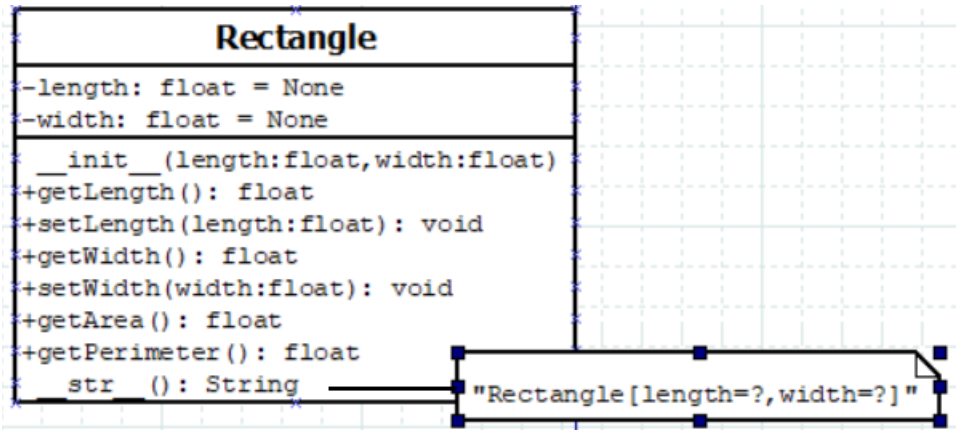
Assignment (BCAC493)

(Basic Class, Constructor, composition, __str__, getter setter methods, @property, @setter)

Q1. A class called `Rectangle`, which models a rectangle with a length and a width (in `float`), is designed as shown in the following class diagram. Write the `Rectangle` class.



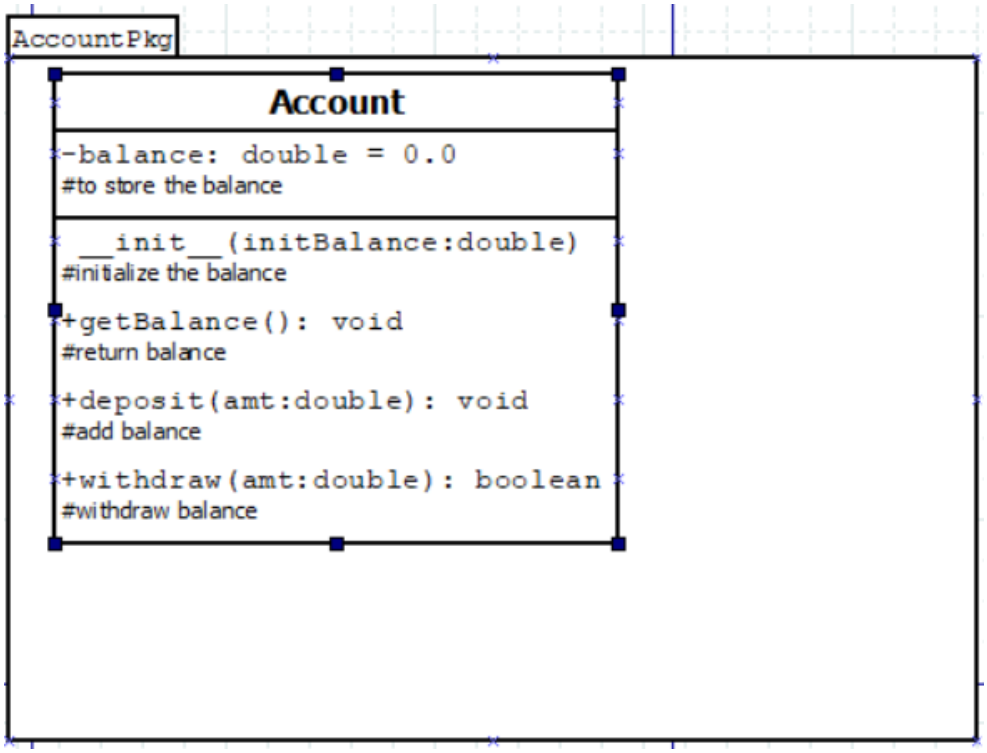
Below is a test driver to test the `Rectangle` class:

```
r1 = Rectangle(1.2, 3.4)
print(r1)
r2 = Rectangle()
print(r2)

r1.setLength(5.6)
r1.setWidth(7.8)
print(r1)
print("length is :", r1.getLength())
print("width is :", r1.getWidth())

print("area is :", r1.getArea())
print("perimeter is :", r1.getPerimeter())
```

The expected output is :

Rectangle [length = 1.2, width = 3.4]
Rectangle [length = 1.0, width = 1.0]
Rectangle [length = 5.6 width = 7.8]
length is: 5.6
width is: 7.8
area is: 43.68
perimeter is: 26.799999999999997

Q2.



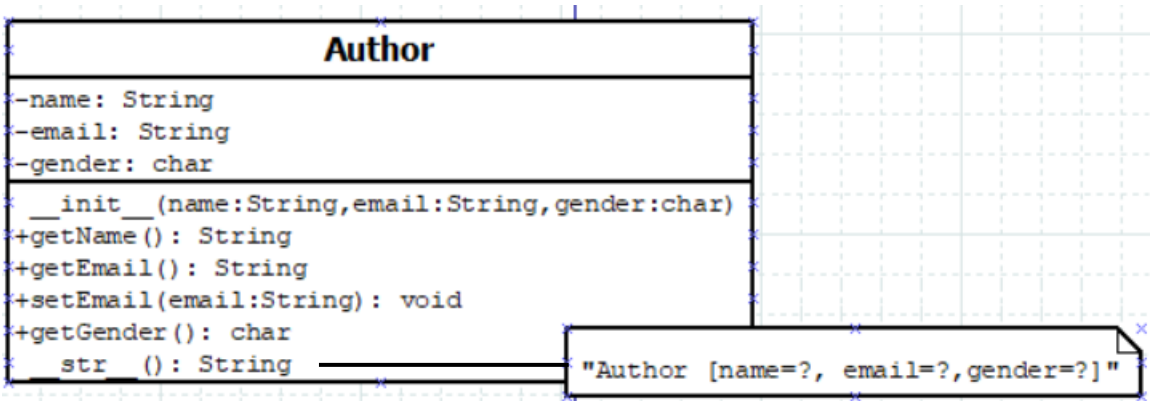
Create the above Account class structure in the package AccountPkg. Whereas using constructor initialize the value of variable balance, getBalance() method return the current balance, deposit() method is used to add amount and withdraw() method is used to deduct amount. The withdraw method should be implemented in such a way that the balance of the bank account should never go below zero.

Create another class TestAccount which acts as a program to create an Account object with an initial balance of hundred. The test program will then add 47 and then subtract 150.Finally the test program must print the balance of the object to the standard output string.

The expected output is:

```
insufficient Balance  147
```

Q3.



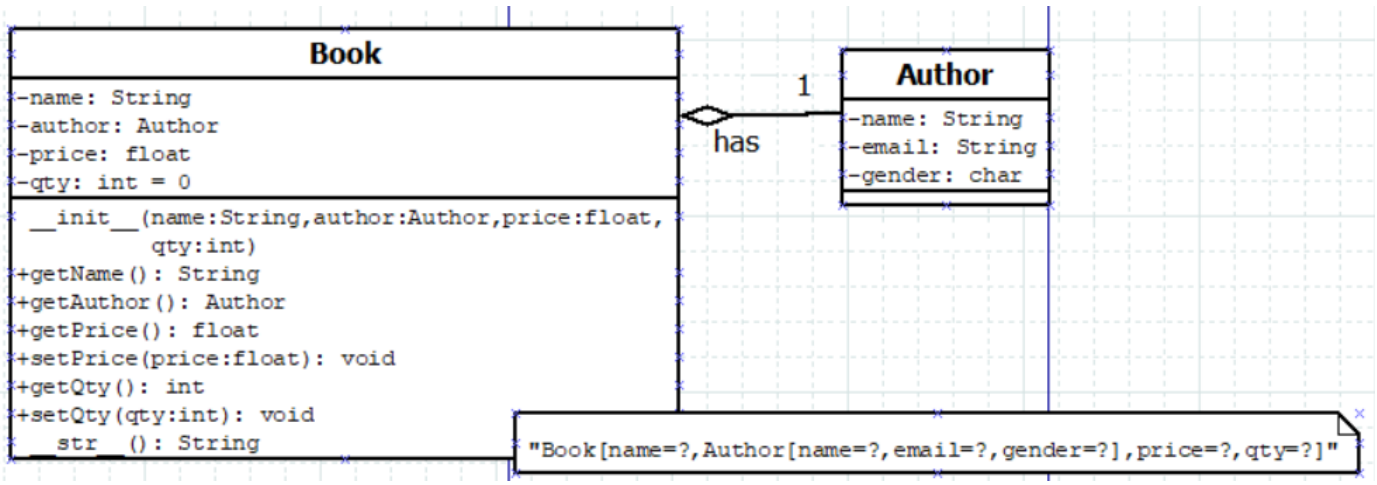
A class called `Author` (as shown in the class diagram) is designed to model a book's author.

Write the `Author` class. Also write a *test driver* called `TestAuthor` to test all the `public` methods, e.g.

```
ahTeck = Author("Tan Ah Teck", "susovan.pan@gmail.com", 'm')
print(ahTeck)  # Test
ahTeck.setEmail("paulTan@nowhere.com")  # Test setter
print("name is: " + ahTeck.getName())  # Test getter
print("email is: " + ahTeck.getEmail())  # Test getter
print("gender is: " + ahTeck.getGender())
```

The expected output is:

```
Author [ name = Tan Ah Teck email = susovan.pan@gmail.com  gender = m]
name is: Tan Ah Teck
email is: susovan.pan@gmail.com
gender is: m
```



A class called Book is designed (as shown in the class diagram) to model a book written by *one* author.

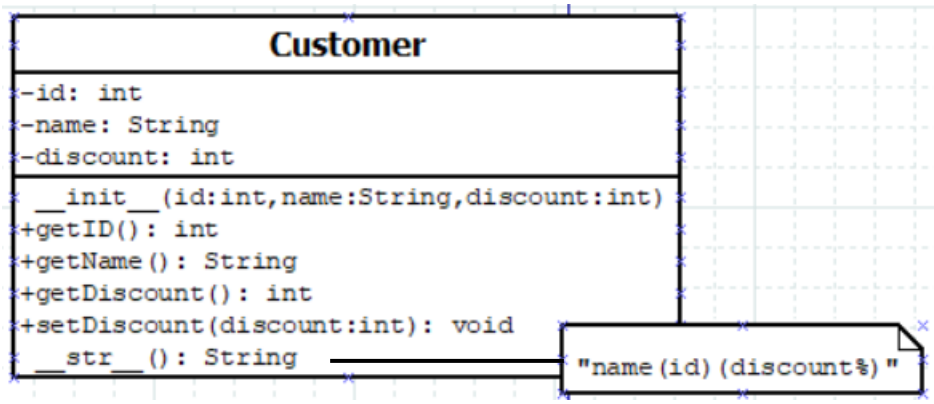
Write the Book class (which uses the Author class written earlier). Also write a test driver called TestBook to test all the public methods in the class Book. Take Note that you have to construct an instance of Author before you can construct an instance of Book. E.g.,

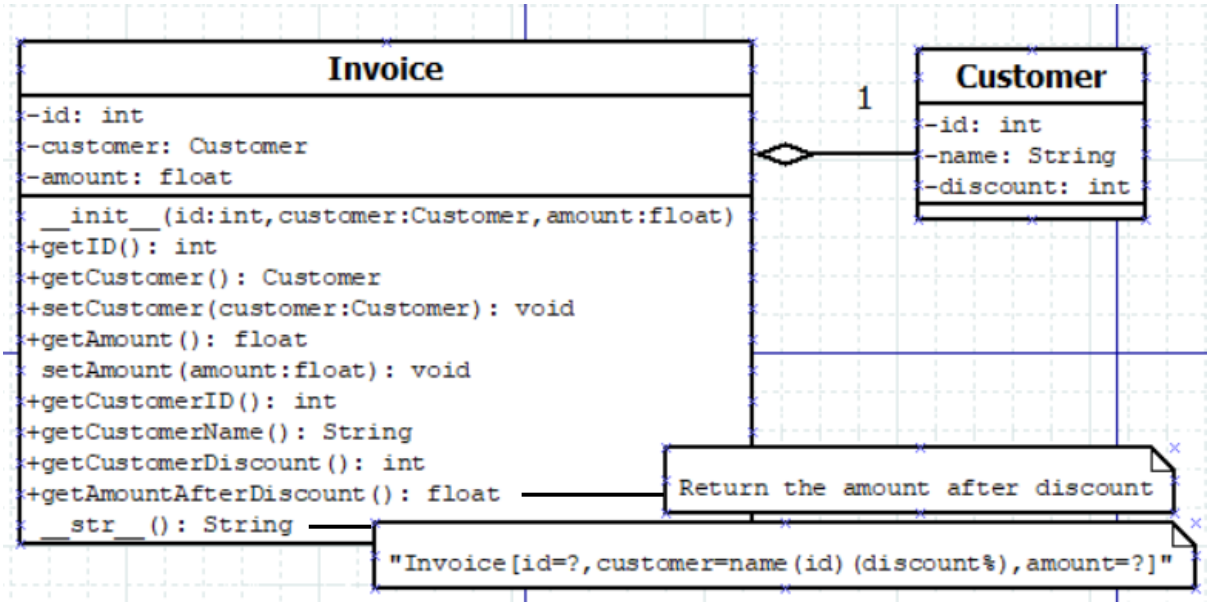
```
ahTeck = Author("Tan Ah Teck", "ahteck@nowhere.com", 'm');
print(ahTeck)  # Author's __str__()

dummyBook = Book("Java for dummy", ahTeck, 19.95, 99)  # Test Book's
Constructor
print(dummyBook)  # Test Book's __str__()

# Test Getters and Setters
dummyBook.setPrice(29.95)
dummyBook.setQty(28)
print("name is: ", dummyBook.getName())
print("price is: ", dummyBook.getPrice())
print("qty is: ", dummyBook.getQty())
print("Author is: ", dummyBook.getAuthor())  # Author's __str__()
print("Author's name is: ", dummyBook.getAuthor().getName())
print("Author's email is: ", dummyBook.getAuthor().getEmail())

# Use an anonymous instance of Author to construct a Book instance
anotherBook = Book("more Java", Author("Paul Tan", "paul@somewhere.com",
'm'), 29.95)
print(anotherBook)  # __str__()
```

The expected output is:

Author [name = Tan Ah Teck email = ahteck@nowhere.com gender = m]
Book[name= Java for dummy, Author [name= Tan Ah Teck, email= ahteck@nowhere.com, gender= m], price= 19.95, qty=99]
name is: Java for dummy
price is: 29.95
qty is: 28
Author is: Author [name = Tan Ah Teck email = ahteck@nowhere.com gender = m]
Author's name is: Tan Ah Teck
Author's email is: ahteck@nowhere.com
Book[name= more Java, Author [name= Paul Tan, email= paul@somewhere.com, gender= m], price= 29.95, qty=0.0]

Q4. A class called Customer, which models a customer in a transaction, is designed as shown in the class diagram. A class called Invoice, which models an invoice for a particular customer and composes an instance of Customer as its instance variable, is also shown. Write the Customer and Invoice classes.





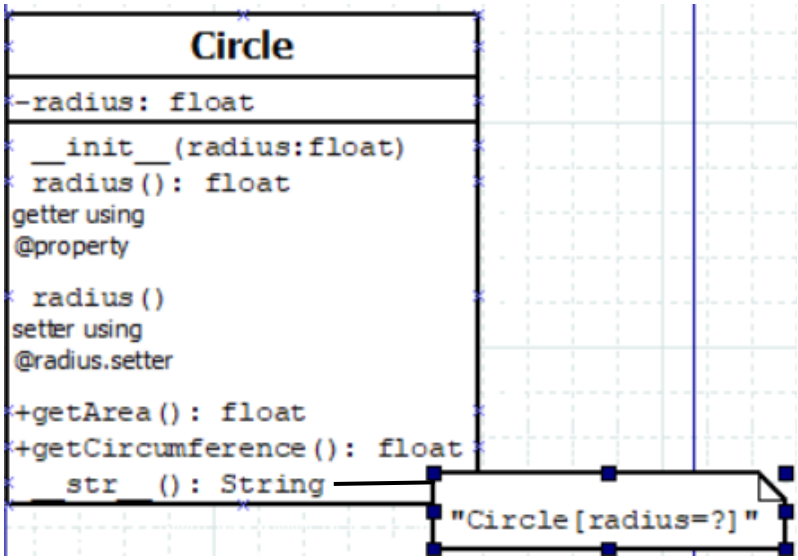
Below is a test driver:

```
c1 = Customer(88, "Susovan Kumar Pan", 20)
print(c1)  # Customer's __str__()
c1.setDiscount(10)
print(c1)
print("id is: ", c1.getID())
print("name is: ", c1.getName())
print("discount is: ", c1.getDiscount())
# Test Invoice class
inv1 = Invoice(101, c1, 800)
print(inv1)

inv1.setAmount(1000)
print(inv1)
print("id is: ", inv1.getID())
print("customer is: ", inv1.getCustomer())  # Customer's __str__()
print("amount is: ", inv1.getAmount())
print("customer's id is: ", inv1.getCustomerID())
print("customer's name is: ", inv1.getCustomerName())
print("customer's discount is: ", inv1.getCustomerDiscount())
print("amount after discount is: ", inv1.getAmountAfterDiscount())
```

The expected output is:

```
Susovan Kumar Pan(88)(20%)
Susovan Kumar Pan(88)(10%)
id is:  88
name is:  Susovan Kumar Pan
discount is:  10
Invoice[id=101,customer=Susovan Kumar Pan(88)(10%),amount=790]
Invoice[id=101,customer=Susovan Kumar Pan(88)(10%),amount=990]
id is:  101
customer is:  Susovan Kumar Pan(88)(10%)
amount is:  990
customer's id is:  88
customer's name is:  Susovan Kumar Pan
customer's discount is:  10
amount after discount is:  980
```

Q5. A class called **`circle`** is designed as shown in the following class diagram.



In the above Circle class, implement radius() using @property decorator.

Let us write a *test program* called `TestCircle`

```
circle1 = Circle()
print(circle1)
print("The circle has radius of ", circle1.radius, "and area of ",
circle1.getArea())

circle2 = Circle(2.0)
print(circle2)
print("The circle has radius of ", circle2.radius, " and area of  ",
circle2.getArea())
circle3 = Circle(6.6)
print(circle3)
```

The expected output is:

```
Circle[radius=1.0]
The circle has radius of  1.0 and area of  3.141592653589793
Circle[radius=2.0]
The circle has radius of  2.0  and area of   12.566370614359172
Circle[radius=6.6]
```