Task 4.1.3 Practice exercises - Working with 2-D arrays of objects

1. Following on from the car park space status monitoring program discussed in the Task 4.1.2 video, write a new program which incorporates the following changes.

Skeleton code will be provided for this task so that you only have to address the changes to the code which need to be made so that it works with a 2-D array of objects, as per the requirements outlined below.

a) Each car park can now have three different states: vacant, occupied or in service, so you should now design and implement a new class `SpaceInfo`, which represents the current status of a car park space, as follows:

- *Vacant* -> car park space is available for use and is unoccupied
- *Occupied* -> car park space is available for use and is currently occupied
- *In Service* -> car park space is not available for use. A car park space must be vacant before its status can be switched to In Service.

This `SpaceInfo` class should provide methods which allow the class user to update the current status to any one of the three states notes above as required, and also allow the current status of the car park space to be retrieved.

When a new `SpaceInfo` object is created, its status should be set to the *Vacant* state by default.

b) Update CarParkMonitoringSystem discussed in the podcast so that it now declares and instantiates a 2-D Array of elements of the newly implemented object type `SpaceInfo`, with the same dimensions set for the array in the original program.

Tip: Consider what the default value will be for the elements in the array and whether any further action may perhaps be required to fully initialise the array to represent the initial collection of vacant car par spaces.

c) Implement updated versions of the methods `spaceEntered()`, `spaceExited()` and `displayCarParkStatus()` that were discussed / demonstrated in the original program, so that they now work with the 2-D array of `SpaceInfo` objects we are now maintaining.

Note that there will now be a need to check to make sure that a space is not currently *In Service* before updating this status to vacant or occupied.

The `displayCarParkStatus()` method should also subtract the number of car park spaces that are currently in service when displaying the total number of spaces available and determining whether the car park is full or open.

d) Also implement new methods which allow a car park space to be placed into or returned from being *In Service* (a car park space must be *Vacant* before it can be put into service and should be reset to *Vacant* after it has been returned from being *In Service*).