***// 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;

***// what is the “Ignore Extra Elements” pragma?***

***// why are we doing FIPS padding here, when there’s a FIPS padding routine in models/county.cs?***

***//***

namespace CPH\_IVT.Models

{

/// <summary>

/// Represents a U.S. state or territory.

/// </summary>

[BsonIgnoreExtraElements]

public class State

{

/// <summary>

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

/// </summary>

private const int REQUIRED\_FIPS\_LENGTH = 2;

/// <summary>

/// Government-assigned state identification code.

/// </summary>

private string \_fips = string.Empty;

/// <summary>

/// Government-assigned state identification code.

/// </summary>

public string FIPS

{

get => \_fips;

set => \_fips = Pad(value, value.Length);

}

/// <summary>

/// Full name.

/// </summary>

public string Name { get; set; }

/// <summary>

/// Two-character abbreviation of <see cref="Name"/>.

/// </summary>

public string Abbreviation { get; set; }

/// <summary>

/// Collection of U.S. counties.

/// </summary>

/// <seealso cref="County"/>

public ICollection<County> Counties { 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>

/// 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>

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

/// <seealso cref="FIPS"/>

private string Pad(string unpaddedFIPS, int stringLength)

{

string paddedFIPS = new string(unpaddedFIPS);

for (int i = stringLength; i < REQUIRED\_FIPS\_LENGTH; i++)

{

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

}

return paddedFIPS;

}

}

}