SET08119 – Object Oriented Software Development
Assessment 1

Learning Outcomes Covered:	1,4,
Assessment Type:	Practical Skills Assessment
Overall module assessment	100% continuous
For this assessment:	20%
Submission Date:	20th October
Submission Time:	17:00
Submission Method:	Upload to Moodle
Module leader:	Neil Urquhart

- You are advised to keep a copy of your assessment solutions.
- Please note regulation Section B5.3.b regards component weighting.
- Late submissions will be penalised following the University guidelines as follows: Choose an item.
- Extensions to the submission date may only be given by the Module Leader for exceptional circumstances. – by submitting appropriate request form from Extenuating circumstances.
- Feedback on submissions will normally be provided within three working weeks from the submission date.

The University rules on Academic Integrity will apply to all submissions. The student academic integrity regulations contain a detailed definition of academic integrity breaches which includes use of commissioned material; knowingly permitting another student to copy all or part of his/her own work

You must not share your work with other students - this includes posting any of your work in any repository that is accessible to others (such as GitHub) and applies also after you have completed the course. You must not ask coursework-related questions in online forums (such as Stackoverflow).

You may use generative tools such as ChatGPT to generate C# boilerplate code but, you must place the prompt used within comments at the start of the class. You must also highlight those methods or properties that you have added or amended. Failure to highlight use of such tools will result in an academic misconduct referral.

By submitting the report, you are confirming that:

- It is your own work except where explicit reference is made to the contribution of others.

- It has not been submitted for any module or programme degree at Edinburgh Napier University or any other institution.

You should follow through the tasks outline in this document, pasting your answers into the appropriate sections. Once you have completed this workbook, you should upload it to Moodle.

Scenario:

You are required to design a class to hold details of a metro station, including the number of tickets sold. The station should have the following properties:

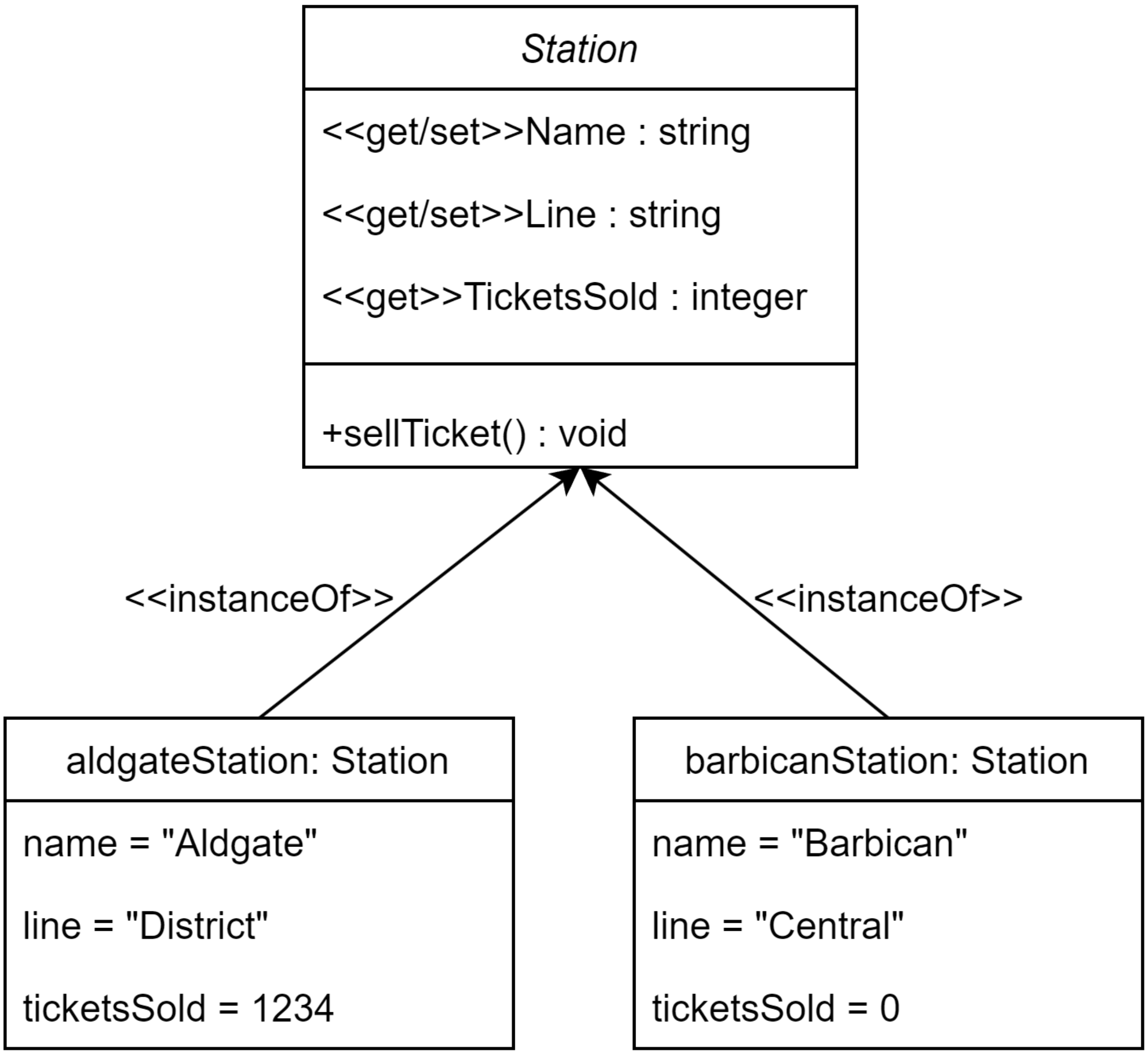
Property Name	**Examples**	**Validation**	**Public Accessors**
Name	“Aldgate” “Barbican”	Not “”	Get Set
Line	“District” “Central”	Not “”	Get Set
TicketsSold	1234	0 to 999999	Get

Examples of names and lines can be found at https://en.wikipedia.org/wiki/List_of_London_Underground_stations .

The Station class should have a public method sellTicket() which adds 1 to the TicketsSold property.

Task 1:

Draw a class diagram showing the Station class – you might want to use draw.io to produce the diagram, paste your diagram into the box below:



Marking criterion

Properties shown correctly:	/2
Method shown correctly:	/2
Diagram notation and presentation	/2

Task 2: Implement your Station class in c# and paste the source code below:

```
using System;

public class Station
{
    private string name;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }

    private string line;
    public string Line
    {
        get { return line; }
        set { line = value; }
    }

    private int ticketsSold;
    public int TicketsSold
    {
        get { return ticketsSold; }
        set { ticketsSold = value; }
    }

    public void SellTicket()
    {
        ticketsSold++;
    }
}
```

Marking criterion

Properties implemented correctly:	/2
Code commented adequately	/2
SellTicket() implementation	/2

Task 3 (Advanced)

Add a property 'ID' to your station class, which can be get and set.   The ID property takes the form of a number in the range 1000 to 2000.

Add the following code to your Station class:

```
...
private String _id;
public string ID   // property
{
get {
      return _id;
     }

set {
    if (value == ""){
      throw new ArgumentException("ID cannot be blank!","ID");
}
      _id = value;
 }
}
...
```

The following code can be used in a console application to test your Station class:

```
//Test for blank
try
{
    myStation.ID = "";
}
catch (Exception ex)
{
Console.WriteLine(ex.Message);
}
```

Modify the set() method of Station.ID to throw an error if the ID is not a number or is out of the 1000 ro 2000 range.

Modify your console app to test for a non-numeric ID, and an id that is out of range. If the ID is valid then an appropriate message should be displayed.

Paste your updated Station.ID.set code below:

```
using System;

public class Station
{
    private string name;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }

    private string line;
    public string Line
    {
        get { return line; }
        set { line = value; }
    }

    private int ticketsSold;
    public int TicketsSold
    {
        get { return ticketsSold; }
        set { ticketsSold = value; }
    }

    public void SellTicket()
    {
        ticketsSold++;
    }

    private String _id;
    public string ID   // property
    {
        get
        {
            return _id;
        }

        set
        {
            if (value == null)
            {
                throw new ArgumentException("ID cannot be null!", "ID");
            }

            /* i used TryParse because it doesn't give
             * errors and validates the id.
             */

            if (int.TryParse(value, out int _idValue))
            {
                if (_idValue >= 1000 && _idValue <= 2000)
                {
                    // if its within the range then it sets it to the id
                    _id = value;
                }
                else
                {
                    throw new ArgumentException("ID needs to be between 1000 and 2000.", "ID");
                }
            }
            else
            {
                throw new ArgumentException("ID needs to be a value.", "ID");
            }
        }
    }
}
```

Paste your main method (console) here:

```
class Program
{
    static void Main()
    {
        // add info for each station
        Station aldgate = new Station();
        aldgate.Name = "Aldgate";
        aldgate.Line = "District";
        aldgate.TicketsSold = 1234;
        aldgate.SellTicket();

        Station barbican = new Station();
        barbican.Name = "Barbican";
        barbican.Line = "Central";
        barbican.TicketsSold = 0;
        barbican.SellTicket();

        // output results
        Console.WriteLine(aldgate.Name + " has sold " + aldgate.TicketsSold);
        Console.WriteLine(barbican.Name + " has sold "+ barbican.TicketsSold);
        Console.ReadLine();

        Station myStation = new Station();

        //Test for blank
        try
        {
            myStation.ID = "hello";
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }

        try
        {
            myStation.ID = "999";
        }
        catch(Exception ex)
        {
            Console.WriteLine (ex.Message);
        }

        try
        {
            myStation.ID = "1001";
            Console.WriteLine(myStation.ID + " is a correct id!");
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        Console.ReadLine();
    }
}
```

Paste the output from your console app here

```
Aldgate has sold 1235

Barbican has sold 1



ID needs to be a value. (Parameter 'ID')

ID needs to be between 1000 and 2000. (Parameter 'ID')

1001 is a correct id!
```

Marking criterion

Test for non-numeric correctly:	/2
Test for range correctly	/2
Displays message for valid id	/2

Submission:

You should upload a .PDF version of this document into Moodle, following the instructions given on the site.

Help and Support

The coursework will be discussed in the lectures with opportunities to ask questions. You can also discuss your work in the practical sessions and get feedback on your work.

As well as the support available in this module, programming surgeries are available as follows:

Monday	**10:00 am– 12:00 pm**	**Online(Slack)**
Tuesday	**1:00 pm – 3:00 pm**	**C27 + Slack**
Wednesday	**1:00 pm – 3:00 pm**	**C27 + Slack**

You can attend in person or via Slack. Support and tuition is available for all aspects and levels of programming.

In advance of submission, you can access the support of the academic skills team. They can help you with any aspect of the assessment that you might struggle with, that is not content related. For example, they can help with time-management, effective reading and note-making, and any aspect of academic writing that you might struggle with. This support is provided through workshops and individual appointments which are bookable online via MyNapier: Improve your Academic & Study Skills (napier.ac.uk)

They are also able to provide formative feedback, within 5-days of your deadline, on a draft section, helping you to identify any issues of focus, structure or academic integrity in your writing. Please email Annemarie Douglas for any specific academic skills support you require: a.douglas@napier.ac.uk