```
/// <summary>
/// Rotates the enemy smoothly towards the player for attack alignment
/// </summary>
private void rotation_the_enemy_to_player()
{
    Vector3 direction = thePlayer.transform.position - transform.position;
    direction.y = 0;
    if (direction != Vector3.zero)
    {
        Quaternion targetRotation = Quaternion.LookRotation(direction);
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, rotationSpeed *
Time.deltaTime);
    }
}

/// <summary>
/// Handles enemy attack sequence triggering animations and colliders
/// </summary>
private void start_attack()
{
    animator.SetBool("run", false);
    ai.speed = 0;
    enemy_do_attack_now = true;
    animator.SetTrigger("attack" + select_attack);
    StartCoroutine(enable_Collider_for_attack(time_for_startAttack[select_attack - 1]));
    StartCoroutine(disable_Collider_for_attack(time_for_finishAttack[select_attack - 1]));
    StartCoroutine(finish_the_attack(time_for_finishAttack[select_attack - 1]));
}

/// <summary>
/// Enables the enemy attack collider and plays attack sound after a delay
/// </summary>
IEnumerator enable_Collider_for_attack(float cooldown)
{
    yield return new WaitForSeconds(cooldown);
    statsPlayer_script.generate_voidSound(transform, sword_attack_clip, 0.5f);
    capsule_for_attack.enabled = true;
}

/// <summary>
/// Disables the enemy attack collider after a delay
/// </summary>
IEnumerator disable_Collider_for_attack(float cooldown)
{
    yield return new WaitForSeconds(cooldown);
    capsule_for_attack.enabled = false;
}

/// <summary>
/// Ends the enemy attack state and resets flags after attack duration
/// </summary>
IEnumerator finish_the_attack(float cooldown)
{
    yield return new WaitForSeconds(cooldown + 1);
    enemy_do_attack_now = false;
    setRandom = false;
}

/// <summary>
/// Detects player staying in attack range
/// </summary>
private void OnTriggerStay(Collider other)
{
    if (other.gameObject.CompareTag("Player"))
        the_player_is_in_attack = true;
}

/// <summary>
/// Detects player leaving attack range
/// </summary>
private void OnTriggerExit(Collider other)
{
    if (other.gameObject.CompareTag("Player"))
        the_player_is_in_attack = false;
}

/// <summary>
/// Applies damage to the enemy, triggers blood effect and sound
/// </summary>
public void getDMG(float getDMG)
{
    hp_Enemy -= getDMG;
    statsPlayer_script.generate_voidSound(transform, get_dmg_clip, 0.5f);
    void_blood_effect();
    GameObject thePlayer = GameObject.FindGameObjectWithTag("Player");
    inventoryPlayer inventoryPlayer_ = thePlayer.GetComponent<inventoryPlayer>();
    inventoryPlayer_.sword_echipe_now.boxCollider.enabled = false;
}

/// <summary>
/// Spawns a blood visual effect at the enemy hit location
/// </summary>
private void void_blood_effect()
{
    GameObject blood = Instantiate(blood_FX);
    blood.transform.SetParent(locationFX.transform);
    blood.transform.localPosition = Vector3.zero;
    blood.transform.localRotation = Quaternion.identity;
    Destroy(blood, 3);
}

/// <summary>
/// Handles enemy death, disables AI, colliders, animations and grants XP
/// </summary>
void death()
{
    if (!theEnemyDefeat)
    {
        Destroy(trigger_for_attack_script.gameObject);
        capsuleCollider.enabled = false;
        animator.SetBool("death", true);
        ai.acceleration = 0;
        ai.angularSpeed = 0;
        ai.speed = 0;
        hp_Slider.gameObject.SetActive(false);
        xp_for_player();
        theEnemyDefeat = true;
        ai.enabled = false;
        theCollider_of_Enemy.SetActive(false);
    }
}

/// <summary>
/// Gives experience points to the player when the enemy is defeated
/// </summary>
void xp_for_player()
{
    if (!giveXP)
    {
        float multimply = 5;
        statsPlayer_script.update_XP(((dmg * multimply) + (hp_Enemy / multimply)) * 2.5f);
        giveXP = true;
    }
}

/// <summary>
/// Coroutine to remove enemy from player's battle count after delay
/// </summary>
IEnumerator substract_the_enemy_from_list_of_statsPlyer()
{
    was_added_enemy_in_list_of_statPlayer = false;
    yield return new WaitForSeconds(4);
    statsPlayer_script.numbers_of_enemy_is_in_battle_with_player--;
}

/// <summary>
/// Deals damage to the player during enemy attack
/// </summary>
public void attack()
{
    statsPlayer_script.takeDMG(dmg * 0.5f);
    giveDMG = true;
}
```