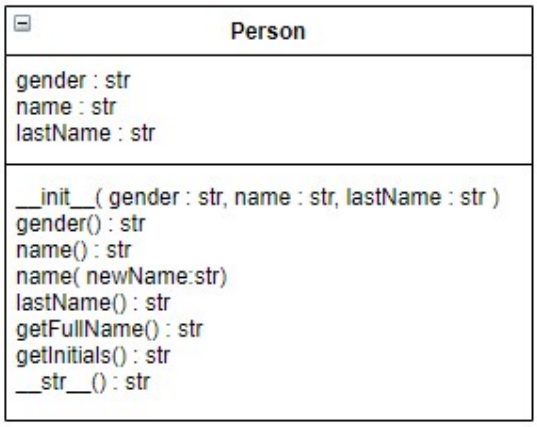
Seminar 1 (Classes and Objects) - Exercises

1. Write a class Person that models a person's particulars as shown in the UML diagram:



The class has 3 instance variables – gender (str), name (str), last name(str).

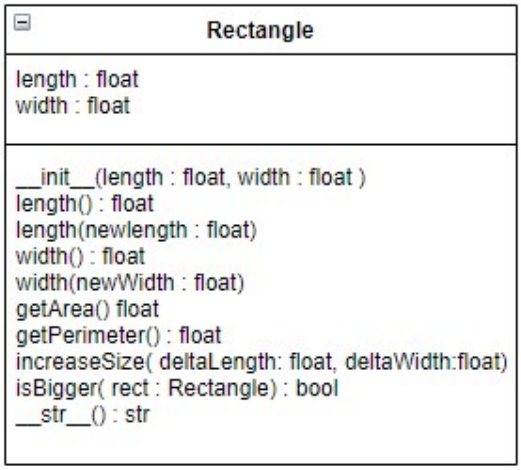
It has a constructor that initializes the gender, name and last name. It has getter and setter methods only for name.

It has the following methods:
- getFullName() returns the full name with a salutation "Mr." or "Ms.", depending on the person's gender (m or f). The name is given in this order: the last name, name followed by the middle name, e.g., "Mr. Ong Ah Seng"
- getInitials() returns the first letter of the name separated by blanks, followed by the lastname. E.g. it may return "A. Ong".
- __str__() method that returns a string representation of a Person object as in:

```
Name: Ong Ah Seng     Gender : Male
```

Test out the class, by creating a Person object and calling the methods in the class.

2. Write a class that models a Rectangle. The class diagram is given as follows:



A Rectangle class has 2 instance variables – length (float) and width (float).

It has a constructor that initializes the length and width. It also has get/set methods for the length and width.

It has the following methods:

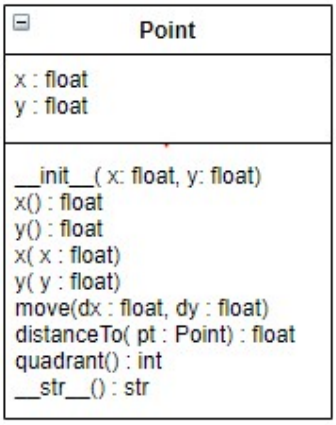
- getArea() that returns the area of the rectangle
- getPerimeter() that returns the perimeter of the rectangle
- increaseSize(deltaLength, deltaWidth) that increases the length and width of the rectangle by deltaLength and deltaWidth respectively.
- isBigger(rect) that has another rectangle as parameter. The method returns True if the current area is bigger than the area of the rectangle rect and False otherwise.
- __str__() method that returns a string representation of a Rectangle object, including its area and perimeter as in:

```
Length: 2.0  Width: 5.0  Area: 10.0  Perimeter: 14.0
```

Test out the Rectangle class with the following statements:

i. Create a Rectangle object r1 with any dimension.
ii. Print the details of r1.
iii. Increase the size of the rectangle by 10 units on both sides.
iv. Print the details of r1 again.
v. Create another rectangle r2 with any dimension.
vi. Print the area and perimeter using the getArea() and getPerimeter() methods.
vii. Compare r1 with r2 using the isBigger() method. Display the outcome.

3. Write a class Point that models a 2 dimensional point (x,y). The UML diagram is as follows:



The Point class has 2 instance variables, x and y.

It has a constructor that initializes 2 instance variables with the value of x and y, with default values (0,0).

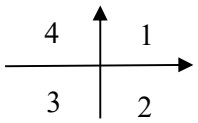
It has the following methods:

- Getter and setter properties for x and y .
- A move(dx, dy) method that moves to (x+dx, y+dy).

- A distanceTo(aPoint) method that returns the distance to another point (x1, y1). The distance is calculated by this formula:

$$distance = \sqrt{(x - x1)^2 + (y - y1)^2}$$

- A quadrant() method that returns the quadrant of the point as follows:



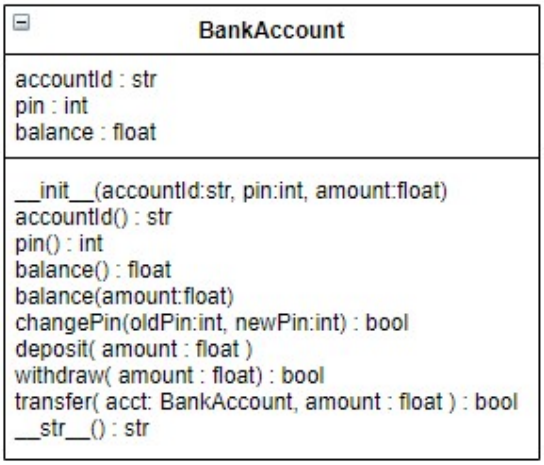
For any point along x or y axis, return 0.

- A __str__() method that returns the string value in this format: (x, y)

Test the Point class, with the following:

i. Create a point object p1, at (5, 1)
ii. Print the coordinates of p1
iii. Move p1 by delta (5, -5)
iv. Create another point p2 at (10 , -10)
v. Print the distance between p1 and p2
vi. Print the quadrants for p1 and p2

4. Write a BankAccount class that models a Bank Account. The class diagram is as follows:



A BankAccount class has 3 instance variables: accountId, pin and balance.

It has a constructor that has 3 parameters to initialize the accountId, pin and the amount. The default balance is $100. It has getter properties for accountId pin, and balance. It has setter property for balance.

It has the following methods:

- A changePin method that has old pin and new pin as parameters. The new pin is updated only if the old pin matches the existing pin. Return true if the change is successful and false otherwise.
- A deposit method with parameter amount which represents the amount to deposit. The method adds the amount to the balance.

- A withdraw method with parameter amount which represents the withdrawal amount. This method deducts the amount from the balance and returns True if there is sufficient balance, and False otherwise.
- A transfer method that has 2 parameters – another bank account to transfer to and the amount to transfer. The method returns True if the transfer is successful and False otherwise.
- The __str__() method returns the accountId and balance as a string.

Test out the BankAccount as follows:

i. Declare a BankAccount object b1 for account 'B1', pin 111, amount 100.
ii. Make a deposit of $100 for b1. Display the details of the account.
iii. Change the pin for b1. Display the outcome of the change.
iv. Declare another BankAccount object b2 for account 'B2', pin 222, amount 100.
v. Make a withdrawal amount of $200 for b2 and display whether the withdrawal is successful.
vi. Transfer $100 from bank account b1 to b2. Display whether the transfer is successful.
vii. Display the bank balances of both accounts.

5. Write a ToDo class that allows a user to record things to do for an event, e.g. travelling trip. The class diagram is as follows:

The class has 2 instance parameters. They are:

- the event (string)
- a list collection to store the to-do actions

The constructor that has an event as parameter. The constructor initializes an empty collection.

It has the following methods:

- A getter property for the event name.
- A addToDo(toDo) method that adds the toDo (string) action to the collection.
- A removeToDoItem(index) method that removes a to-do item using the index position of the todo item. Return True if successful and False otherwise.
- __str__() method that returns a string of all the toDo action items in the following format:

  Event: travelling
  1. Bring passport

2. Change money
3. Bring medicine
…

Test the ToDo class, with the following:

i. Create a ToDo object for an “Orientation camp” event.
ii. Add a few to do actions to the object.
iii. Display the to do list.
iv. Remove a to do action and display the outcome of the removal.