***// each of these import declarations should be commented –***

***// - what are we importing from each module?***

***// what do each of the imported functions do, in one phrase?***

***//***

using MongoDB.Bson.Serialization.Attributes;

using System.Collections.Generic;

***// the following seems straightforward***

***//***

namespace CPH\_IVT.Models

{

/// <summary>

/// Represents a U.S. county.

/// </summary>

[BsonIgnoreExtraElements]

public class County

{

/// <summary>

/// Length of all government-standard state FIPS codes.

/// </summary>

private const int REQUIRED\_STATE\_FIPS\_LENGTH = 2;

/// <summary>

/// Length of all government-standard county FIPS codes.

/// </summary>

private const int REQUIRED\_COUNTY\_FIPS\_LENGTH = 3;

/// <summary>

/// Unique identifier for a <see cref="County"/> object, composed of a

/// <see cref="State.FIPS"/> and <see cref="CountyFIPS"/>.

/// </summary>

public string CountyId { get; set; }

/// <summary>

/// Government-assigned county identification code.

/// </summary>

private string \_countyFIPS = string.Empty;

/// <summary>

/// Government-assigned county identification code.

/// </summary>

public string CountyFIPS

{

get => \_countyFIPS;

set => \_countyFIPS = Pad(value, value.Length, REQUIRED\_COUNTY\_FIPS\_LENGTH);

}

/// <summary>

/// Full name.

/// </summary>

public string Name { get; set; }

/// <summary>

/// Collection of health indicators.

/// </summary>

/// <seealso cref="HealthIndicator"/>

public ICollection<HealthIndicator> Indicators { get; set; }

///// <summary>

///// <see cref="CensusRegion.Number"/>

///// </summary>

//public string CensusRegionNumber { get; set; }

///// <summary>

///// <see cref="CensusDivision.Number"/>

///// </summary>

//public string CensusDivisionNumber { get; set; }

/// <summary>

/// <see cref="State.FIPS"/>

/// </summary>

private string \_stateFIPS = string.Empty;

/// <summary>

/// <see cref="State.FIPS"/>

/// </summary>

public string StateFIPS

{

get => \_stateFIPS;

set => \_stateFIPS = Pad(value, value.Length, REQUIRED\_STATE\_FIPS\_LENGTH);

}

/// <summary>

/// Initializes a new instance of the <see cref="County"/> class.

/// </summary>

/// <param name="stateFIPS"><see cref="State.FIPS"/></param>

/// <param name="countyFIPS"><see cref="CountyFIPS"/></param>

public County(string stateFIPS, string countyFIPS)

{

StateFIPS = stateFIPS;

CountyFIPS = countyFIPS;

CountyId = string.Concat(StateFIPS, CountyFIPS);

}

/// <summary>

/// Inserts leading zeros to adhere to government standard of FIPS representation.

/// </summary>

/// <param name="unpaddedFIPS">Non-standard FIPS code</param>

/// <param name="stringLength">Length of <paramref name="unpaddedFIPS"/></param>

/// <param name="lengthToPad">Variable length of government standard FIPS code representation</param>

/// <returns>Government standard FIPS representation</returns>

/// <seealso cref="REQUIRED\_STATE\_FIPS\_LENGTH"/>

/// <seealso cref="REQUIRED\_COUNTY\_FIPS\_LENGTH"/>

private static string Pad(string unpaddedFIPS, int stringLength, int lengthToPad)

{

string paddedFIPS = new string(unpaddedFIPS);

for(int i = stringLength; i < lengthToPad; i++)

{

paddedFIPS = paddedFIPS.Insert(0, "0");

}

return paddedFIPS;

}

}

}