```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System;
using UnityEngine;

public class CritterBehaviourController : MonoBehaviour {

    [Header("References")]
    public CritterBase critterBaseRef;
    public NavigationControllerBase navigationControllerRef;

    [Header("No behaviour")]
    public bool noBehaviourFound;
    public bool waitingAfterBehaviour;
    public float noBehaviourWaitTimer = 5; //Will wait to try and check again
    public float afterBehaviourWaitTimer;
    public float currentWaitTimer;

    [Header("Active Behaviour")]
    public BehaviourBase activeBehaviour; //The behaviour that is currently Executed
    BehaviourBase nullBehaviour;

    [Header("List of all Behaviours")]
    public BehaviourBase[] behaviours; //An ordered list of behaviours


    List<BehaviourBase> behaviourInitialList = new List<BehaviourBase>(); //Here genes add new behaviours




    public void Initiate(CritterBase critterbase, NavigationControllerBase navigationController) {
        critterBaseRef = critterbase;
        navigationControllerRef = navigationController;

        SetReferences();
        SetupBehaviours();
```

```
    }

    void SetupBehaviours() {
        //Order behaviours from lowest to highest and convert it to behaviours array.
        if (behaviourInitialList.Count <= 0) {
            Debug.LogError("(" + gameObject.name + ") No behaviours Found and thus CritterControllerBase is removed");
            Destroy(this);
        }

        //List to proper Behaviour Array for faster calculation
        behaviours = behaviourInitialList.ToArray();

        //Initialize behaviours
        for (int i = 0; i < behaviours.Length; i++) {
            behaviours[i].InitializeBehaviour(critterBaseRef);
        }

        //Reorder Array from highest to lowest priority
        Array.Sort(behaviours, delegate (BehaviourBase x, BehaviourBase y) { return
y.behaviourPriority.CompareTo(x.behaviourPriority); });

        nullBehaviour = new BehaviourBase();
        nullBehaviour.name = "NULL";
        nullBehaviour.isValid = false;

        activeBehaviour = nullBehaviour;
    }

    void Update() {
        //Update behaviour if has one
        UpdateActiveBehaviour();
    }


    //Find an behaviour if has none. Updates behaviour logic until behaviour is no longer valid.
    void UpdateActiveBehaviour() {
        //if (activeBehaviour.name == "NULL")
        //    Debug.Log("There is no active behaviour");
```

```
        if (critterBaseRef.status_canExecuteBehaviours) {
            //Update active behaviour
            if (activeBehaviour.name != "NULL" && activeBehaviour.isValid) {
                activeBehaviour.BehaviourUpdate();
            }

            // Wait time between behaviours
            else if (waitingAfterBehaviour) {
                currentWaitTimer += Global.customGameTime;

                if (currentWaitTimer >= afterBehaviourWaitTimer) {
                    waitingAfterBehaviour = false;
                    currentWaitTimer = 0;
                }
            }

            //Find new behaviour
            else if (activeBehaviour.name == "NULL") {
                //Debug.Log("(" + gameObject.name + ") Active behaviour: '" + activeBehaviour.name + "' is no longer valid and
a new behaviour will be picked.");
                GetNewBehaviour();
            }

            //Active behaviour is no longer valid
            else if (!activeBehaviour.isValid && !waitingAfterBehaviour) {
                waitingAfterBehaviour = true;
                activeBehaviour = nullBehaviour;
            }
            else {
                Debug.LogError("(" + gameObject.name + ") This should never be called!");
            }
        }
        //character was moving But behaviours are disabled so disable the movement
        else if (activeBehaviour.name != "NULL" || waitingAfterBehaviour) {
            ResetBehaviours();
        }
    }
```

```
void ResetBehaviours() {
    if (navigationControllerRef.isMoving)
        navigationControllerRef.ClearMovement();

    activeBehaviour = nullBehaviour;
    waitingAfterBehaviour = false;
    currentWaitTimer = 0;
}


//Gets a new behaviour from list
void GetNewBehaviour() {
    if (!noBehaviourFound) {
        for (int i = 0; i < behaviours.Length; i++) {
            if (behaviours[i].CheckIfValid()) {
                activeBehaviour = behaviours[i];
                //Debug.Log("(" + gameObject.name + ") A new active behaviour is set: '" + activeBehaviour.name + "'");
                return;
            }
        }

        //Could not find a behaviour so wait x sec to try again
    } else {
        if (currentWaitTimer < noBehaviourWaitTimer) {
            currentWaitTimer += Global.customGameTime;

            if (currentWaitTimer >= noBehaviourWaitTimer) {
                currentWaitTimer = 0;
                noBehaviourFound = false;
            }
        }
        return;
    }

    noBehaviourFound = true;
    Debug.Log("(" + gameObject.name + ") No Behaviours are valid to execute. Will wait a bit before checking again.");
}
```

```
    //Set WaitTimes
    public void SetAfterBehaviourWaitTime(Vector2 minMaxWaitTime){
        afterBehaviourWaitTimer = UnityEngine.Random.Range(minMaxWaitTime.x, minMaxWaitTime.y);
    }

    public void SetAfterBehaviourWaitTime(float waitTime){
        afterBehaviourWaitTimer = waitTime;
    }


    //A Gene can call this to add a behaviour
    public void GeneAddBehaviour(BehaviourBase behaviour) {
        behaviourInitialList.Add(behaviour);
    }


    //==============================OTHER FUNCTIONS==============================\\
    void SetReferences() {
        if (critterBaseRef == null)
            Debug.LogError("(" + gameObject.name + ") Has no critter base for some reason!");
        if (navigationControllerRef == null)
            Debug.LogError("(" + gameObject.name + ") Has no navigation Controller for some reason!");
    }
}
```