```
//Telling what Libraries we can get Components from.
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VillagerMovement : MonoBehaviour {
    //Define a bunch of variables.
    public float moveSpeed;
    private Vector2 minWalkPoint;
    private Vector2 maxWalkPoint;

    private Rigidbody2D myRigidBody;

    public bool isWalking;

    public float walkTime;
    private float walkCounter;
    public float waitTime;
    private float waitCounter;

    private int WalkDirection;

    public Collider2D walkZone;
    private bool hasWalkZone;

    public bool canMove;


    // Use this for initialization
    void Start () {
        //villager bodies are now defined as RigidBody2D
        myRigidBody = GetComponent<Rigidbody2D>();


        walkCounter = walkTime;
        waitCounter = waitTime;

        //once game chooses the direction
        ChooseDirection();

        if (walkZone != null)
        {
            //setting bounds for where the villagers can move in with a collider
              acting as trigger to
            //show where they can go.
            minWalkPoint = walkZone.bounds.min;
            maxWalkPoint = walkZone.bounds.max;
            hasWalkZone = true;
        }

        canMove = true;
    }
```

```csharp

    // Update is called once per frame
    void Update () {


        {
            canMove = true;
        }

        if(!canMove)
        {
            myRigidBody.velocity = Vector2.zero;
            return;
        }
        if(isWalking)
        {
            walkCounter -= Time.deltaTime;

            //What this script below is doing is in the random cases decribed below, each one defines a specific
            //movement for the villager. When the Villager hits the BoxCollider2D set as a trigger, checks where it
            //is. if the villager is outside of the box it moves it in, and if it is inside it stays inside. It is
            //continously checking on whether or not the villager is in, so it the villager will always stop before
            //the villager goes outside. This is helpful for making sure villager accidently don't trip the warp
            //feature described below.

            switch(WalkDirection)
            {
            case 0:
                myRigidBody.velocity = new Vector2(0, moveSpeed);
                    if(hasWalkZone && transform.position.y > maxWalkPoint.y)
                    {
                        isWalking = false;
                        waitCounter = waitTime;
                    }

                break;

            case 1:
                myRigidBody.velocity = new Vector2(moveSpeed, 0);
                    if (hasWalkZone && transform.position.x > maxWalkPoint.x)
                    {
                        isWalking = false;
                        waitCounter = waitTime;
                    }
                    break;
            case 2:
                myRigidBody.velocity = new Vector2(0, -moveSpeed);
```

```
                    if (hasWalkZone && transform.position.y < minWalkPoint.y)
                    {
                        isWalking = false;
                        waitCounter = waitTime;
                    }
                    break;

                case 3:
                    myRigidBody.velocity = new Vector2(-moveSpeed, 0);
                    if (hasWalkZone && transform.position.x < minWalkPoint.x)
                    {
                        isWalking = false;
                        waitCounter = waitTime;
                    }
                    break;
            }

            if (walkCounter < 0)
            {
                isWalking = false;
                waitCounter = waitTime;
            }

        }
        else
        {
            waitCounter -= Time.deltaTime;

            myRigidBody.velocity = Vector2.zero;

            if(waitCounter < 0)
            {
                ChooseDirection();
            }
        }
    }

    public void ChooseDirection()
    {
        //Essentially RNG with directions 0=up 1=right 2=down 3=left; range
          doesn't include 4
        WalkDirection = Random.Range(0, 4);
        isWalking = true;
        walkCounter = walkTime;

    }

}
```