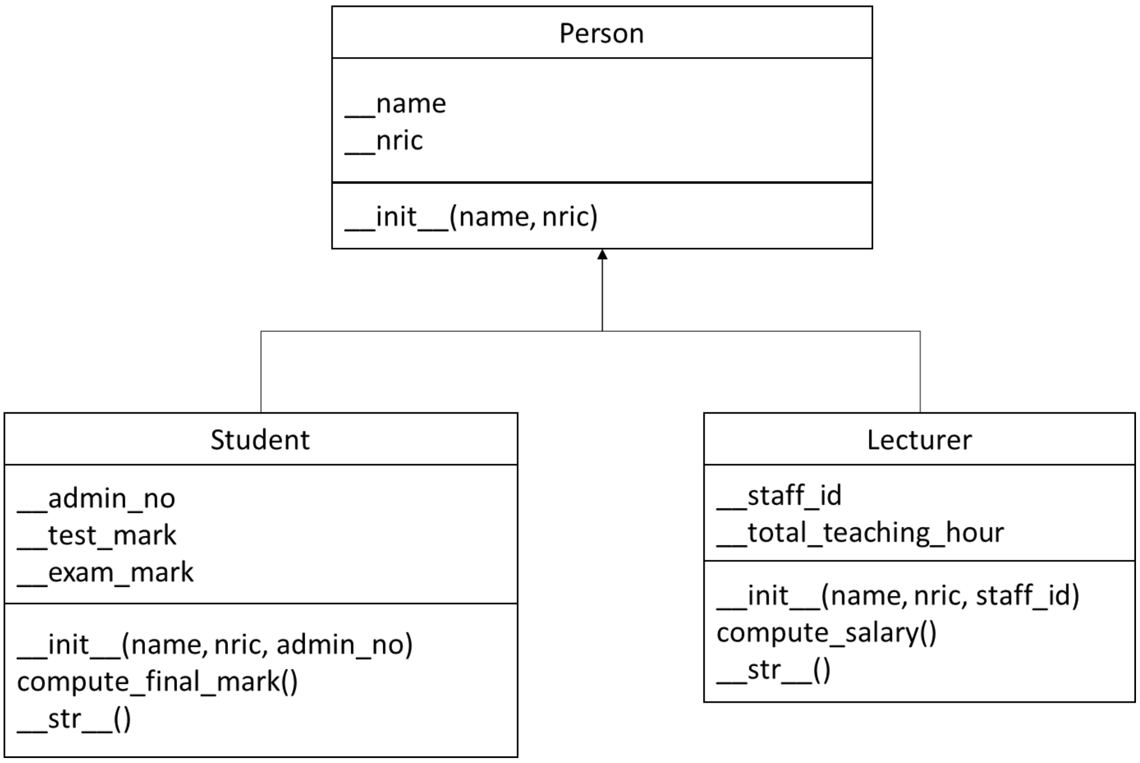
Activity
Inheritance & Polymorphism

Learning Outcome

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

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

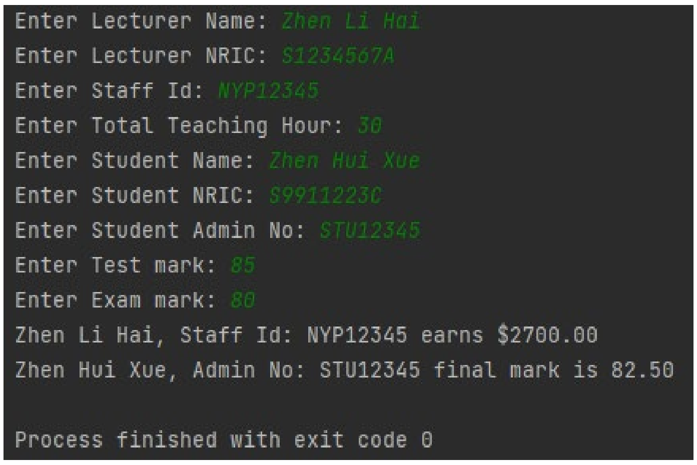
1. Implement the concept of inheritance by:
   a) Create a base class Person in Person.py as shown in the following UML diagram:
      - Include all accessor and mutator methods.



   b) Implement the Student class in Student.py as shown in the UML diagram:
      - Call the initializer method in base class using **class** name.
      - Include all accessor and mutator methods.
      - The compute_final_mark method will compute the final mark where 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.
      - built-in-function __str__() to display the details as the sample output

   c) Implement the Lecturer class in Lecturer.py as shown in the UML diagram:
      - Call the initializer method in base class using **super()** built-in function.
      - Include all accessor and mutator methods.
      - The compute_salary method will compute the salary where the salary of a lecturer is computed using an hourly rate of $90 e.g. salary of $720 for teaching 8 hours.
      - built-in-function __str__() to display the details as the sample output

d) Create a test program TestApp.py that creates an object of the Lecturer and Student 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.



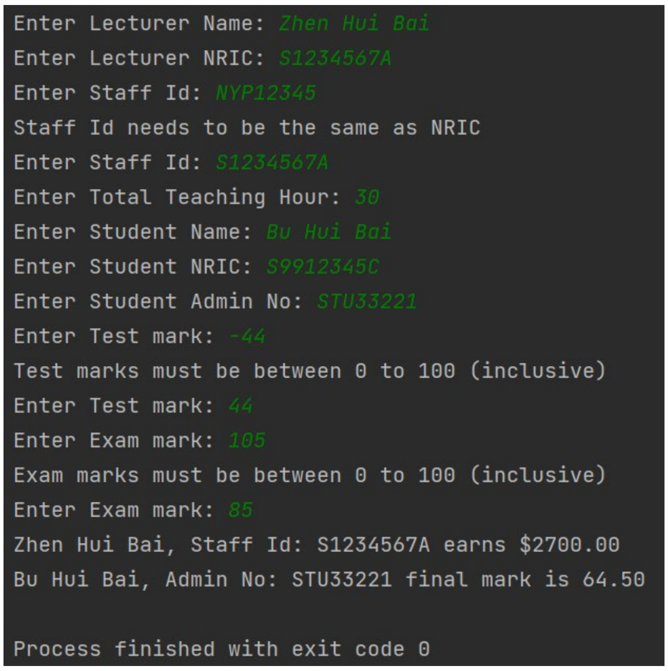
```
Enter Lecturer Name: Zhen Li Hai
Enter Lecturer NRIC: S1234567A
Enter Staff Id: NYP12345
Enter Total Teaching Hour: 30
Enter Student Name: Zhen Hui Xue
Enter Student NRIC: S9911223C
Enter Student Admin No: STU12345
Enter Test mark: 85
Enter Exam mark: 80
Zhen Li Hai, Staff Id: NYP12345 earns $2700.00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

Process finished with exit code 0
```

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)

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

   Sample Output:



```
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
```

   Note: Where should you add the validation? Initializer? Mutator method? Test program?

3. a) Write a **Player** class with data attribute for player name. It should contain an initializer that takes parameter to set the attribute. In the same py file, write a class named **BasketballPlayer** that is a subclass of the **Player** class. The **BasketballPlayer** class have a data attribute for the player's playing position.

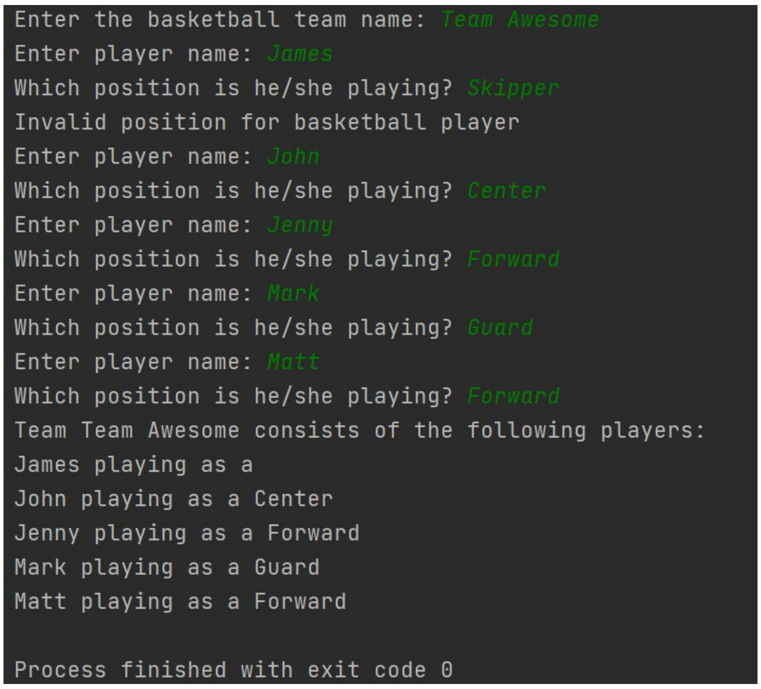
   Write the appropriate accessor and mutator methods for each class.

The valid playing positions are ‘Guard’, ‘Forward’ and ‘Center’, write a <u>class attribute</u> called <u>positions</u> for **BasketballPlayer** class to store these valid positions in a <u>List</u>. Modify the mutator method in **BasketballPlayer** to only assign a valid position to the player. Otherwise display error message “*Invalid position for basketball player*”.

Implement the built-in-function __str__() to display the details of BasketballPlayer with its’ name and position.

b) Write a test program that creates and store a basketball team with 5 players with their respective position in a list.
The program will also display the details of the basketball team as the sample output.

Sample Output:



```
Enter the basketball team name: Team Awesome
Enter player name: James
Which position is he/she playing? Skipper
Invalid position for basketball player
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 
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
```

Note: Why no position for James?

4. a) Create a class called Monster in Monster.py that has the private attributes name, health, attack and 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 it’s “name is a Monster”.

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

	name	health	attack	defence
FireMonster	firebug	10	9	4
WaterMonster	waterbird	15	6	3

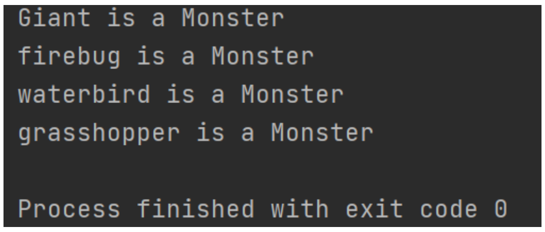
GrassMonster	grasshopper	20	5	3

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

c) Create a test program Main.py that create an object of Monster with the values in the given table and the 3 subclass objects. Print out the messages accordingly to the sample output below:

	name	health	attack	defence
Monster	Giant	17	6	7

Sample Output:

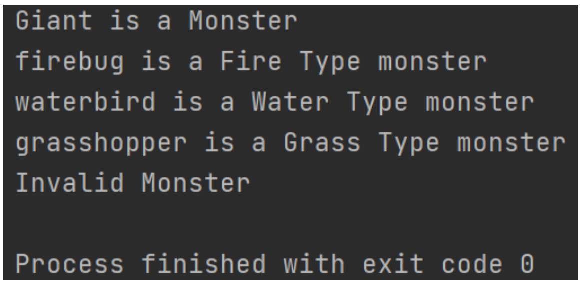


```
Giant is a Monster
firebug is a Monster
waterbird is a Monster
grasshopper is a Monster

Process finished with exit code 0
```

5. Modify Question 4:
   a) Override the superclass's display method, for example, to allow the subclass to display the corresponding message – “Grasshopper is a grass type monster”.
   b) Modify the test programme to include a function, display_info (monster), that will verify if a passed in monster argument is an instance of Monster and print out the messages accordingly. Otherwise, it will print "Invalid Monster".

   Call the display_info(monster) with the objects previously created in Question 4.
   Sample Output:



```
Giant is a Monster
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
```

   Note: What argument will in "Invalid Monster" being printed?
   Note: How to verify if the passed in monster argument is an instance of the subclass? e.g print a message only for an instance of FireMonster?

-End-