# Web Apps and Services (CIS2021-N) Lab: Reprised

## Objective

Learn to use the C# programming language, and store your work within Git repositories, using Visual Studio by revisiting selected tutorial exercises from your first-year programming modules.

Before starting this tutorial, ensure you have reviewed the **C# Introduction** slide deck.

## Instructions

Before starting this lab, review the Introduction to C# slides.

Ensure you can open **Visual Studio 2022** and then work through the following tutorial projects. You should make them each as separate **C# .NET Core console apps**.

## Tutorial: Employees

In this tutorial you will create a simple Console application to calculate employee wages.

### New Application

* Start Visual Studio and **Create a new project** (from the Start Window or by using File > New > Project )
* Select **Console App** and click **Next**
* Set the Project Name to **EmployeeManager** and click **Next**
* For the Framework, ensure **.Net 6.x** is selected and click **Create**.
* Run the application (Debug > Start Debugging). You should see a screen similar to:



* Press any key to close the window.

### Employee Class

Add a new class called **Employee** to the EmployeeManager:

* In the Solution Explorer **Right-click** on the Project Name (Employee Manager – under the solution name) and then select **Add** > **Class…**
* Set the name to **Employee.cs** and click **Add**.
* Add the following to the new class (remember to test regularly by adding test code to **program.cs**):
* Properties:
  + Employee Name
  + Employee Id
  + Hours Worked
  + Hourly Rate
* Constructor that sets name, id, hours worked from parameter values and the Hourly Rate to a default value of 9.5
* ToString **override** method to return the employee name and id for example Koby Hills (D65).
* Add a method to Employee to calculate and return the wage (hours worked x hourly rate).

### Employee Name and Number

Update the **program.cs** file to ask the user to enter an Employee Name and Id (a letter followed by two digits) and display the weekly wage. For example (underline indicates keyboard inputs):

Employee Name: Koby Hills

Employee Id: D65

Hours Worked: 35

The weekly wage is £332.50

Note: weekly wage should be display with two decimal places.

### Validate the Employee Name

Update the Employee class to incorporate validation method for the Employee Name (names must be between 1 and 50 characters long inclusive):

* The validation should be implemented using a **static** method called **IsValidName**.
* The method takes a single parameter (the employee name) and returns **true** for valid names.

Update program.cs to incorporate the new method to validate the name – If the name is not valid, an appropriate error message is displayed and the user is asked to input the name again. Once a valid name is entered it is returned from the method.

### Additional Validation

Update the EmployeeManager to validate the following using the above approach:

* Employee Id: must start with a letter and be followed by two digits, for example: D65.
* Hours Worked: cannot be below 1 or over 100.

Remember to make an appropriate Git commit once this feature is working correctly.

### Employee List

Update the EmployeeManager to include an Employee List and menu to implement the following features:

1. Add Employee
2. List Employees *(with the position number in the list)*
3. Remove Employee *(based on a position number provided by the user)*

Remember to add a quit/exit option to the menu.

## Tutorial: Guess the Number (Optional)

### Implement a simple game

Design and build a simple text-based “Guess the Number” game that generates a random number (1-100) and asks the user to guess the number. After each guess, the user is told if the number was correct and the number of attempts so far. The game continues until the user guesses the number correctly.

You will need to make use of the C# documentation and/or websites like *StackOverflow* to discover the appropriate C# classes and functions to use.

You should make use of the Visual Studio debugger to help you identify and resolve any bugs.

### Add new features

Extend your design, test data and implementation to incorporate the following features:

* Limit the number of attempts to 20 and include a countdown of the attempts.
* Give a clue after each guess (too high or too low)
* Difficulty Level:
  + Easy (number range: 1-100 and 20 turns)
  + Medium (number range: 1-500 and 15 turns)
  + Hard (number range: -1000 to +1000 and 7 turns)

Incorporate one feature at a time.

### Add new play modes

Extend your design, test data and implementation to include two modes:

* Player guesses number (computer provides feedback: correct, too high or too low).
* Computer guesses number (user provides feedback: correct, too high or too low).