PGCert IT: Programming for Industry

Homework Task 02

Time estimate: **90 minutes**

Notes:

- There are **two questions** in this practice task. It is worth **2.5%** of your final grade.
- You will be receiving marks by demonstrating your efforts on attempting the question.
- The question format will be similar to the actual test.
- You will be provided with a Zip file containing an IntelliJ project which will serve as a starting point for the task.
- To submit your answers, Zip your project and submit the single Zip file to Canvas or Moodle at or before the due date.
- **Important:** Make sure to read all instructions carefully before attempting the question.

Question One - Flying

Complete the Flying program (the files are in the Flying program folder) as per the instructions listed in Steps 1 – 4 below so that it produces the following output:

```
1968 Piper Cherokee
1980 Cessna 152
2006 Cessna 172
1980 Piper Tomahawk
1956 Piper Cub
1970 Learjet 23
1968 Piper Cherokee
2001 Airbus A320
2006 Cirrus SR22
1986 Cessna Caravan

Oldest aeroplane is: 1956 Piper Cub
```

1. Declare an array of `Aeroplane` objects. Name the array `planes`.
2. Construct the `planes` array that you declared in Step 1. The planes array needs to be big enough to hold 10 Aeroplane objects.
3. Write the `printPlanesArray()` method. This method takes an array of `Aeroplane` objects as a parameter and prints all the elements as per the screenshot above. Note that the `toString()` method in the Aeroplane class can be called to obtain a String containing the instance variables of a particular Aeroplane, formatted in the required manner.
4. Write the `getOldestAeroplane()` method. This method takes an array of `Aeroplane` objects as a parameter and returns a reference to the oldest Aeroplane. Note that the isOlderThan() method in the Aeroplane class can be used to determine if an Aeroplane is older than another Aeroplane.

Question Two - Pokemon

Complete the instructions listed in steps a – g. In this question, you are required to create the following classes:

- `ElectricType`
- `WaterType`
- `FireTepe`
- `Psyduck`
- `Charmander`
- `Squirtle`

You are required to modify the following classes:

- `Pikachu`
- `PokemonGenerator`

The following classes are provided to you in the `question1` folder:

- `Pokemon`
- `PokemonGenerator`
- `IType`
- `INoise`
- `Pikachu`

After completing these steps, the `PokemonGenerator` class should produce output similar to that in the screenshot overleaf.

a. Create three classes, `ElectricType`, `WaterType` and `FireType`, which implement the `IType` interface.
   i. The `getType()` method in the `ElectricType` class returns a String `"Electric"`
   ii. The `getType()` method in the `WaterType` class returns a String `"Water"`
   iii. The `getType()` method in the `FireType` class returns a String `"Fire"`

b. Examine the `Pikachu` class that's provided to you.
   i. Make sure you understand what the `attack()` and `levelUp()` methods do. Understanding of these methods will be *very* helpful when implementing other Pokemon below.
   ii. Modify the constructor for the `Pikachu` class to initialize its `type` to a new `ElectricType` object. Remember that the `type` variable is already defined in the `Pokemon` class.
   iii. Make `Pikachu` implement the `INoise` interface. Modify Pikachu's makeNoise() method to return a String `"Pika pika"`.

```
Greetings from Pokemon
====================
I am Squirtle, my current level is 1
I am Charmander, my current level is 5
I am Pikachu, my current level is 500
I am Psyduck, my current level is 5
====================

Water Pokemon show-off time
------------------------------
I say "Squirtle squirt" when I attack!
I say "Psy~~~duck" when I attack!
------------------------------

Random attack time!
------------------
Charmander attacked Pikachu!
Charmander never levels up :(
Psyduck attacked Squirtle!
Not enough experience for Psyduck!
Pikachu attacked Charmander!
Pikachu levelled up!
------------------

Pokemons' status after the attacks
==================================
I am Squirtle, my current level is 1
I am Charmander, my current level is 5
I am Pikachu, my current level is 200
I am Psyduck, my current level is 5
==================================
```

c. Create a `Psyduck` class which extends the `Pokemon` class and implements the `INoise` interface.
   i. Create the constructor for the `Psyduck` class. It should take a String `name` and int `level` as the parameters. Within the constructor, set its `name` and `level` using the values from the parameters. Initialise `type` to a new `WaterType` object.
   ii. The `makeNoise()` method should return a String `"Psy~~~duck"`
   iii. Complete the `levelUp()` method so if the `Psyduck`'s life points are more than 10 times its current level, then it can level up. When Psyduck levels up, you should print out the following message `"Psyduck levelled up!"`. Otherwise, `"Not enough experience for Psyduck!"` should be printed.
   iv. Complete the `attack()` method which will make the defending Pokemon (supplied as an argument) lose life points equal to 20 times Psyduck's current level. Psyduck should also gain one life point for each attack. Finally, Psyduck should check if it levels up at this point. *Remember* you can use the existing Pikachu class as a guide on how to implement these methods.

d. Create a `Squirtle` class which extends the `Pokemon` class and implements the `INoise` interface.
    i. Create the constructor for the `Squirtle` class. It should take a String `name` and int `level` as the parameters. Within the constructor, set its `name` and `level` using the values from the parameters. Initialise `type` to a new `WaterType` object.
    ii. The `makeNoise()` method should return a String `"Squirtle squirt"`
    iii. Complete the `levelUp()` method so if the `Squirtle`'s life points are more than 6 times its current level, then it can level up. When Squirtle levels up, you should print out the following message `"Squirtle levelled up!"`. Otherwise, `"Not enough experience for Squirtle!"` should be printed.
    iv. Complete the `attack()` method which will make the defending Pokemon lose life points equal to 2 times Squirtle's current level. Squirtle should also gain two life points for each attack. Finally, Squirtle should check if it levels up at this point.

e. Create a `Charmander` class which extends the `Pokemon` class and implements the `INoise` interface.
    i. Create the constructor for the `Charmander` class. It should take a String `name` and int `level` as the parameters. Within the constructor, set its `name` and `level` using the values from the parameters. Initialise `type` to a new `FireType` object.
    ii. The `makeNoise()` method should return a String `"Charmander Char"`
    iii. Complete the `levelUp()` method so it displays `"Charmander never levels up :("` every time it's called.
    iv. Complete the `attack()` method which will make the defending Pokemon lose life points equal to 100 times Charmander's current level. Charmander should also gain five life points for each attack. Finally, Charmander should check if it levels up at this point (even though it never actually will - just in case we want to change this later!).

f. Modify the `PokemonGenerator` class to complete the program.

   i. At the location marked “step i”, create a new method called `generateRandomLevel()`. This should return a random integer between 1 - 5, inclusive.

   ii. At the location marked “step ii”, initialize the provided `pokemons` array to hold four elements, and fill it with one of each kind of Pokemon we’ve created thus far (Squirtle, Charmander, Pikachu, Psyduck). The Pokemon should be in the order as per the screenshot above. Each Pokemon’s level should be randomly assigned using the method you created in step i above.

   iii. Write the `printPokemonGreetings()` method in the marked location. It should print all the `Pokemon` elements as per the screenshot above. Note that you also need to include the current level of each Pokemon by using its `getLevel()` method (i.e. not the field value). Once you’ve written the method, call it in the **two** marked locations.

   iv. Write the `printWaterPokemon()` method in the marked location. It should print all the sounds of each water Pokemon element. The sounds should be formatted as shown in the screenshot. Once you’ve written the method, call it in an appropriate location.

   v. At the marked location, randomly generate three attacks. The console output should be similar to that shown in the screenshot.
      - For each attack, randomly choose one Pokemon to be the attacker and another to be the defender.
      - A Pokemon cannot attack itself.