Activity
Classes, Objects and Methods

Learning Outcome

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

- Explain the OO concepts of classes, objects, methods, and messages.
- Implement a class with instance variables, instance methods and constructors.
- Explain the concept of abstraction and encapsulation.
- Construct a program using classes, objects, methods and messages.

1. Write a class named Customer with the following data attributes:
   - __name (for recording the name of the Customer)
   - __email (for recording email address of the Customer)
   - __mobile_number (for recording contact number of the Customer)

   a) It should contain an __init__() initializer that creates and initializes the data attributes with the parameters.

   b) Include the given get_customer_info() method (for displaying the data attributes of the object).

```
def get_customer_info(self):
    return f"Name: {self.__name}, Email: {self.__email}, Mobile Number: {self.__mobile_number}"
```

   c) Write a test program that creates an object of the class Customer with John as the name, email john@nyp.edu.sg and mobile number is 92345678. The test program should also display the data attributes of the object created.

   Sample Output:

```
Name: John, Email: john@nyp.edu.sg, Mobile Number: 92345678
```

2. Write a class named Phone with the following data attributes:
   - __make
   - __model
   - __price

   a) It should contain an __init__() initializer that creates and initializes the data attributes with the parameters.

   b) Include a get_phone_info() method which display the data attributes as follows:

   Sample output for a phone that make by Samsung, model S22 with price $1568:

```
The price for Samsung S22 is $1568
```

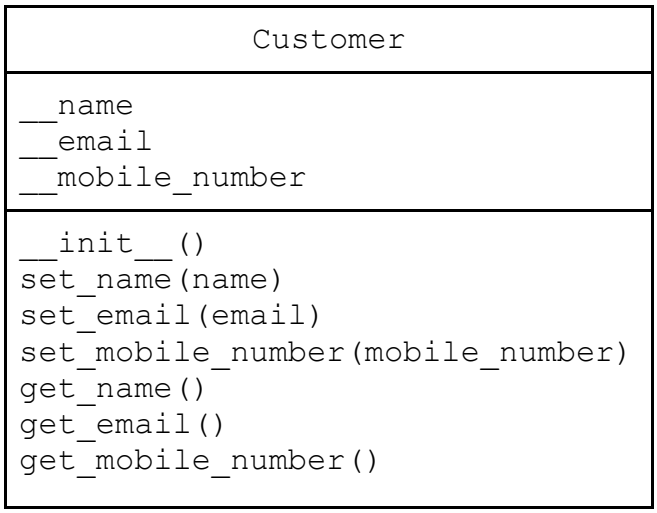
   c) Write a test program that creates an object of the class Phone. The test program should prompt the user to enter the value for each data attribute. At the end of the program, display the data attributes of Phone object created on the screen.

   Sample Output:

```
Enter the make of the phone: Apple
Enter the model of the phone: iPhone 12 Pro Max
Enter the price of the phone: 1500
The price of Apple iPhone 12 Pro Max is $1500

Process finished with exit code 0
```

3. Modifying solution in Question 1 to implement the concept of encapsulation. Using the provided UML diagram below:



a) Modify the Customer class:
   i. In __init__(), assign all attributes to 'None' value (None is used to define a null variable or an object)

      ***Let's Ponder**: Do you notice the difference between the Question 1 initializer and this __init__()?*

   ii. Create accessor and mutator methods for all the attributes defined in __init__()

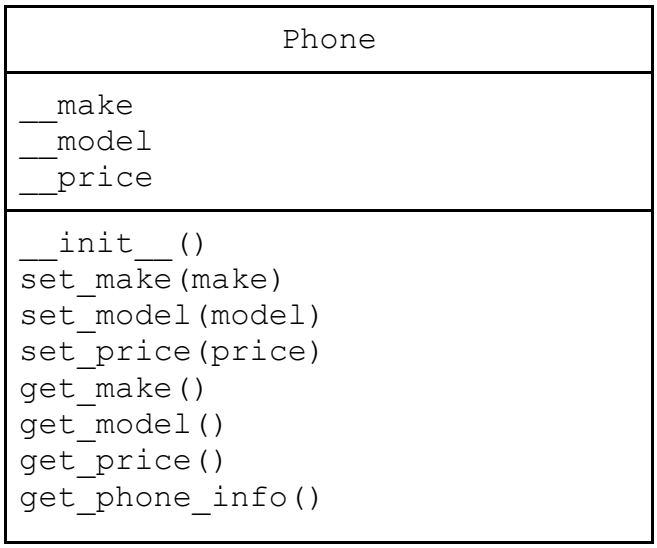
b) Write a test program for the newly encapsulated Customer class. The test program should perform the following:
   i. Prompt the user to enter the value for each attribute
   ii. Make use of the mutator methods to set the values for each attribute
   iii. Make use of the accessor methods to retrieve the customer information and print the results

   The sample output as follows:

```
Enter your name: Ah Hua
Enter your email: ahhua@gmail.com
Enter your mobile number: 91234567
Name: Ah Hua
Email: ahhua@gmail.com
Mobile number: 91234567

Process finished with exit code 0
```

4. Modify the solution in Question 2 using the provided UML diagram below:



a) Modify the Phone class:
   i. In __init__(), assign 0 to price and 'None' to make and model
   ii. Create accessor and mutator methods for all the attributes defined in __init__()
   iii. Write validation code in the mutator method for mobile price to ensure only numbers are entered. If the input is not in numeric form, display an error informing that the mobile price input is not in numerical form. Making use of the function, isnumeric(), to perform a check on price attribute

b) Write a test program for the newly encapsulated Phone class. The test program should perform the following:
   i. Prompt the user to enter the value for each attribute
   ii. Make use of the mutator methods to set the values for each attribute
   iii. Print out the results by calling get_phone_info()

   Sample Output 1:

```
Enter the make of the phone: Apple
Enter the model of the phone: iPhone 13
Enter the price of the phone: ABC
Price should be in numbers.
The price of Apple iPhone 13 is $0
```

   ***Let's Ponder**: Why the price is 0?*

   Sample Output 2:

```
Enter the make of the phone: Samsung
Enter the model of the phone: Galaxy S
Enter the price of the phone: 256
The price of Samsung Galaxy S is $256

Process finished with exit code 0
```

5. Creating a new class named SalesPerson with the following data attributes:
   - __name
   - __commission (for recording commission of the SalesPerson after selling a phone)

   c) It should contain an __init__() initializer that creates and initializes the data attributes name and commission to None and 0.

   d) The class should contain the following:
      i. accessor and mutator methods for only __name attribute
      ii. Create a method named salesperson_commission that accepts the payment_received parameter. This method will calculate commission of the salesperson and assign to the commission attribute. Commission of the salesperson is 2% of the payment_received
      iii. built-in-function __str__() to display the data attributes

   e) Write a test program for SalesPerson class that will perform the following:
      i. Prompt the user to enter the value for salesperson name, and payment received
      ii. Print salesperson information using built-in-function

      Sample Output:

```
Enter salesperson name: Xiao Qiang
Enter payment received by salesperson: 1500
The commission of salesperson Xiao Qiang is $30.00

Process finished with exit code 0
```

6. Implement the concept of abstraction, modules by modifying SalesPerson class to import and use methods from Phone class.

   a) Modify the Phone class to include a class attribute count and replace get_phone_info() with built-in-function __str__(). Output from __str__() must display as follows:

```
The phone created is Apple iPhone 13 priced at $1500. Now has 1 phone in total
```

   b) Modify SalesPerson class to include the following:
      i. Add phone attribute into __init__() and initialize it to None
      ii. Create a method named salesperson_sold that accepts a Phone object as parameter and assigns the object to phone attribute
      iii. Modify __str__() such that it displays the following:

```
Salesperson Xiao Qiang sold Apple iPhone 13 at $1500 and earned a commission of $30.00.
```

c) Write a test program named TestProgram.py that will perform the following:
   i. Prompt user to enter the make, model, color and price of the phone
   ii. Create Phone object and use mutator method to set the respective attributes of the phone
   iii. Print phone information
   iv. Prompt the user to enter the value for salesperson name, and payment received
   v. Print salesperson information

Proceed to run and complete the test with the following results:

```
Enter the make of the phone: Apple
Enter the model of the phone: iPhone 13
Enter the price of the phone: 1500
The phone created is Apple iPhone 13 priced at $1500. Now has 1 phone in total
Enter salesperson name: Xiao Qiang
Enter payment received by salesperson: 1500
Salesperson Xiao Qiang sold Apple iPhone 13 at $1500 and earned a commission of $30.00.

Process finished with exit code 0
```

-End-