IT2154 *Assignment 1*: Total 100 marks (25%)

Instructions:

1. This is an individual assignment. Student who is caught cheating and practice ***plagiarism*** will face the following penalty according to **NYP policy**.
    a. First time in any assessment will fail the unit module
    b. Second time in any assessment will fail all units in that semester
    c. Third time in any assessment will be removed from the Polytechnic
2. Students should properly ***cite*** the source and ***credit*** the AI tools in their academic work and must NOT use AI tools for plagiarism by presenting generated output ***without proper acknowledgement*** of the ***original source.***
3. Download the PokemonPocket.zip file from Brightspace.
4. You are responsible for the correct and complete submission of your assignment.
5. Include your name and admin number in the beginning of each file.
6. Zip up the whole application folder, rename it as "<admin_no>_<name>_ASN1.zip" and submit via Brightspace.
7. Submit Assignment 1 **before 1 Jun (Sun), 23:59**.
8. **Penalty for late submission**:
    a. Cap at 50% of total marks for submission within 5 calendar days from deadline.
    b. From 6th day within deadline onwards, 0 mark will be given.

OBJECTIVE OF THE ASSIGNMENT

Apply C# programming knowledge such as OOP concept & Entity Framework in a scenario-based console application.

(Refer to **Appendix 1** for the Assessment Rubrics).

SCENARIO OF THE ASSIGNMENT

This assignment is to create a **Pokémon Pocket** console application to let Pokémon's player keep, view the Pokémon they caught and check if they can evolve. If the Pokémon in the pocket are ready to evolve, the player can evolve their Pokémon characters. Each time when players caught a Pokémon, they can add the Pokémon to this Pokémon Pocket program.

The Pokémon evolution criteria in this application are the imaginary version which means that it could be different from the actual game that you have played in the Pokémon App. The evolution of a Pokémon here is based on the number of the same Pokémon type that the player collected.

The current program has 1 class object, PokemonMaster class with attributes that describe the evaluation criteria such as the name of the Pokémon, number of the same Pokémon character to be evolved and what it will evolve to. This class has 3 objects – **Pikachu**, **Eevee** and **Charmander**.

Create a **Pokemon** class and implement subclasses for these characters. This means that the players can only add these 3 Pokémon characters to the pocket. You can add more Pokémon subclasses to make your program more interesting if you want to. The Pokémon class should have the following attributes:

- Name of Pokémon,
- HP,
- Exp (Experience),
- Skill and
- Skill Damage

Skill damage is the damage value deducted from the opponent's HP after a strike. The value for skill and skill damage attributes should be assigned in each of the subclasses based on the different type of the Pokémon characters shown in the table below. No change to the skill and skill damage for the evolved Pokémon.

Pokémon	**Skill**	**Skill Damage**
Pikachu	Lightning Bolt	30
Eevee	Run Away	25
Charmander	Solar Power	10

Each subclass will have a method named **calculateDamage** to calculate damage and update HP after each possible strike from other Pokémon. The method takes in skill damage of the striker as an argument and does not return a value. There is special requirement to calculate damage for each type of Pokémon before updating its HP as shown below. HP can be updated by subtracting the damage from current HP.

Pokémon	**Calculation of damage for specific Pokémon type**
Pikachu	3 * skill damage
Eevee	2 * skill damage
Charmander	1 * skill damage

SPECIFICATIONS FOR THE POKÉMON POCKET PROGRAM

1. Repeatedly display a menu for players to enter their choices until 'Q' or 'q' is being entered. Input validation is needed to prevent run-time exception.

```
*****************************
Welcome to Pokemon Pocket App
*****************************
(1). Add pokemon to my pocket
(2). List pokemon(s) in my Pocket
(3). Check if I can evolve pokemon
(4). Evolve pokemon
Please only enter [1,2,3,4] or Q to quit:
```

2. For implementation of option (1) in the menu, prompt user to enter data of the Pokémon they caught and store it in the Pokémon Pocket. Input validation is needed to prevent run-time exception.

```
*****************************
Welcome to Pokemon Pocket App
*****************************
(1). Add pokemon to my pocket
(2). List pokemon(s) in my Pocket
(3). Check if I can evolve pokemon
(4). Evolve pokemon
Please only enter [1,2,3,4] or Q to quit: 1
Enter Pokemon's Name: pikachu
Enter Pokemon's HP: 45
Enter Pokemon's Exp: 22
```

3. For implementation of option (2), list down the Pokémon that the user entered and sort the list in **Exp** order (*descending* order).

```
*****************************
Welcome to Pokemon Pocket App
*****************************
(1). Add pokemon to my pocket
(2). List pokemon(s) in my Pocket
(3). Check if I can evolve pokemon
(4). Evolve pokemon
Please only enter [1,2,3,4] or Q to quit: 2

Name: Eevee
HP: 45
Exp: 89
Skill: Run Away
------------------------------
Name: Eevee
HP: 23
Exp: 87
Skill: Run Away
------------------------------
Name: Charmander
HP: 67
Exp: 56
Skill: Solar Power
------------------------------
Name: Pikachu
HP: 23
Exp: 45
Skill: Lightning Bolt
------------------------------
Name: Pikachu
HP: 12
Exp: 34
Skill: Lightning Bolt
------------------------------
Name: Eevee
HP: 12
Exp: 21
Skill: Run Away
------------------------------
```

4. For implementation of option (3), list available Pokémon characters that can be evolved in the following format.

```
*****************************
Welcome to Pokemon Pocket App
*****************************
(1). Add pokemon to my pocket
(2). List pokemon(s) in my Pocket
(3). Check if I can evolve pokemon
(4). Evolve pokemon
Please only enter [1,2,3,4] or Q to quit: 3
Charmander --> Charmeleon
```

If there are 2 and more Pokémon can be transformed, it will display in the following format.

```
*****************************
Welcome to Pokemon Pocket App
*****************************
(1). Add pokemon to my pocket
(2). List pokemon(s) in my Pocket
(3). Check if I can evolve pokemon
(4). Evolve pokemon
Please only enter [1,2,3,4] or Q to quit: 3
2 Pikachu --> 1 Raichu
3 Eevee --> 1 Flareon
1 Charmander --> 1 Charmeleon
```

5. For implementation of option (4), evolve **ALL** available Pokémon character(s) at one time. The newly evolved Pokémon will have its HP=100 and Exp=0.

```
*****************************
Welcome to Pokemon Pocket App
*****************************
(1). Add pokemon to my pocket
(2). List pokemon(s) in my Pocket
(3). Check if I can evolve pokemon
(4). Evolve pokemon
Please only enter [1,2,3,4] or Q to quit: 2

Name: Charmeleon
HP: 100
Exp: 0
Skill: Solar Power
------------------------------
Name: Raichu
HP: 100
Exp: 0
Skill: Lightning Bolt
------------------------------
Name: Flareon
HP: 100
Exp: 0
Skill: Run Away
------------------------------
```

--End of Assignment—

Appendix 1: IT2154 Assignment 1 Rubrics: Total 100 marks (25%)

Criteria	Not Ready (Below D) 0-40 marks	Beginning (D/D+) < 60 marks	Developing (C/C+) < 70 marks	Functional (B/B+) < 80 marks	Advanced (A/A+) 80-100 marks
Implementation (55%)	Does not implement at least 1 basic feature as required for the assignment. The program does not work and has many errors.	Successfully implemented at least 2 basic features as required. Program works with minimum errors but does not produce correct results nor display correctly. No validation and exception handling implemented.	Successfully implemented at least 3 basic features as required. Program works with minimum non-critical errors and most of the time produces expected outcomes. Little validation and exception handling implemented.	Besides successful implementation of all basic features as required, at least 1 additional simple feature with good usability is correctly implemented. Program works well with minimum errors and consistently produces expected outcomes. Adequate validation and exception handling implemented.	Besides successful implementation of all basic features as required, at least 1 additional comprehensive & creative feature with good usability are well implemented. Program works well with no errors and consistently produces expected outcomes. Full validation and exception handling implemented.
Coding Practices (15%)	No demonstration of any good coding practice (e.g., naming convention, readable alignment, etc)	The program lacks organization and logic.	The program has some organization and logic.	The program is organized, logical.	The program is particularly well-organized, logical, and well-debugged.

OOP Concepts & Usage of Entity Framework (30%)	No demonstration of OOP concept	The program code shows poor understanding and implementation of applying OOP in C# programming and how the objects work together. No problem-solving skill in C# observed in the code.	The program code shows little understanding and implementation of applying OOP in C# programming and how objects work together. It demonstrates little problem-solving skill in C# observed in the code.	The program code shows understanding and implementation of applying OOP in C# programming and how the objects work together to meet a goal. It demonstrates adequate problem-solving skill in C# observed in the code.	The program code shows advanced understanding and implementation of applying OOP and Entity Framework in C# programming. It demonstrates good problem-solving level in C# observed in the code.

--End of Assignment 1 Rubrics--