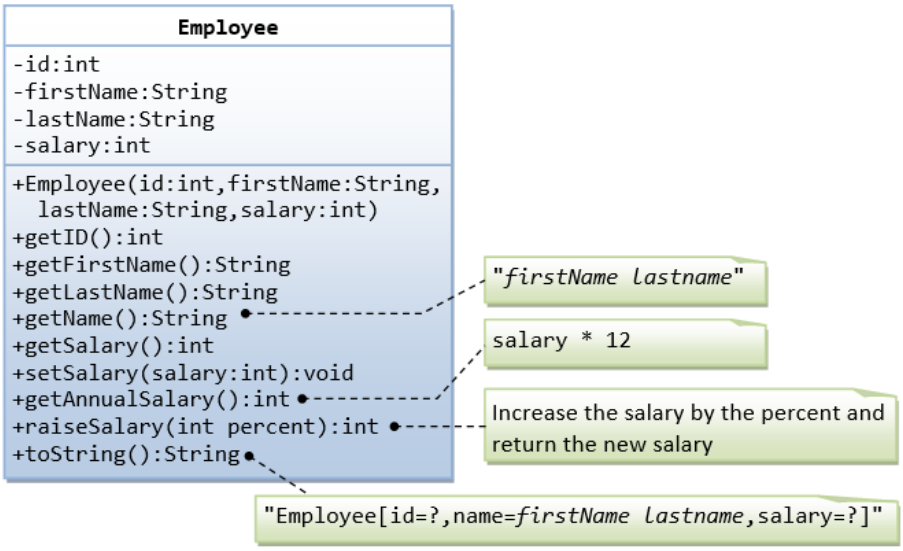
Assignment 2: Basic UML, classes, Object-Oriented programming, Inhertiance, Polymorphism, Interface

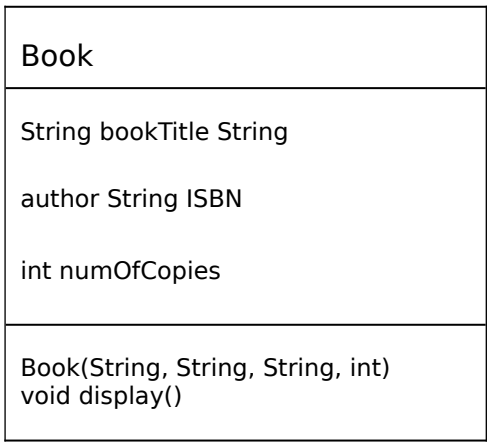
Q1: Implement following UML diagram, Write a program to test circle class.



Q2. Create a book store application which will help a book store to keep the record of its books. For each book, the application will have the Book Title, Book Author, Book ISBN along with the number of copies for each book. The system will allow you to display all books, order new/existing books and sell books. With sell or order of existing books, number of copies will decrease/increase. With order of new book, a new book entry will be added to the system.

Here is what you need to do.

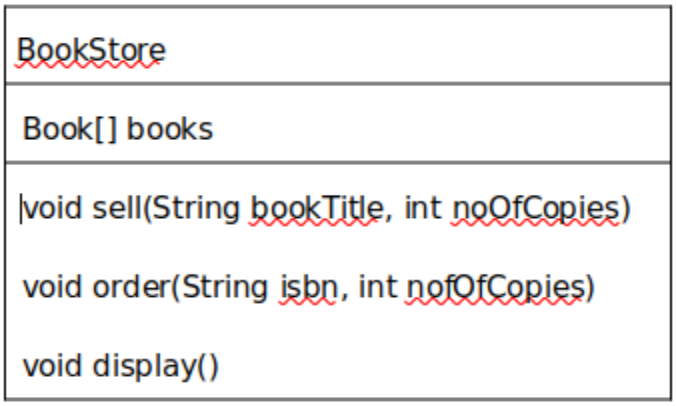
1) Create the following Book class.



Here is what you need to do.

1) Create the following Book class.
2) display() method will display the book info in “Title – Author – ISBN – Quantity” format.

2) Create another class “BookStore” which should contain all the book objects. For now use an array of Book type and assume you can have maximum 10 different books but each could have multiple copies.



a. sell(String, int) method will search for the book in books array using the bookTitle value. If the book is found in the list, number of copies will decrease. If the book is not found a message should display that book is not found

b. order(String, int) method will order book for the book store. You have to handle both new book and existing book scenario.
i. First search for the book in "books" array using the isbn value.
ii. If the book is found in the list (which means the book already exists in the system), number of copies will increase.
iii. If the book is not found (which means the book does not exists in the system and you need to order new book), a new book entry will be added to the books array.

c. display() method will display info of all books in "books" array "Title – Author – ISBN – Quantity" format. Use Book class's display() method to display each book's info.

3) Now create class "BookStoreApp" which should contain the main method. In main method create an object of BookStore class and then Test the different functionality of BookStore

Q3. Assignment on Inheritance, Method overriding, Method overloading

A Banking System

Create a Banking System, where a user can create new account, deposit money, withdraw money and check the balance.
For Simplicity, we will create the system for one customer. First the application will require the user to create an account. After creating the account, he should have options to deposit, withdraw money, and check balance as many times as he wants (until he exits the system).

When a user opens/creates an account, he has the option to open a "Saving Account" or a "Current Account".
See below for the requirements of Current and Savings account.

4) Saving account: A saving account allows user to accumulate interest on funds he has saved for future needs. Interest rates can be compounded on a daily, weekly, monthly, or annual basis. Saving account required a minimum balance. For our purpose let's assume the minimum balance is 5000 and interest rate is 5% (From savings account, user is only allowed to withdraw a maximum amount of money that available to his account)

5) Current account: Current account offers easy access to your money for your daily transactional needs and helps keep your cash secure. You need a trading license to open a Current account.The have overdraft option with current account

What you need to do:

Create the Account class:

1. This will have name, accountNumber, and accountBalance instance variables;
2. One method name deposit(double).

Create a SavingsAccount class:

1. This class is a subclass of Account class.
2. This will have two additional instance variables
   1. One is interest and initialized to 5.
   2. Another instance variable for maximum withdraw amount limit(that is equal to amount int the account)
   3. Implement getBalance() method.
      This method will add the total interest to the accountBalance value and return the value but it won't change the accountBalance value.

4. Implement withdraw(double) method. This method will allow to withdraw money if the withdraw amount is less than the maximum amount limit and doesn't set the balance less than minimum balance after withdraw.

Create a CurrentAccount class:

1. Should extend the Account class
2. Add an instance variable tradeLicenseNumber and overdraft
3. Implement getBalance() method. This method will return the accountBalance.
4. Implement withdraw(double) method. This method will allow to withdraw money if the withdraw amount doesn't exceed the accountBalance value (plus overdraft)

Q4. An Employee Record System

You need to implement the Employee records of a company. The Company has 3 types of employee;

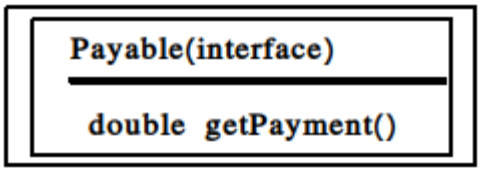
1) Salaried employee-> This type of employees are paid a fixed weekly salary regardless of the number of hours worked.
2) Hourly employee -> They are paid by the hour. They have an hourly rate and their payment will depend on how many hours they worked. The more they work, the more they will be paid. So, the salary will be [hour worked per week* hourly rate].
3) Commission employee-> They are paid a percentage of their sales. If their percentage is "a" and total weekly sale is "b", the total weekly salary will be [a*b/100];

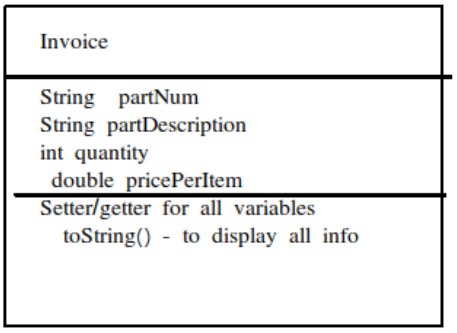
1. Implement the system, where you can get the weekly salary of any employee,
2. The company also wants option to increase the salary of a particular type of employee by a specific percentage.
3. use Array List to store the list of the employee.

Q5 The Payment System

Now we need to implement **the Payment system** for that company. The company wants to handle the **employee** payment and **invoice** in the same application. As Employee and Invoice are totally unrelated objects, we cannot use the same class hierarchy; we have to use an interface called "***Payable***" and implement that in both ***Employee*** class and ***Invoice*** class. So, do the following.

1) Implement the following **Payable** interface and **Invoice** class.





2) Update both ***Invoice*** and ***Employee*** class and "*implements*" ***Payable*** interface. Make necessary code changes.
   - ***getPayment()*** should display all info of respective class and the total payment.

3) Create the application class. In main method **create objects** of **each type of Employees & Invoice** class and call the ***getPayment()*** method.

4) Write a main class to test the code