased = false;
        }
    }

    void AlertTimer ()
    {
        alertPhaseTimer -= Time.deltaTime;
        if (alertPhaseTimer <= 0)
        {
            curState = State.Idle;
            alertPhaseTimer = m_staticAlertPhaseTimer;
        }
    }

    void BlindTimer ()
    {
        blindChaseTimer -= Time.deltaTime;
        if (blindChaseTimer <= 0)
        {
            curState = State.Alert;
            agent.Stop();
            blindChaseTimer = m_staticBlindChaseTimer;
        }
    }

    void resetTimers()
    {
        alertPhaseTimer = m_staticAlertPhaseTimer;
        blindChaseTimer = m_staticBlindChaseTimer;
    }

    void ChasePlayer ()
    {
        if(gameObject.activeSelf)
        {
            agent.SetDestination(target.transform.position);
        }
    }

    void AttackPlayer()
    {
        transform.LookAt(target.transform.position);
        attackSpeed -= Time.deltaTime;

        if(attackSpeed <= 0)
        {
            target.GetComponent<p_Stats>().health -= damage;
            attackSpeed = m_staticAttackSpeed;
        }
    }

    void SearchForPlayer()
    {
        if(gameObject.activeSelf)
        {
            agent.SetDestination(target.transform.position);
        }
    }

    public void CheckLife ()
    {
        if (health <= 0)
        {
            target.GetComponent<p_Stats>().IncreaseCounter(PowerPoints);
            RollForHealthDrop();
            gameObject.SetActive (false);
            //Destroy(gameObject);
        }
    }

    void ListenForPlayer()
    {
        if(Vector3.Distance(transform.position, target.transform.position) <= hearingRange && target.GetComponent<p_Stats>().p_Firing)
        {
            curState = State.Chasing;
        }
    }

    void RollForHealthDrop()
    {
        _healthDropRollOne = Random.Range(0, healthDropChance);
        _healthDropRollTwo = Random.Range(0, healthDropChance);

        if(_healthDropRollOne == _healthDropRollTwo)
        {
            GameObject healthDrop = Instantiate(healthPowerup, transform.position, transform.rotation) as GameObject;
        }

    }

    public void Reset()
    {
        health = _staticHealth;
        m_inRayLos = false;
        m_inVisionCone = false;
        m_playerVisible = false;
        m_inPlayerRange = false;
        m_blindChase = false;
        m_recentlyChased = false;
        m_alerted = false;

        curState = State.Idle;
    }
}
```