COIS 2020H Assignment 1

For Brian Srivastava

[bsrivastava@trentu.ca](mailto:bsrivastava@trentu.ca)

# Submission guidelines:

Please create single Word/PDF document where you have a copy of your testing (screenshots are fine), and an explanation of theory answers. Put a copy of your source code as an appendix. Put all of your files, including the whole (visual studio project directory) into one big assignment directory with everything, and zip it up, and then name the file with your trent username.zip, which you upload to blackboard.   
  
While not explicitly for marks, your assignment should have readable spelling and grammar, your source code should have comments explaining what each class, and what each method is for, and your testing should cover the major points listed below. You should also have a brief explanation of how to run your program if it isn’t just a generic C# project in visual studio, and the program should be sufficiently functional that I can figure out how to use it. You should include some program output (in this case a console window) in your testing document as well, but it should cover major use cases.

# Primary Programming Question (5 marks)

**Main**

There is a more complete explanation of what your program should do towards the end.

Make a main program that creates a list and an array, of objects defined below. You will need to randomly generate many of the properties of these objects. There is a more complete description at the end of the class specification.

You can use existing List implementations in C# for now, (A2 I’ll make you write your own).

Random numbers can be generated using the code in Lab 1.

**General Overview**

All objects in our system are going to be on a grid. The grid (a square grid) starts at position 0,0 in the upper left corner of the screen. X increases to the right, Y increases in the downward direction. Each cell in the grid has an ID (starting at 0, incrementing to the right first, by row). (So the first square is from 0,0 to 10,10, with 0,0 as the top left corner). There will be a number of mobile objects, (Cats and Snakes), which will have a position, meaning they are in a specific cell on our grid.

Class **Position**

3 floats (x, y, z), while these can be any values you should generally avoid generating things well off screen.

Class **Cell**

Each cell has an integer ID.  
All cells have the same width and height (10.0 or 5.0 if that makes things easier for you)

Position (x, y)

(Advanced: List of Mobile objects at this location)

Class **Grid**

A grid is a 2D array of cells. You will need to text two different grid sizes (10x10 and 16x16).

The grid has a method GetCellID(position), this takes a position object, and returns the cell ID where it would be located. This needs to work with both grid sizes.

Base Class:

**MobileObject**

* Name
* ID
* Position (x,y,z) - is generated randomly as an x, y, z
* PositionCell this is the specific cell ID it is located on.
* All mobile objects have a method move(dx, dy,dz) which will change their position, and calculate a new cell location by changing their position by + dx, +dy + dz (which can be negative values)

ID should be unique, but it should be assigned to the object when it is created, your main program should have a list of Mobile Objects, and the ID’s should be created in sequence. Names should be loaded from the catnames and snakenames text files attached (notice the files are deliberately in different formats), you can use names in sequence or randomly.

First Sub class of **MobileObject**: Type **Cat.**

A Cat should have the following properties that must be present when the object is created:

Cats have

* a torso length (float > 0)
* a head length (float >0)
* a tail length (float >0)
* leg length (float >0)
* mass (float>0)
* A total length calculated as a private method using torso, head and leg properties.

Objects of type Cat should calculate their total length. The method which generates this should be private.

Second Subclass **MobileObject**: Type **Snake**

* Length (float > 0)
* radius (float >0)
* Mass (float > 0)
* Number of vertebrae (integer between 200 and 300)
* An estimated cylindrical volume (calculated as ) which should be a private property of the snake

Generate a number (5 each ) of objects type Cat and Snake and put them in a list of mobile objects.

# What Should A Minimally Functional Program Do (Checklist)?

* Create a collection of objects on the grid (test two different grid sizes, I suggest 10x10 and 16x16, but 5x5 and 8x8 work too if you prefer to make it easier to visualise).
* Generate a List of several Mobile objects of each type with random positions and properties.
* (Advanced: Make an Array of the same objects at the same time)
* Allow users to create a new object
* Object creation should fail if values entered are invalid, (e.g. a negative value for heights, or widths, positions can be negative).
* Allow a user to enter an object ID, and ‘move’ that object. (your program should not crash on an invalid ID)
* All the user to run a ‘MoveAndUpdate’ method, which randomly moves each object by some random (but small) dx, dy, and dz.
* Implement a have a ‘MoveOrigin’ method – this takes 3 float inputs (dx, dy, dz) and shifts every mobile object by (-dx, -dy, -dz)
* Implement a ‘FindWithinDistance’ method, that takes one parameter (a float) and prints off all object names and ID’s (and possibly their positions for testing) within that distance from the origin (distance is sqrt(x2+y2+z2) where x, y and z are the positions of each object.
* Print all of the MobileObjects and their properties

**Encapsulation (part of programming question).**

Public:

It should be possible to ‘print’ the status of an object, which either returns a string or prints to the console all of the properties of the object, meaning you need a method ‘print’ for each object type.

An object should have a series of **setters** and **getters** which allow you to set and return the various properties. Setters and getters are controversial in object oriented programming for the obvious reason that some of them do nothing but set a variable equal to some input value, on the other hand, used well, they prevent invalid or unworkable values being set (e.g. torso height = 10 000 is a perfectly valid integer but not so sensible if the other values are in the range 1-10).

Private: as mentioned above, total cat length and snake bounding volume should be calculated as a private method.

Protected and Internal: you do not need to use protected or internal methods for this question.

Don’t worry about extreme edge cases (like extremely large integer values, strings that can overflow or any of that sort of stuff).

# More Advanced Problem (2 marks)

Each cell should keep track of a list of all of the objects in its location, that list should be sorted by z value. (Note that if an object moves it needs to be removed from the list). I don’t care which sort you use.

You should be able to print the Grid, which will print each cell ID, and the number objects at that location.

# Programming mini challenge (1 mark)

Create a randomly ordered array from the existing array you have (that was presumably created in order by ID). Do this by making a new array and copying elements from the old into the new, but obviously, in a random order. (If you are looking for how to do this with an array you can google search answers but be careful, the Fisher-Yates, or “Knuth” Shuffle which are **NOT** what I want, they are an in place shuffle and I want you to create a new array in a random order).

Testing for this is tricky, since you don’t want to copy duplicates, and you don’t want to try and copy a non existent value. A linked list might be easier, but we’ll have to get to that later, that’s sort of the point: this is one of the weaknesses of arrays.

# Some theory questions: (2 marks total).

50 – 100 words each for answers, just think about the problems and give a general sense of how you’d deal with it.

1. What would go wrong if we could add and remove Mobile objects (e.g. spawn and destroy them) and they were all stored in an array, in order by ID?
2. For the grid, how would you change your implementation if the grid was no longer a regular pattern, and you couldn’t rely on math for fast calculations to determine which cell to move to (particularly between rows).
3. If you wanted to change the grid from squares to hexagons (or some other regular shape) what would you do?
4. One very common situation would be to try and prioritise some set of objects over another (e.g. ones closest to the player get processed first), how would you change the class design and the list which stores the mobile objects to support this?