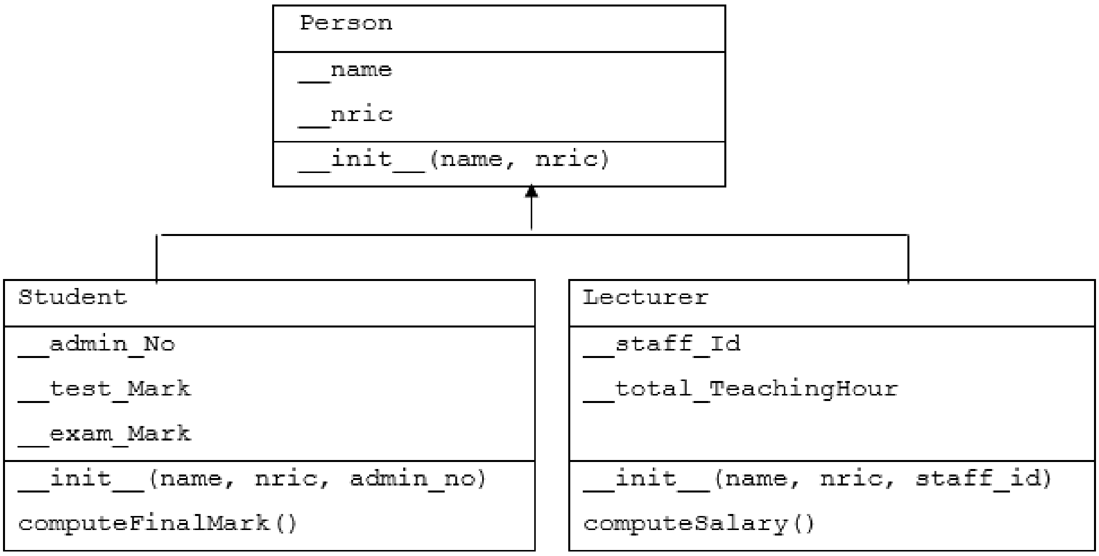
Activity
Inheritance & Polymorphism

Learning Outcome

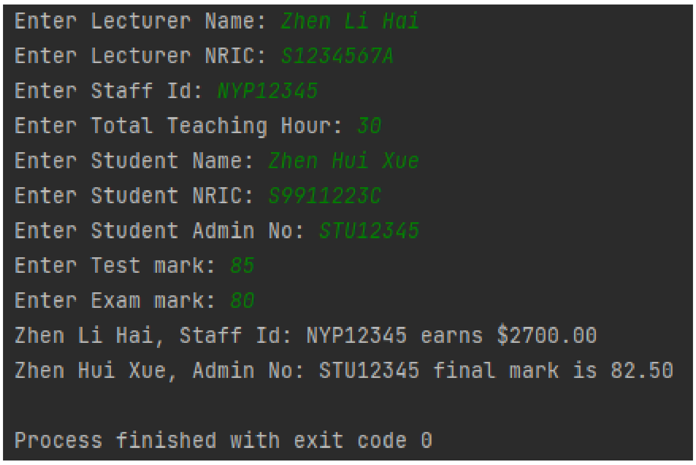
On successful completion of this practical, the students should be able to:

- Explain the concept of inheritance.
- Identify and apply superclasses and subclasses.
- Apply inheritance in a program.
- Explain the concept of polymorphism.

1. Implement the concept of inheritance by:
   a) Implementing the 3 classes (Person.py, Student.py, and Lecturer.py) as shown in the following UML diagram:
      - Include all accessor and mutator methods.
      - For student class, call the initializer method in base class using class name.
      - For Lecturer class, call the initializer method in base class using super() built-in function.
      - The base mark of the test and exam is 100% and each contributes 50% to the final mark of a student e.g final mark 65 for test mark 50 and exam mark 80.
      - The salary of a lecturer is computed using an hourly rate of $90 e.g salary of $720 for teaching 8 hours.



   b) Create a program TestApp.py that creates an object of the Lecturer and Students class using the input entered by the user. At the end of the program, display the Lecturer and Student's details on the screen. The sample output is given below.

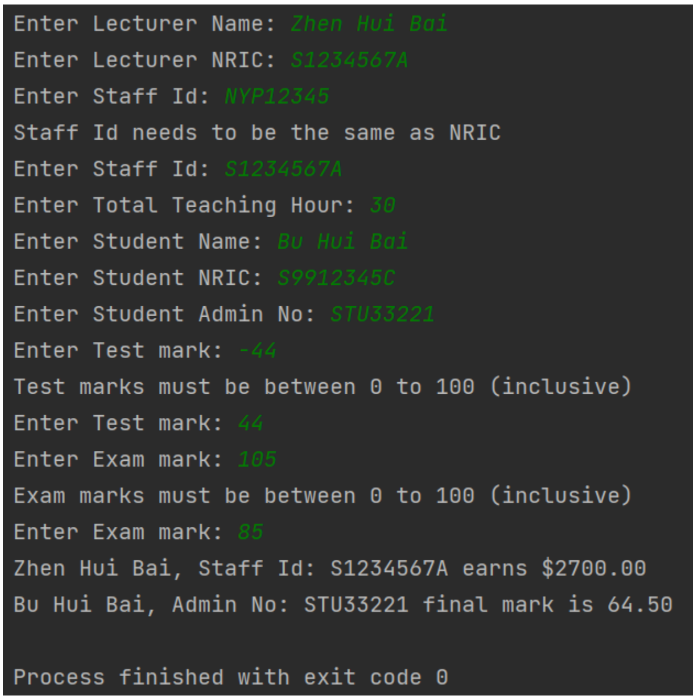


```
Enter Lecturer Name: Zhen Li Hai
Enter Lecturer NRIC: S1234567A
Enter Staff Id: NYP12345
Enter Total Teaching Hour: 30
Enter Student Name: Zhen Hui Xue
Enter Student NRIC: S9911223C
Enter Student Admin No: STU12345
Enter Test mark: 85
Enter Exam mark: 80
Zhen Li Hai, Staff Id: NYP12345 earns $2700.00
Zhen Hui Xue, Admin No: STU12345 final mark is 82.50

Process finished with exit code 0
```

Remarks: You might want to modify Student and Lecturer class to include a built-in-function __str__ for displaying the Lecturer and Student's details on the screen.

2. Modify Question 1 to add in the following validation:
    - Lecturer's NRIC must be same as staff Id
    - Test mark and exam mark must be between 0 and 100 (inclusive)



```
Enter Lecturer Name: Zhen Hui Bai
Enter Lecturer NRIC: S1234567A
Enter Staff Id: NYP12345
Staff Id needs to be the same as NRIC
Enter Staff Id: S1234567A
Enter Total Teaching Hour: 30
Enter Student Name: Bu Hui Bai
Enter Student NRIC: S9912345C
Enter Student Admin No: STU33221
Enter Test mark: -44
Test marks must be between 0 to 100 (inclusive)
Enter Test mark: 44
Enter Exam mark: 105
Exam marks must be between 0 to 100 (inclusive)
Enter Exam mark: 85
Zhen Hui Bai, Staff Id: S1234567A earns $2700.00
Bu Hui Bai, Admin No: STU33221 final mark is 64.50

Process finished with exit code 0
```

Remarks: The program should continue to ask for input until valid input entered.

3. Create a class called Monster that has the following private attributes
    - name
    - health
    - attack
    - defence

Provide the initializer that will initialize all the attributes of this class. Implement mutator and accessor methods for all the attributes and provide a display method that prints “name is a Monster”.

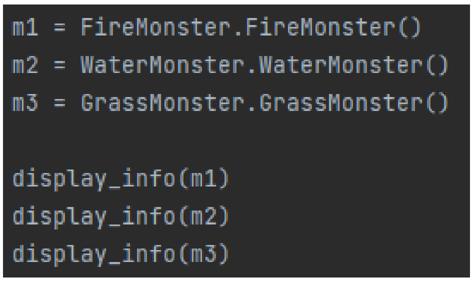
Create a test program to verify the Monster class.

Create 3 subclasses called FireMonster, WaterMonster and GrassMonster that inherits from the Monster class. Provides an __init__() method with no argument that overrides the parent initializer with the default attribute values for each type of monster given below:

	name	health	attack	defence
FireMonster	firebug	10	9	4
WaterMonster	waterbird	15	6	3
GrassMonster	grasshopper	20	5	3

Remarks: For child class initializer, explore using __init__('firebug', 10, 9, 4) or __init__(name='grasshopper', health=20, attack=5, defence=3)

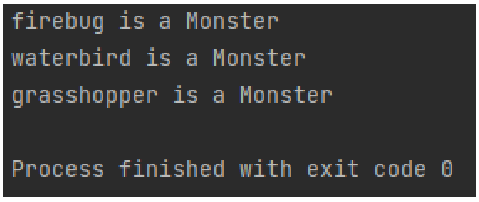
Creates a test program with a function, display_info(monster) that will check if a passed in monster argument is a Fire, Water or GrassMonster and print out the messages accordingly. Given the following sample test program,



```
m1 = FireMonster.FireMonster()
m2 = WaterMonster.WaterMonster()
m3 = GrassMonster.GrassMonster()

display_info(m1)
display_info(m2)
display_info(m3)
```

The sample output is given as below:

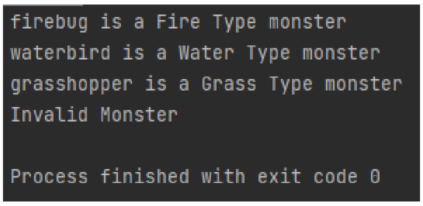


```
firebug is a Monster
waterbird is a Monster
grasshopper is a Monster

Process finished with exit code 0
```

Override the superclass’s display method to allow subclass to display corresponding message. Modify the display_info function to handle invalid monster type argument.

The sample output is given as below:



```
firebug is a Fire Type monster
waterbird is a Water Type monster
grasshopper is a Grass Type monster
Invalid Monster

Process finished with exit code 0
```

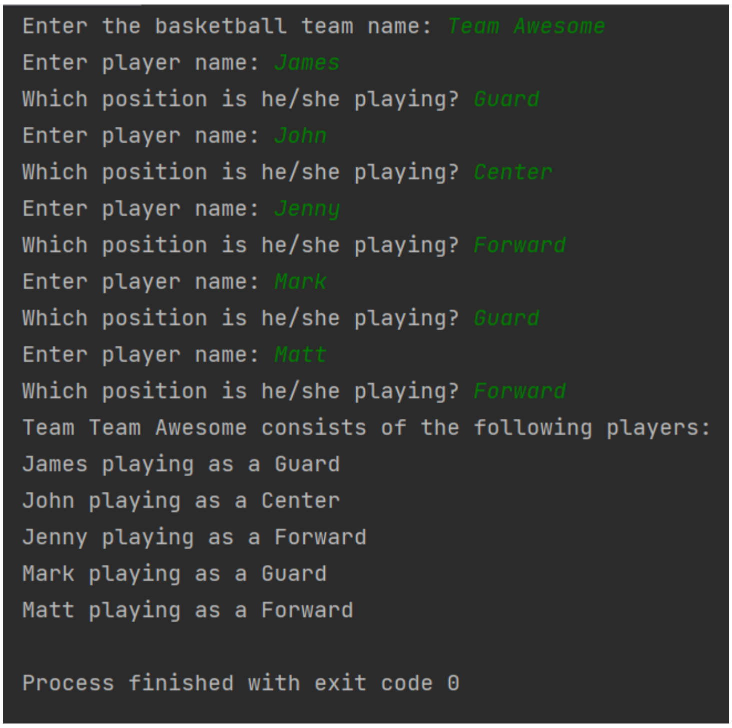
4. Create a class called MonsterGame that contains the following attributes:
   - computer_monster
   - player_monster

   a) Creates a method, **choose_monster()** that will allow the players to choose a monster (Fire, Water or Grass) and assign this to the attribute **player_monster**.
   b) Creates another method, **generate_monster()** that will create a random monster and assign to the attribute **computer_monster**. Hint: Python random generator
   c) Create an intializer that will call these 2 methods and assign to the attributes accordingly

5. (Challenge) Write a **Player** class with data attribute for player name. It should contain an initializer that takes parameter to set the attribute. Next, write a class named **BasketballPlayer** that is a subclass of the **Player** class. The **BasketballPlayer** class should have a data attribute for the player's playing position. Write the appropriate accessor and mutator methods for each class.

   The valid playing positions are 'Guard', 'Forward' and 'Center', write a class attribute called positions for **BasketballPlayer** class to store these valid positions in a List. Modify the mutator method in **BasketballPlayer** to only assign a valid position to the player.

   Write a program that creates a basketball team with 5 players with their respective position. The sample output is given below:



```
Enter the basketball team name: Team Awesome
Enter player name: James
Which position is he/she playing? Guard
Enter player name: John
Which position is he/she playing? Center
Enter player name: Jenny
Which position is he/she playing? Forward
Enter player name: Mark
Which position is he/she playing? Guard
Enter player name: Matt
Which position is he/she playing? Forward
Team Team Awesome consists of the following players:
James playing as a Guard
John playing as a Center
Jenny playing as a Forward
Mark playing as a Guard
Matt playing as a Forward

Process finished with exit code 0
```

Remarks: Explore using list to store the created players.

-End-