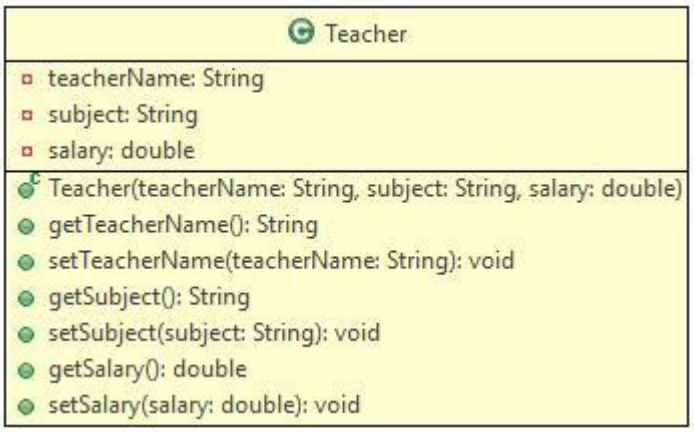
1. Implement the class Teacher based on the class diagram and description given below.



Method Description

Teacher(String teacherName, String subject, double salary)

Initialize the values of all the instance variables appropriately with the values passed

Create a Tester class. Create 4 objects of Teacher class. Create an array of type Teacher store the created objects and display the details of the teachers.

Sample Input and Output

Input

Teacher object	Instance variables	Values
Teacher object1	teacherName	Alex
	subject	Java Fundamentals
	salary	1200L
Teacher object2	teacherName	John
	subject	RDBMS
	salary	800L
Teacher object3	teacherName	Sam
	subject	Networking
	salary	900L
Teacher object4	teacherName	Maria
	subject	Python
	salary	900L

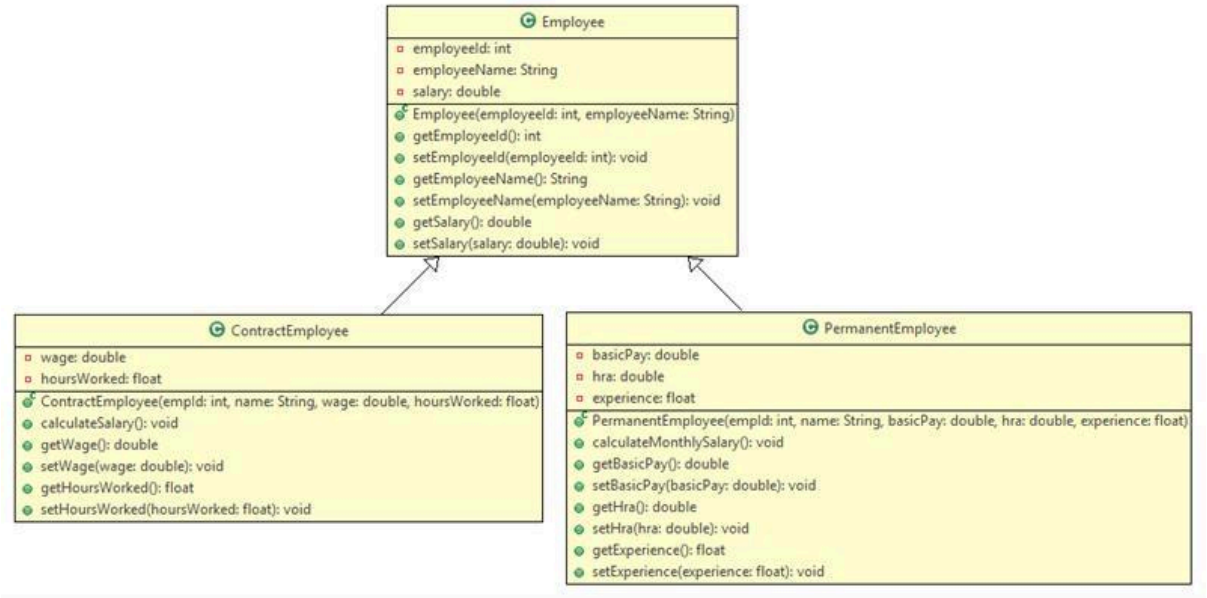
Output

```
Name : Alex, Subject : Java Fundamental, Salary : 1200.0
Name : John, Subject : RDBMS, Salary : 800.0
Name : Sam, Subject : Networking, Salary : 900.0
Name : Maria, Subject : Python, Salary : 900.0
```

2. **Problem Statement**

A construction company wants to keep a record of the employees working in it. There are permanent employees as well as contract employees. Contract employees work on an hourly basis whereas permanent employees are paid a monthly salary. An application needs to be developed for the company for storing the employee details.

Implement the classes based on the class diagram and description given below.



Method Description

Employee

Employee(intemployeeId, String employeeName)

- Initialize the employeeId and employeeName instance variables appropriately with the values passed to the constructor.

Implement the getter and setter methods appropriately.

PermanentEmployee

PermanentEmployee(intempId, String name, double basicPay, double hra, float experience)

- Initialize the employeeId, employeeName, basicPay, hra and experience instance variables appropriately with the values passed to the constructor.

calculateMonthlySalary()

- Calculate the salary of the employee using the formula given below.

  salary = basic pay + hra + variable component

- Variable component is calculated based on the employee's experience according to the table given below.

Experience (in Years)	% of the basic pay
<3	0
>=3 and <5	5
>=5 and <10	7
>=10	12

Implement the getter and setter methods appropriately.

ContractEmployee

ContractEmployee(intempId, String name, double wage, float hoursWorked)

- Initialize the employeeId, employeeName, wage and hoursWorked instance variables appropriately with the values passed to the constructor.

calculateSalary()

- Calculate the salary of the employee using the formula given below.

  salary = hoursWorked * wage

Implement the getter and setter methods appropriately.

Test the functionalities using the provided Tester class.

Input and Output

For PermanentEmployee

Input

Instance variables	Values
employeeId	711211
employeeName	Rafael
basicPay	$1850
hra	$115
experience	3.5

Output

```
Hi Rafael, your salary is $2057.5
```

For ContractEmployee

Input

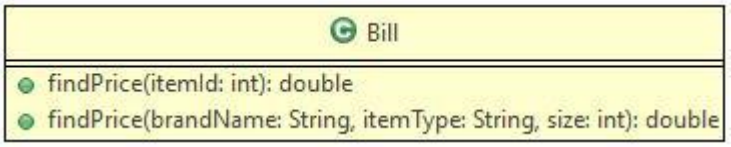
Instance variables	Values
employeeId	102
employeeName	Jennifer
wage	$16
hoursWorked	90

Output

```
Hi Jennifer, your salary is $1440.0
```

3. **Problem Statement**

The Bill class is used to find the price of items for calculation. Implement a class Bill based on the class diagram and description given below.



The details of the items are given below.

Brand Name	Item Id	Item Type	Size	Price
Puma	1001	T-shirt	34	$25
			36	
	1002	Skirt	38	$20
			40	
Reebok	1003	T-shirt	34	$23
			36	
	1004	Skirt	38	$18
			40	

Method Description

findPrice(intitemId)

- Find and return the price based on the itemId using the table given above.
- If the itemId passed to method is invalid, return the price as 0.

findPrice(String brandName, String itemType, int size)

- Find and return the price based on the brandName, itemType and size using the table given above.
- If any invalid details are passed to the method, return the price as 0.

Test the functionalities using the provided Tester class.

Sample Input and Output

For findPrice(intitemId)

Input

Attribute	Value
itemId	1001

Output

```
Price of the selected item is $25.0
```

For findPrice(String brandName, String itemType, int size)

Input

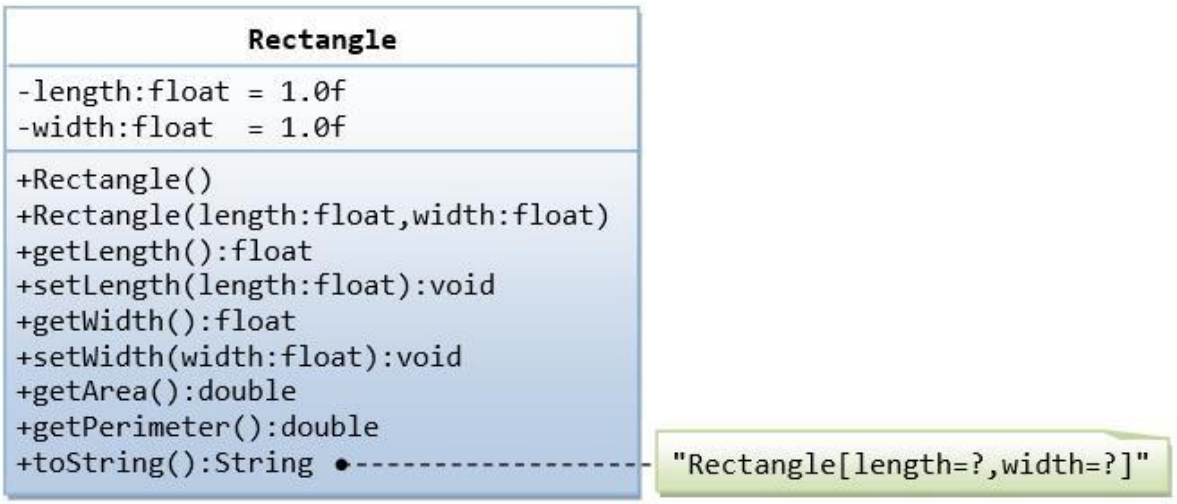
Instance Variables	Values
brandName	Reebok
itemType	T-shirt
size	34

Output

```
Price of the selected item is $23.0
```

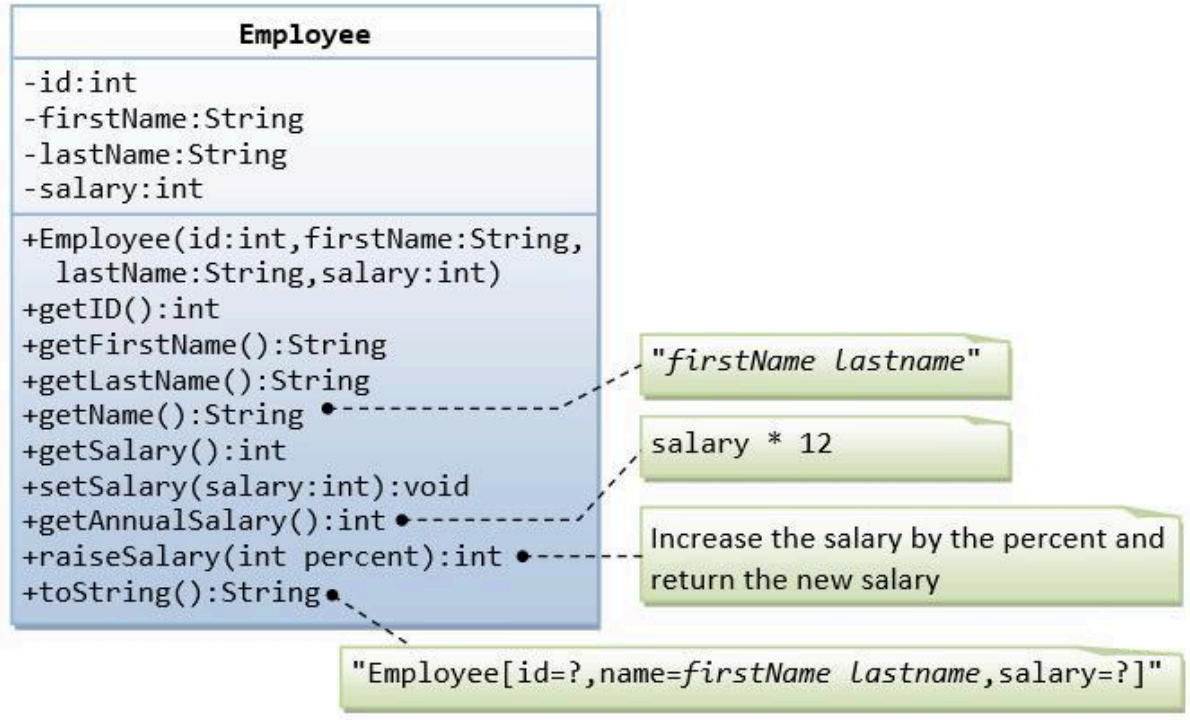
Problem-1

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.



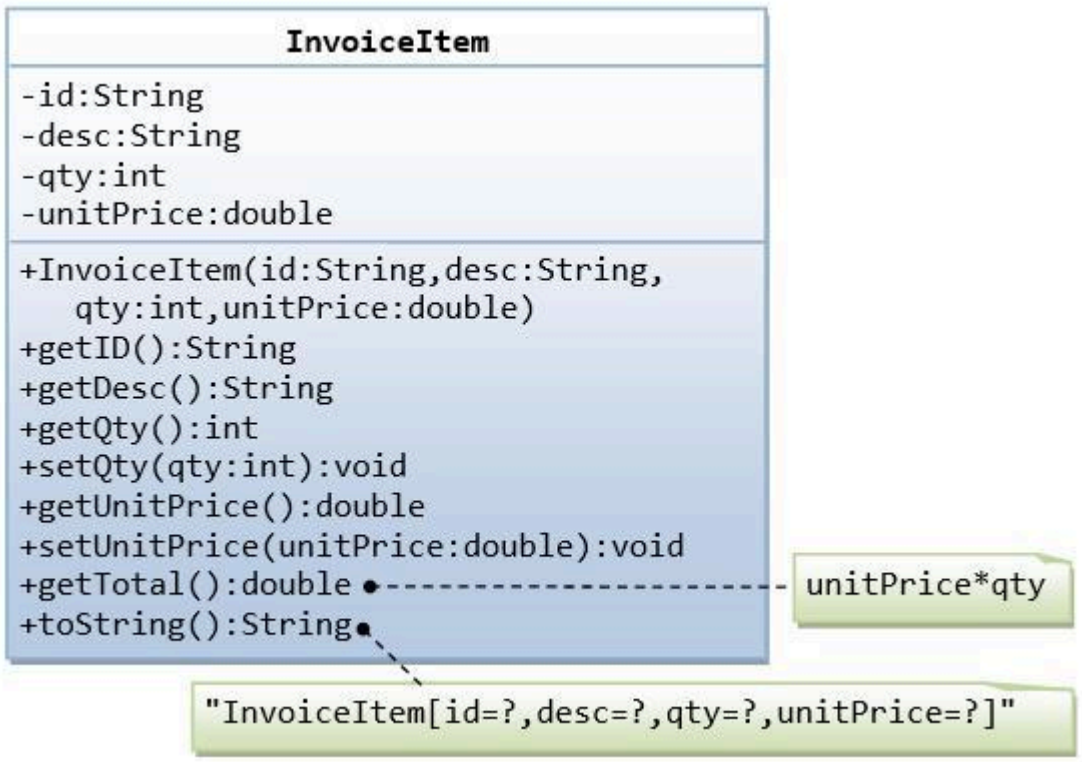
Problem-2

A class called `Employee`, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method `raiseSalary(percent)` increases the salary by the given percentage. Write the `Employee` class.



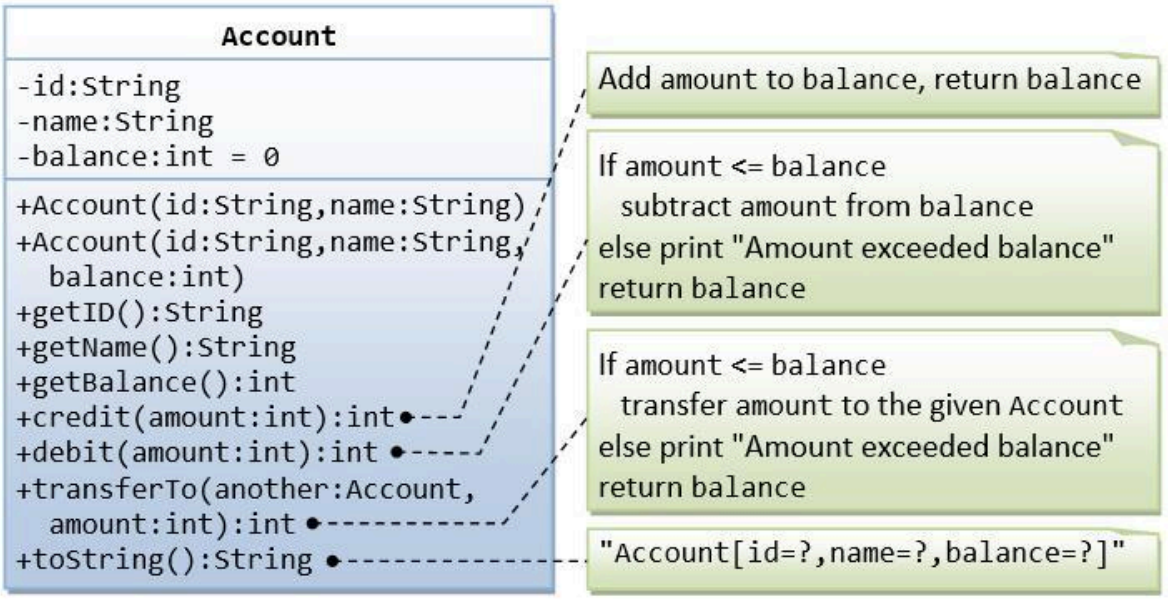
Problem-3

A class called InvoiceItem, which models an item of an invoice, with ID, description, quantity and unit price, is designed as shown in the following class diagram. Write the InvoiceItem class.



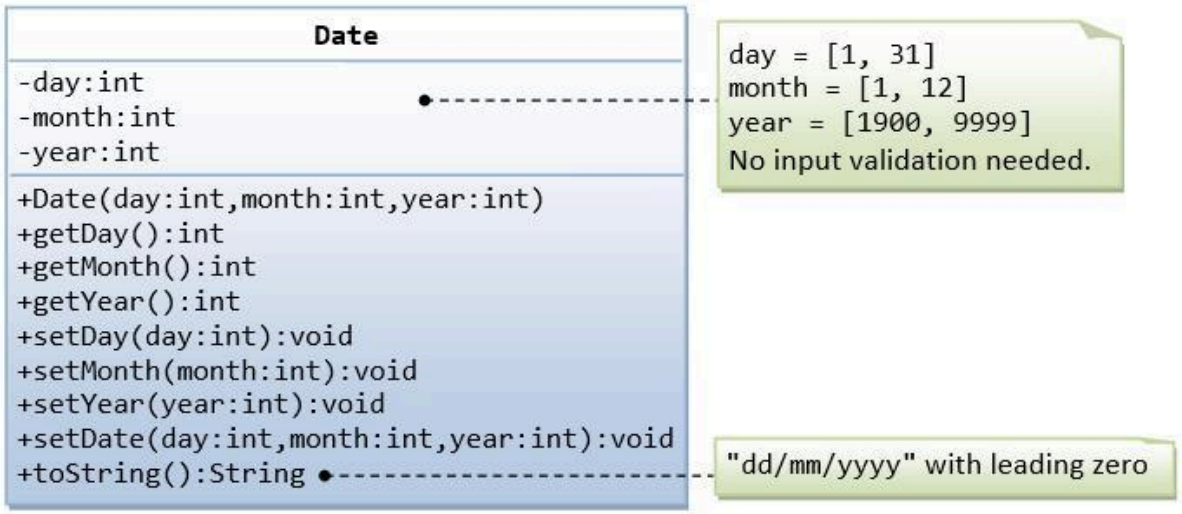
Problem-4

A class called Account, which models a bank account of a customer, is designed as shown in the following class diagram. The methods credit(amount) and debit(amount) add or subtract the given amount to the balance. The method transferTo(anotherAccount, amount) transfers the given amount from this Account to the given anotherAccount. Write the Account class.

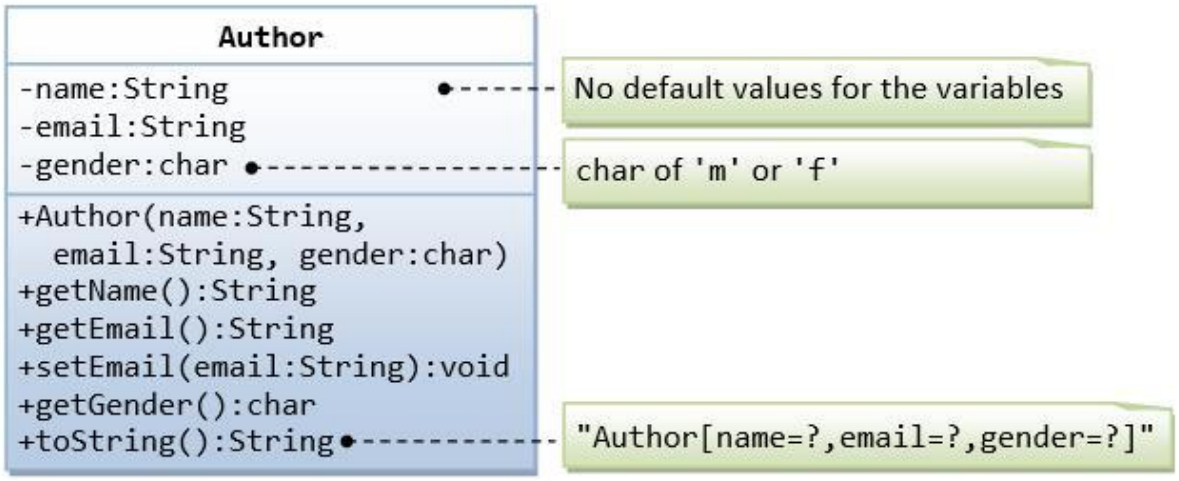


Problem-5

A class called Date, which models a calendar date, is designed as shown in the following class diagram. Write the Date class.



Problem-6



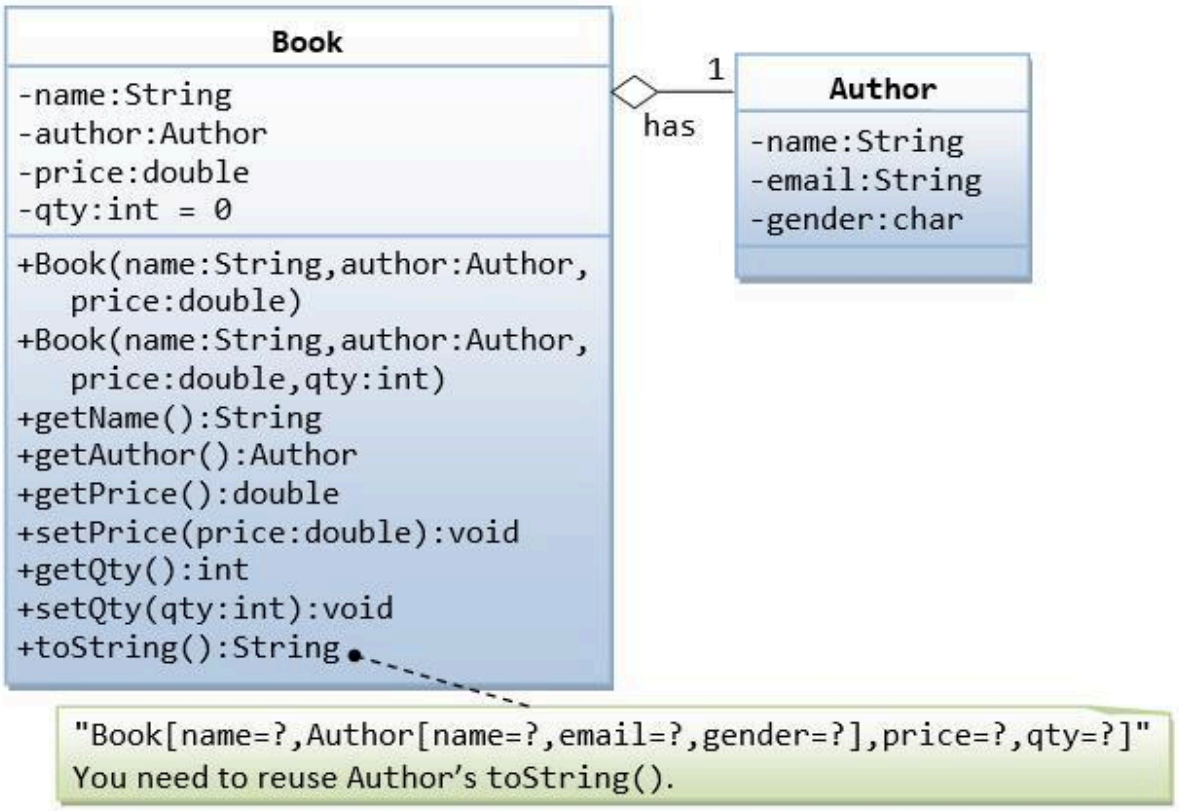
A class called Author (as shown in the class diagram) is designed to model a book's author. It contains:

- Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');
- One constructor to initialize the name, email and gender with the given values;

  public Author (String name, String email, char gender) {.....}

  (There is no default constructor for Author, as there are no defaults for name, email and gender.)

- public getters/setters: getName(), getEmail(), setEmail(), and getGender();
  (There are no setters for name and gender, as these attributes cannot be changed.)
- A toString() method that returns "Author[name=?,email=?,gender=?]", e.g., "Author[name=Tan Ah Teck,email=ahTeck@somewhere.com,gender=m]".



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

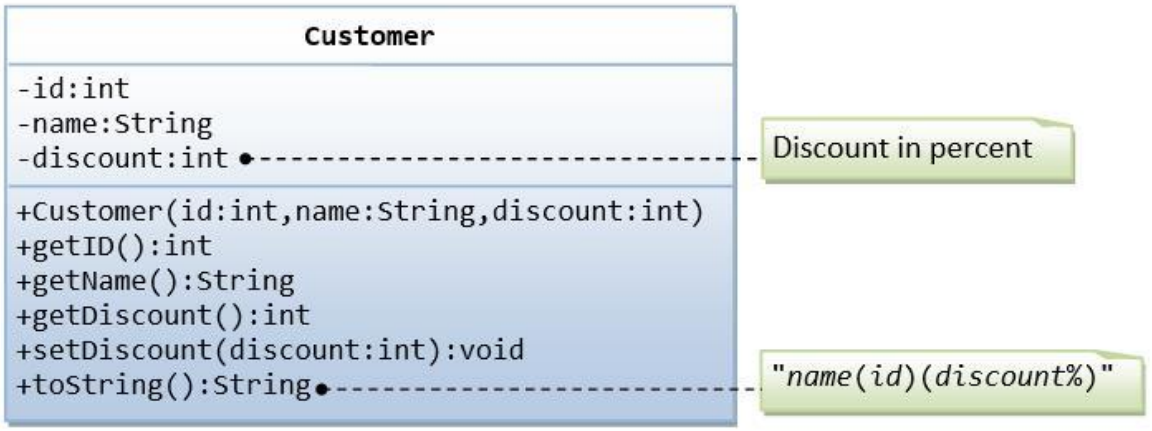
- Four private instance variables: name (String), author (of the class Author you have just created, assume that a book has one and only one author), price (double), and qty (int);
- Two constructors:
- public Book (String name, Author author, double price) { ..... }

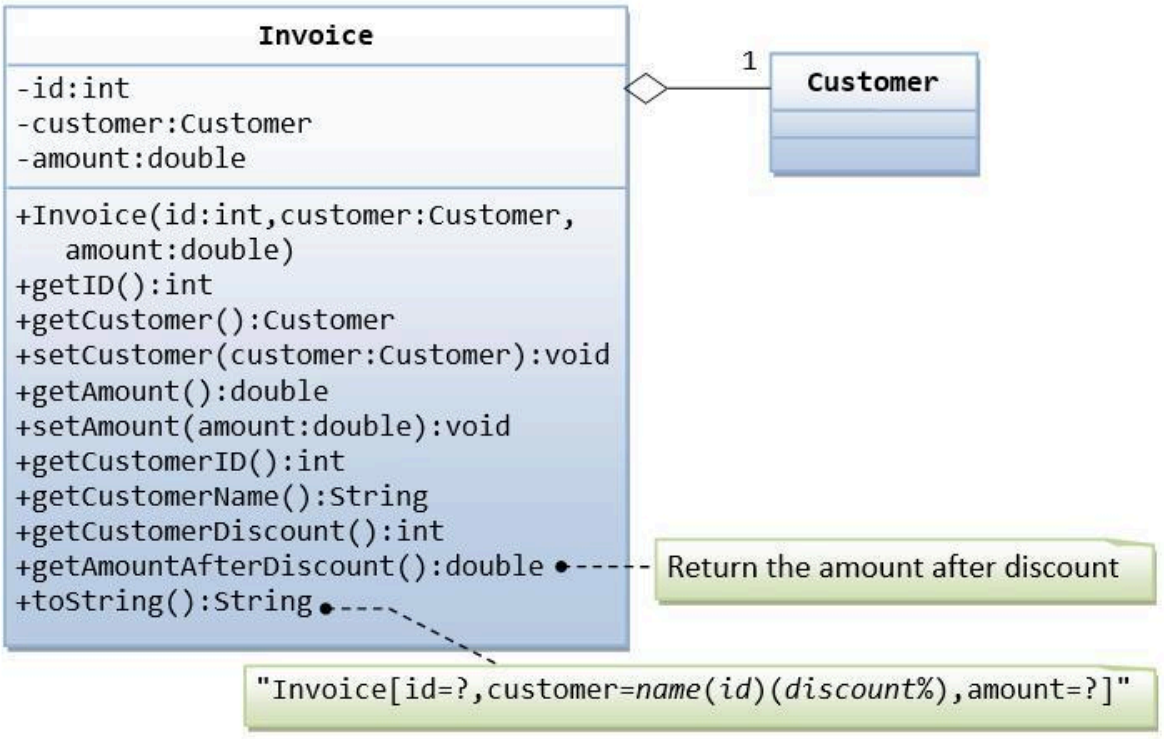
  public Book (String name, Author author, double price, intqty) { ..... }

- public methods getName(), getAuthor(), getPrice(), setPrice(), getQty(), setQty().
- A toString() that returns "Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?". You should reuse Author's toString().

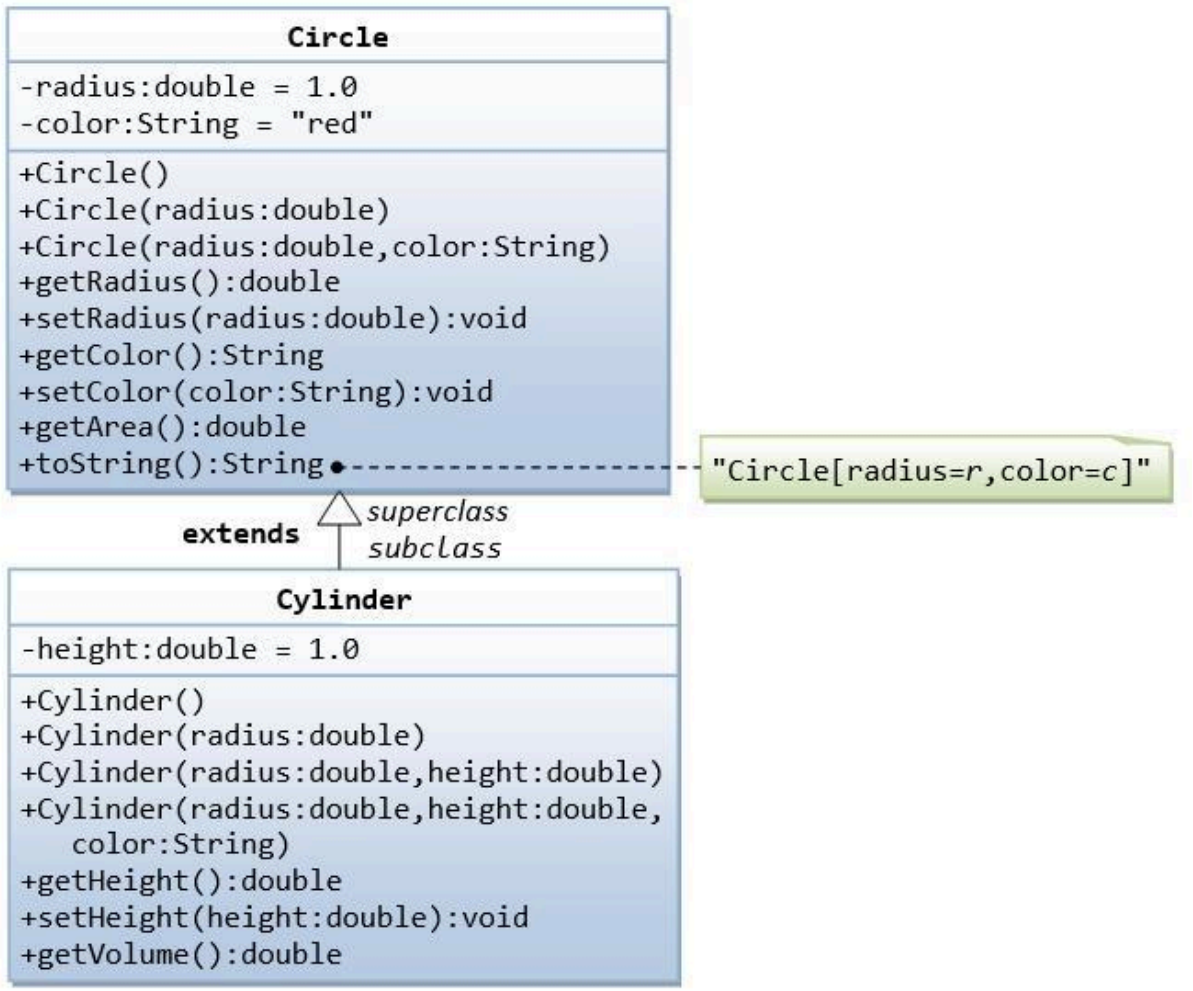
Problem-7

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.


Customer
-id:int
-name:String
-discount:int
Discount in percent
+Customer(id:int,name:String,discount:int)
+getID():int
+getName():String
+getDiscount():int
+setDiscount(discount:int):void
+toString():String
"name(id)(discount%)"


Invoice
-id:int
-customer:Customer
-amount:double
+Invoice(id:int,customer:Customer,
amount:double)
+getID():int
+getCustomer():Customer
+setCustomer(customer:Customer):void
+getAmount():double
+setAmount(amount:double):void
+getCustomerID():int
+getCustomerName():String
+getCustomerDiscount():int
+getAmountAfterDiscount():double
+toString():String
1
Customer
Return the amount after discount
"Invoice[id=?,customer=name(id)(discount%),amount=?]"

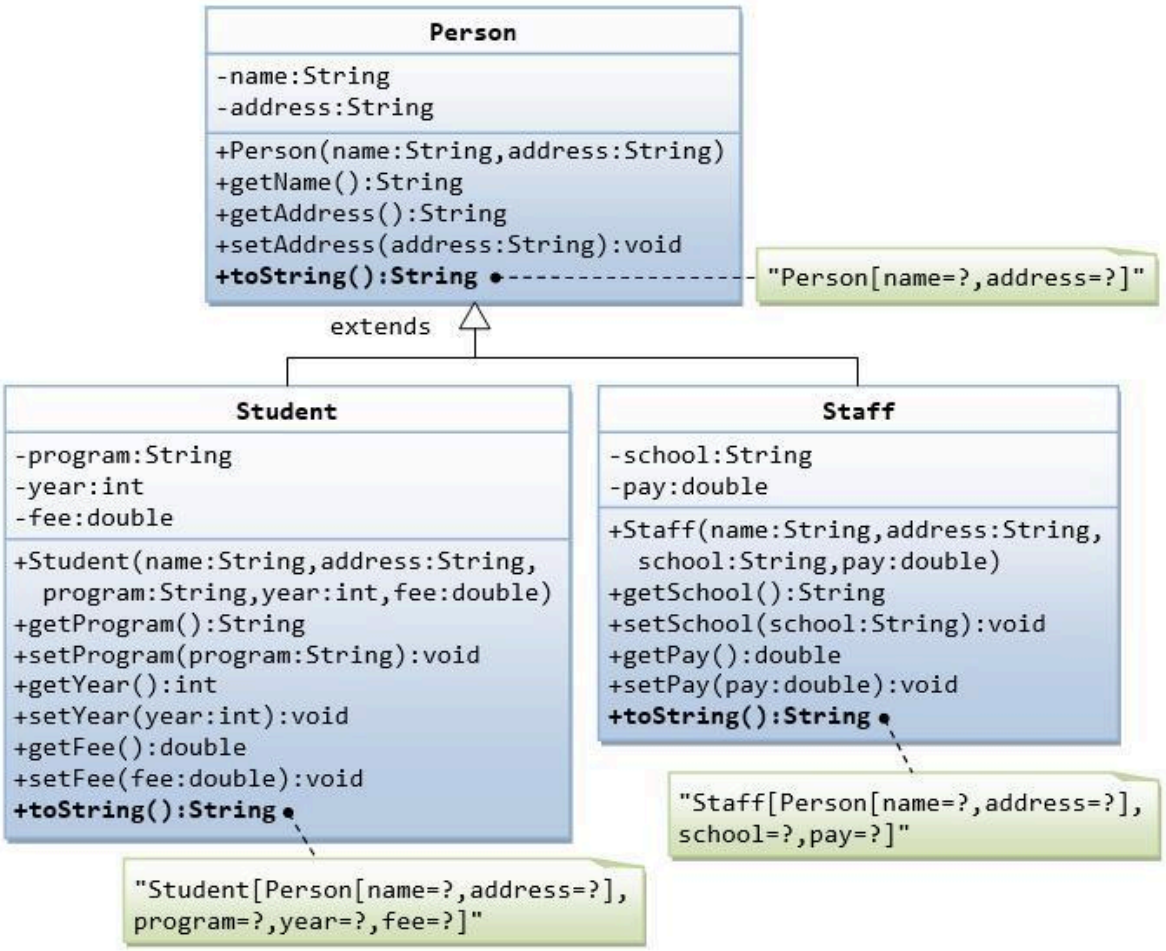
Problem-8



In this exercise, a subclass called `Cylinder` is derived from the superclass `Circle` as shown in the class diagram (where an an arrow pointing up from the subclass to its superclass). Study how the subclass `Cylinder` invokes the superclass' constructors (via `super()` and `super(radius)`) and inherits the variables and methods from the superclass `Circle`.

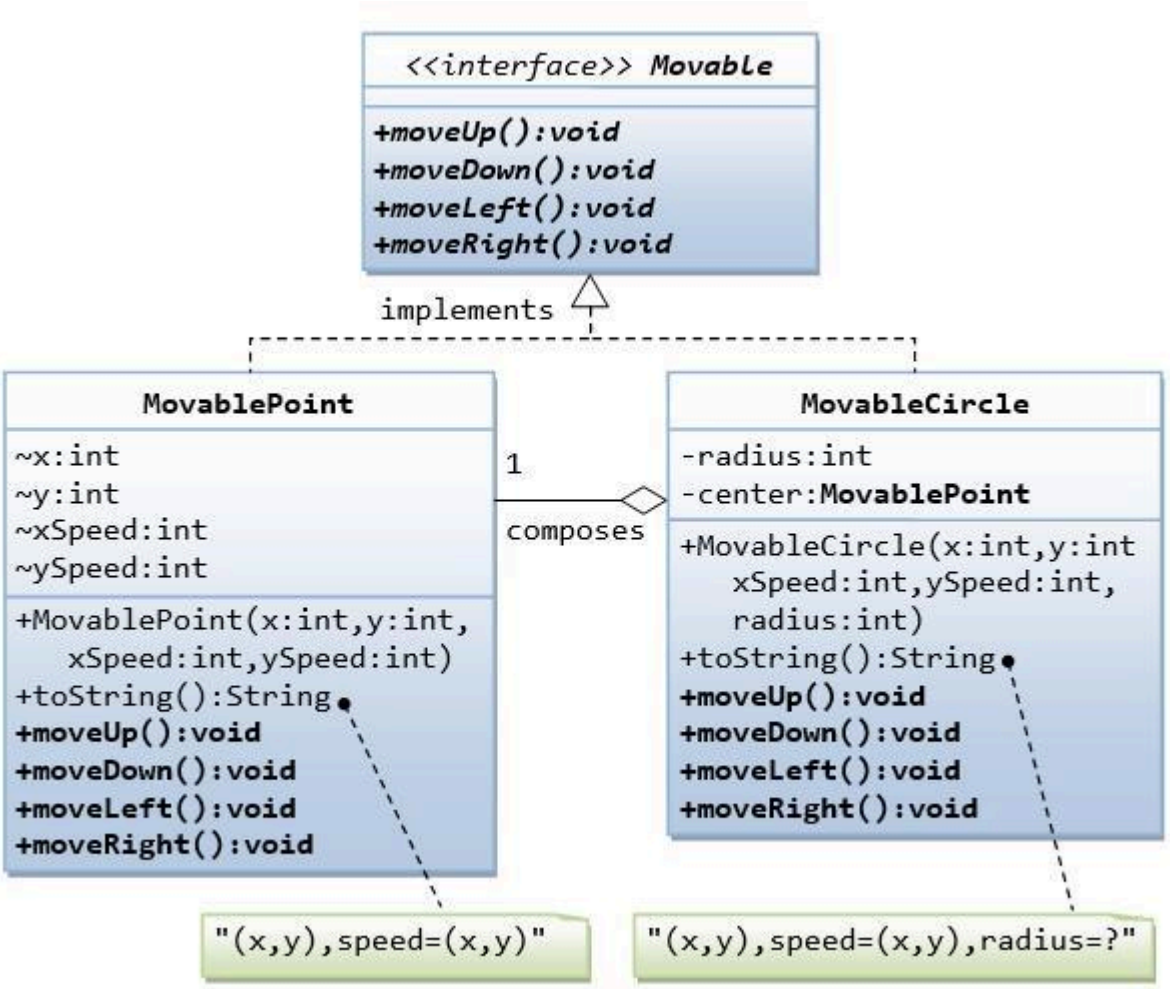
Problem-9

Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation `@Override`.



Problem-10

Write an interface called Movaable, which contains 4 abstract methods moveUp(), moveDown(), moveLeft() and moveRight(), as shown in the class diagram. Also write the implementation classes called MovablePoint and MovableCircle. Mark all the overridden methods with annotation @Override.


<<interface>> Movable
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
implements
MovablePoint
~x:int
~y:int
~xSpeed:int
~ySpeed:int
+MovablePoint(x:int,y:int,
xSpeed:int,ySpeed:int)
+toString():String
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
1
composes
MovableCircle
-radius:int
-center:MovablePoint
+MovableCircle(x:int,y:int
xSpeed:int,ySpeed:int,
radius:int)
+toString():String
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
"(x,y),speed=(x,y)"
"(x,y),speed=(x,y),radius=?"