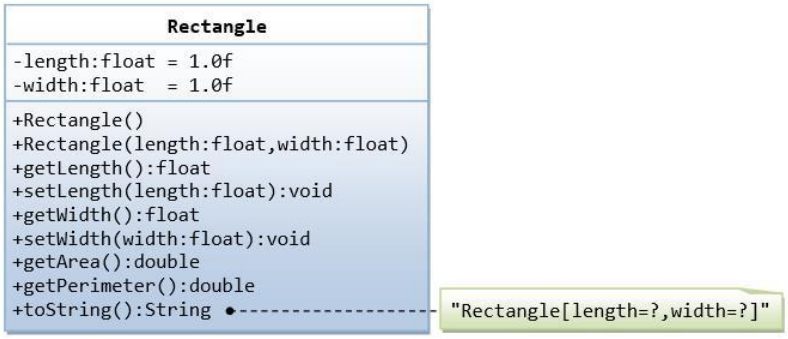
Capstone project Day2 Asiignment-2

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.



Answer:

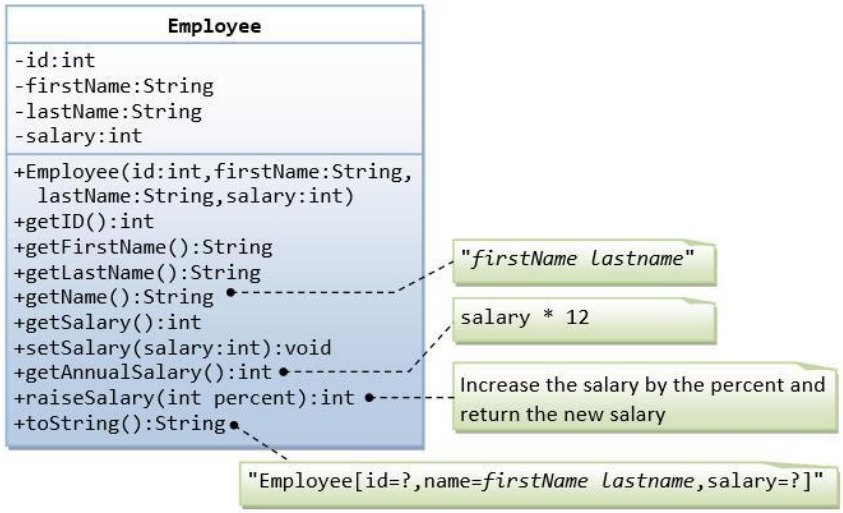
```
public class Rectangle {

	private float  Length;

	private float Width;

	public double getLength() {

		return Length;

	}

	public void setLength(float Length) {

		this.Length = Length;

	}

	public double getWidth() {

		return Width;

	}

	public void setWidth(float Width) {

		this.Width = Width;

	}
```

```
    Rectangle(){
        Length=Width=1.0f;
    }
    Rectangle (float length, float width){
        Length = length;
        Width=width;
    }
    public void getArea() {
        double area = Length*Width;
        System.out.println("Area = " +area);
    }
    public void getPerimeter() {
        double P = (Length+Width)*2;
        System.out.println("Perimeter = " +P);
    }
    public String toString() {
        return "Rectangle [Length = " + Length + " , Width =" +Width + " ]";
    }
    public static void main(String[] args) {
        Rectangle rec = new Rectangle (1.0f,1.0f);
        System.out.println(rec);
        rec.getArea();
        rec.getPerimeter();
    }
}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac Rectangle.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java Rectangle
Rectangle [Length = 1.0 , Width =1.0 ]
Area = 1.0
Perimeter = 4.0
```

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.



Answer:

```java
import java.util.Scanner;

public class Employee {

	private int ID;
	private String FirstName;
	private String LastName;
	private int Salary;

	public int getID() {
		return ID;
```

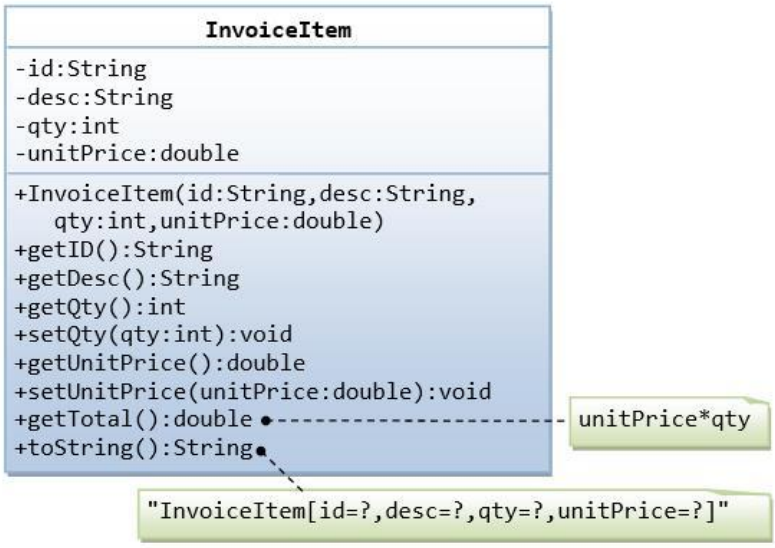
```
	}

	public void setID(int ID) {
		this.ID = ID;
	}

	public String getFirstName() {
		return FirstName;
	}

	public void setFirstName(String FirstName) {
		this.FirstName = FirstName;
	}

	public String getLastName() {
		return LastName;
	}

	public void setLastName(String LastName) {
		this.LastName = LastName;
	}

	public int getSalary() {
		return Salary;
	}

	public void setSalary(int Salary) {
		this.Salary = Salary;
	}

	Employee(int ID, String FirstName, String LastName, int
Salary){
		this.ID=ID;
		this.FirstName=FirstName;
		this.LastName=LastName;
		this.Salary=Salary;
	}

	public void getName() {
		String Name = FirstName+ " " + LastName;
		System.out.println("Name= " +Name);
	}

	public void getAnnualSalary() {
		int as = Salary*12;
		System.out.println("Annual Salary= " +as);
	}

	public void getRaiseSalary(double percent) {
```

```
            double rs = Salary*12*percent;
            System.out.println("New Salary= " +rs);
      }

      public String toString() {
            return "Employee [ID = " + ID + " , Name ="
+FirstName + " " + LastName + ",Salary = " +Salary +"]";
      }

      public static void main(String[] args) {
            Scanner sc = new Scanner (System.in);
            System.out.println("Enter Employee ID");
            int id = sc.nextInt();
            System.out.println("Enter Employee FirstName");
            String fn = sc.next();
            System.out.println("Enter Employee LastName");
            String ln = sc.next();
            System.out.println("Enter Employee Salary");
            int s = sc.nextInt();
            Employee emp = new Employee(id,fn,ln,s);
            System.out.println(emp);
            emp.getName();
            emp.getAnnualSalary();
            emp.getRaiseSalary(0.15);
      }

}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac Employee.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java Employee
Enter Employee ID
1
Enter Employee FirstName
Deepika
Enter Employee LastName
Naik
Enter Employee Salary
350000
Employee [ID = 1 , Name =Deepika Naik,Salary = 350000]
Name= Deepika Naik
Annual Salary= 4200000
New Salary= 630000.0
```

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.



Answer:

```
class Invoice{
        private int id;
   private String description;
   private int quantity;
   private double unitPrice;
   public Invoice(int id, String description, int quantity, double unitPrice) {
                super();
                this.id = id;
                this.description = description;
                this.quantity = quantity;
                this.unitPrice = unitPrice;
        }

        public int getId() {
                return id;
```

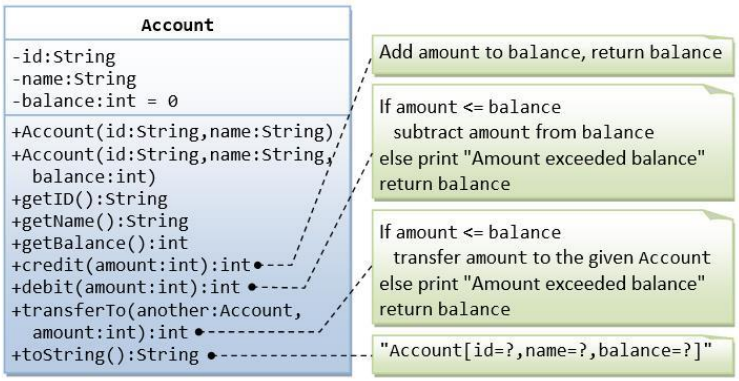
```
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getUnitPrice() {
        return unitPrice;
    }
    public void setUnitPrice(double unitPrice) {
        this.unitPrice = unitPrice;
    }
    public double calculateTotalPrice() {
  return quantity * unitPrice;
}
```

```
}
public class InvoiceItem {
        public static void main(String[] args) {
             Invoice[] items = new Invoice[3];
            items[0] = new Invoice(1, "Item A", 2, 15.99);
            items[1] = new Invoice(2, "Item B", 3, 24.99);
            items[2] = new Invoice(3, "Item C", 1, 9.99);
            // Print the details of each InvoiceItem
            for (Invoice item : items) {
               System.out.println("ID: " + item.getId());
               System.out.println("Description: " + item.getDescription());
               System.out.println("Quantity: " + item.getQuantity());
               System.out.println("Unit Price: " + item.getUnitPrice());
               System.out.println("Total Price: " + item.calculateTotalPrice());
               System.out.println();
            }
   }
}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac InvoiceItem.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java InvoiceItem
ID: 1
Description: Item A
Quantity: 2
Unit Price: 15.99
Total Price: 31.98

ID: 2
Description: Item B
Quantity: 3
Unit Price: 24.99
Total Price: 74.97

ID: 3
Description: Item C
Quantity: 1
Unit Price: 9.99
Total Price: 9.99
```

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.



Answer:

```
class Account {

    private String accountNumber;
    private double balance;
    private String Name;

    public String getName() {
          return Name;
     }

     public void setName(String name) {
          this.Name = name;
     }

     public String getAccountNumber() {
            return accountNumber;
     }

     public void setAccountNumber(String accountNumber) {
          this.accountNumber = accountNumber;
     }

     public void setBalance(double balance) {
          this.balance = balance;
     }
```

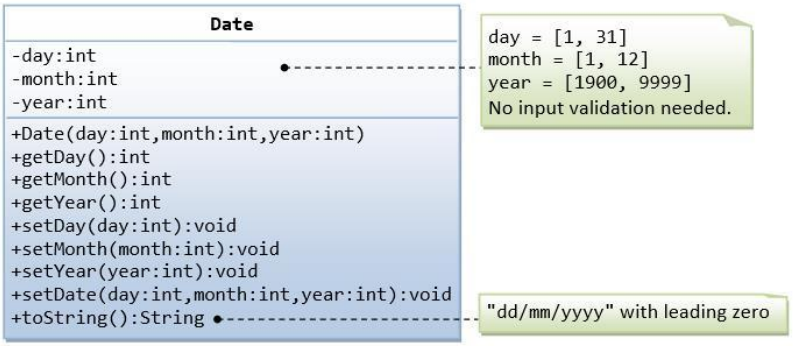
```
      public double getBalance() {
         return balance;
     }

     Account(String accountNumber, String Name) {
         this.accountNumber = accountNumber;
         this.Name = Name;
     }

     Account(String accountNumber, String Name, double
initialBalance) {
         this.accountNumber = accountNumber;
         this.balance = initialBalance;
     }

     public void credit(double amount) {
         if (amount > 0) {
             balance += amount;
             System.out.println("Credited Amount : " + amount +
"/- New balance: " + balance +"/-");
         } else {
             System.out.println("Invalid credit amount...");
         }
     }

     public void debit(double amount) {
         if (amount > 0) {
             if (balance >= amount) {
                 balance -= amount;
                 System.out.println("Debited Amount : " +
amount + "/- New balance: " + balance+"/-");
             } else {
                 System.out.println("Amount exceeded. Your
Current balance: " + balance+"/-");
             }
         }
     }

     public void transferTo(Account anotherAccount, double
amount) {
         if (anotherAccount != null) {
             if (amount > 0) {
                 if (balance >= amount) {
                     balance -= amount;
                     anotherAccount.credit(amount);
                     System.out.println("Transferred Amount : "
+ amount +"/-"+ " to account: " +
anotherAccount.getAccountNumber());
                 } else {
```

```
                    System.out.println("Amount excceded. Your
Current balance: " + balance);
                }
            } else {
                System.out.println("Invalid transfer amount.");
            }
        } else {
            System.out.println("Invalid account.");
        }
    }

    public String toString() {
          return "Account [ID = " + accountNumber + " , Name
=" +Name+ ",Balance = " +balance +"]";
     }
}

public class AccountClass {
     public static void main(String[] args) {

        Account account1 = new Account("87654321234","Deepika",
10000.0);
        System.out.println(account1);
        Account account2 = new
Account("98765432123","Priyanka", 500.0);
        Account account3 = new Account("54321234567", "Deepika
Naik", 2000.);

        account1.credit(200);
        account1.debit(150);
        account1.transferTo(account2, 300);

        System.out.println("Balance of account1: " +
account1.getBalance()+"/-");
        System.out.println("Balance of account2: " +
account2.getBalance()+"/-");

     }
}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac AccountClass.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java AccountClass
Account [ID = 87654321234 , Name =null,Balance = 10000.0]
Credited Amount : 200.0/- New balance: 10200.0/-
Debited Amount : 150.0/- New balance: 10050.0/-
Credited Amount : 300.0/- New balance: 800.0/-
Transferred Amount : 300.0/- to account: 98765432123
Balance of account1: 9750.0/-
Balance of account2: 800.0/-
```

Problem-5

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



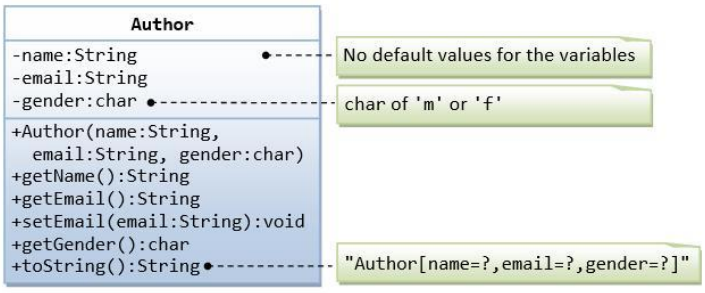
Answer:

```
class Date {
	private int day;
	private int month;
	private int year;

	public Date(int day, int month, int year) {
		this.day = day;
		this.month = month;
		this.year = year;
	}

	public int getDay() {
		return day;
	}

	public void setDay(int day) {
		this.day = day;
	}

	public int getMonth() {
		return month;
	}

	public void setMonth(int month) {
		this.month = month;
	}

	public int getYear() {
		return year;
	}
```

```
	public void setYear(int year) {
		this.year = year;
	}
	public void displayDate() {
		System.out.println(day + "/" + month + "/" + year);
	}
}

public class DateClass {
	public static void main(String[] args) {
		// Create a Date object
		Date date = new Date(12, 7, 2024);

		// Print the date
		System.out.println("Date: ");
		date.displayDate();

		// Change the date
		date.setDay(22);
		date.setMonth(9);
		date.setYear(2025);

		// Print the updated date
		System.out.println("Updated Date: ");
		date.displayDate();
	}
}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac DateClass.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java DateClass
Date:
12/7/2024
Updated Date:
22/9/2025
```

Problem-6

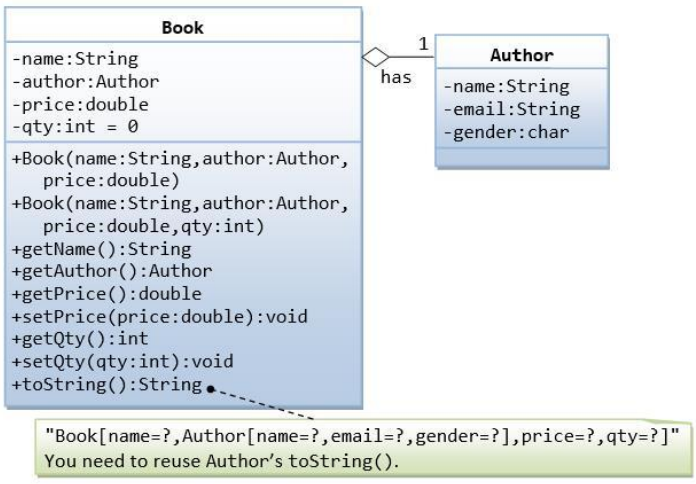


A class called Author (as shown in the class diagram) is designed to model a book's author. It contains:

- 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().

Answer:

```
class Author {
      private String name;
      private String email;
      private char gender;
```

```
	// Constructor to initialize the name, email, and gender
	public Author(String name, String email, char gender) {
		this.name = name;
		this.email = email;
		this.gender = gender;
	}

	// Getter methods
	public String getName() {
		return name;
	}

	public String getEmail() {
		return email;
	}

	public char getGender() {
		return gender;
	}

	// Setter method for email
	public void setEmail(String email) {
		this.email = email;
	}

	// toString method
	@Override
	public String toString() {
		return "Author[name=" + name + ",email=" + email +
",gender=" + gender + "]";
	}
}


class Book {
	private String name;
	private Author author; // An instance of Author
	private double price;
	private int qty;

	// Constructor with three parameters
	public Book(String name, Author author, double price) {
		this.name = name;
		this.author = author;
		this.price = price;
		this.qty = 0; // Default quantity
	}
```

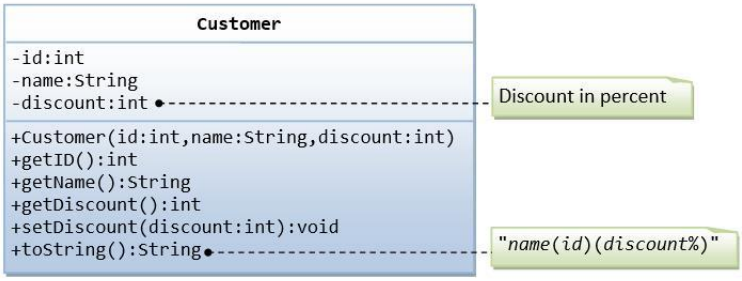
```
	// Constructor with four parameters
	public Book(String name, Author author, double price, int
qty) {
		this.name = name;
		this.author = author;
		this.price = price;
		this.qty = qty;
	}

	// Getter methods
	public String getName() {
		return name;
	}

	public Author getAuthor() {
		return author;
	}

	public double getPrice() {
		return price;
	}

	public int getQty() {
		return qty;
	}

	// Setter methods
	public void setPrice(double price) {
		this.price = price;
	}

	public void setQty(int qty) {
		this.qty = qty;
	}

	// toString method
	@Override
	public String toString() {
		return "Book[name=" + name + "," + author.toString()
+ ",price=" + price + ",qty=" + qty + "]";
	}
}

public class AuthorClass {

	public static void main(String[] args) {


		// Create an Author instance
```

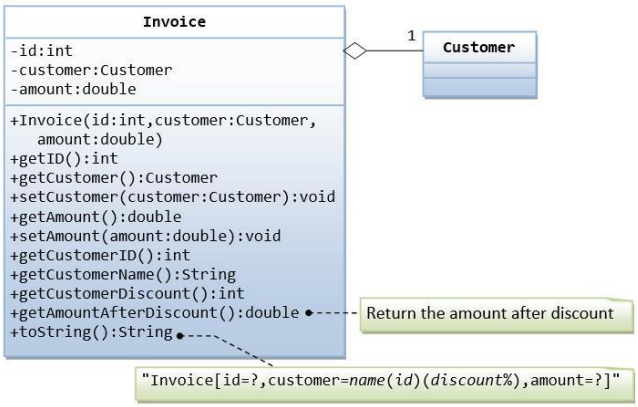
```
            Author author = new Author("JemsGohsling",
"gems@somewhere.com", 'm');

            // Create a Book instance
            Book book1 = new Book("Java for Beginners", author,
29.95);
            Book book2 = new Book("Advanced Java", author, 39.95,
10);

            // Display the details of the book
            System.out.println(book1.toString());
            System.out.println(book2.toString());

            // Change the price and quantity of book1
            book1.setPrice(25.95);
            book1.setQty(5);

            // Display the updated details of book1
            System.out.println(book1.toString());
      }
}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac AuthorClass.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java AuthorClass
Book[name=Java for Beginners,Author[name=JemsGohsling,email=gems@gmail.com,gender=m],price=59.95,qty=0]
Book[name=Advanced Java,Author[name=JemsGohsling,email=gems@gmail.com,gender=m],price=50.95,qty=10]
Book[name=Java for Beginners,Author[name=JemsGohsling,email=gems@gmail.com,gender=m],price=100.95,qty=5]
```

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.





Answer:

```
class Customer {
    private int id;
    private String name;
    private int discount;
    public Customer(int id, String name, int discount) {
        this.id = id;
        this.name = name;
        this.discount = discount;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public int getDiscount() {
        return discount;
```

```
    }
    public void setDiscount(int discount) {
        this.discount = discount;
    }
    @Override
    public String toString() {
        return "Customer[id=" + id + ",name=" + name + ",discount=" + discount + "]";
    }
}
class Invoice1 {
    private int id;
    private Customer customer;
    private double amount;
    public Invoice1(int id, Customer customer, double amount) {
        this.id = id;
        this.customer = customer;
        this.amount = amount;
    }
    public int getId() {
        return id;
    }
    public Customer getCustomer() {
        return customer;
    }
    public double getAmount() {
```

```
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
    public int getCustomerId() {
        return customer.getId();
    }
    public String getCustomerName() {
        return customer.getName();
    }

    public double getAmountAfterDiscount() {
        return amount * (1 - customer.getDiscount() / 100.0);
    }
    @Override
    public String toString() {
        return "Invoice[id=" + id + "," + customer.toString() + ",amount=" + amount +
"]";
    }
}
public class CustomerClass {
        public static void main(String[] args) {
                Customer customer = new Customer(1, "John Doe", 10);
                Invoice1 invoice = new Invoice1(1001, customer, 200.00);
                System.out.println(invoice.toString());
```

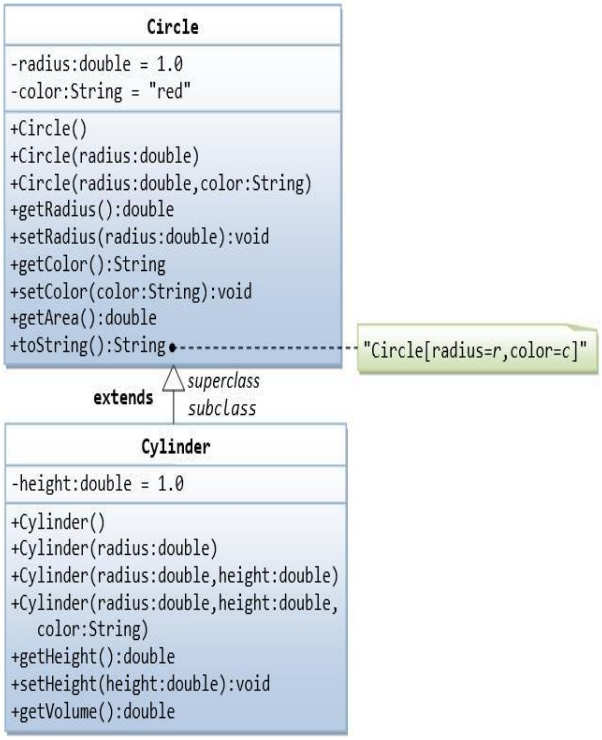
System.out.println("Amount after discount: " + invoice.getAmountAfterDiscount());

}

}

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac CustomerClass.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java CustomerClass
Invoice[id=1001,Customer[id=1,name=John Doe,discount=10],amount=200.0]
Amount after discount: 180.0
```

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`.

Answer:

```
package com.capstone2;
class Circle {
```

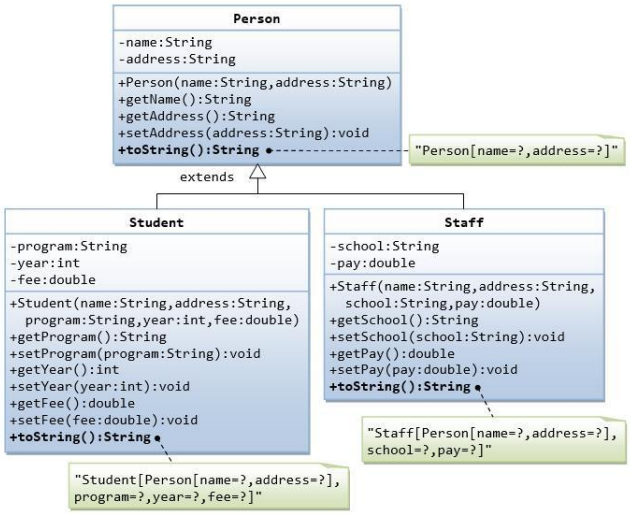
```
	private double radius;
	private String Color;

	public void setRadius(double radius) {
		this.radius = radius;
	}
	public double getRadius() {
		return radius;
	}

	public void setColor(String color) {
		Color = color;
	}
	public String getColor() {
		return Color;
	}

	Circle() {
		this.radius = 1.0;
		this.Color = "red";
	}

	Circle(double radius) {
		this.radius = radius;
	}

	Circle(double radius,String Color) {
		this.radius = radius;
		this.Color=Color;
	}

	public double getArea() {
		return Math.PI * radius * radius;
	}

	public String toString() {
		return "Circle [ radius = " + getRadius() + " ,
color = " + getColor()+" ]";
	}
}

class Cylinder extends Circle {
	private double height;

	public void setHeight(double height) {
		this.height = height;
	}
```

```
	public double getHeight() {
		return height;
	}

	Cylinder() {
		super();
		this.height = 1.0;
	}

	Cylinder(double radius, double height) {
		super(radius);
		this.height = height;
	}
	Cylinder(double radius, double height, String Color) {
		super(radius,Color);
		this.height = height;
	}

	public double getVolume() {
		return getArea() * height;
	}

}

public class CircleClass {

	public static void main(String[] args) {
		Circle circle = new Circle(5);
		Circle circle1 = new Circle();
		System.out.println(circle1);

		System.out.println("Circle Radius: " +
circle.getRadius());
		System.out.println("Circle Area: " +
circle.getArea());

		Cylinder cylinder = new Cylinder(5, 10);
		System.out.println("Cylinder Volume: " +
cylinder.getVolume());
	}

}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac CircleClass.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java CircleClass
Circle [ radius = 1.0 , color = red ]
Circle Radius: 5.0
Circle Area: 78.53981633974483
Cylinder Volume: 785.3981633974483
```

Problem-9

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



Answer:

```
class Person {

        private String Name;

        private String Address;

        public String getName() {

                return Name;

        }

        public void setName(String Name) {

                this.Name = Name;

        }       public void setAddress(String address) {

                Address = address;



        }

        public String getAddress() {
```

```
		return Address;
	}
	Person(String Name, String Address) {
		this.Name = Name;
		this.Address = Address;

	}
	@Override
	public String toString() {
		return "Person [ name= " + Name + " , address= " + getAddress() + " ]";
	}
}
class Student extends Person {
	private String Program;
	private int Year;
	private double Fee;
	public Person person;
	public String getProgram() {
		return Program;
	}
	public void setProgram(String program) {
		Program = program;
	}
	public int getYear() {
		return Year;
```

```
	}
	public void setYear(int year) {
		Year = year;
	}
	public double getFee() {
		return Fee;
	}
	public void setFee(double fee) {
		Fee = fee;
	}
	Student(String Name, String Address, String Program, int Year, double Fee) {
		super(Name,Address);
		person = new Person(Name,Address);
		this.Program = Program;
		this.Year = Year;
		this.Fee = Fee;
	}
	@Override
	public String toString() {
		return "Student [ "+ person + ", Program= " + Program + ", Year= " +
Year + "Fee= " + Fee + " ]";
	}
}
class Staff extends Person {
	private String SchoolName;
	private double Pay;
```

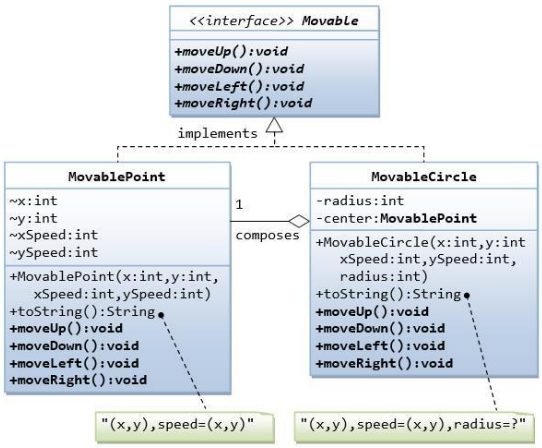
```
	public Person person;
	public String getSchoolName() {
		return SchoolName;
	}
	public void setSchoolName(String schoolName) {
		SchoolName = schoolName;
	}
	public double getPay() {
		return Pay;
	}
	public void setPay(double pay) {
		Pay = pay;
	}
	public Staff(String Name, String Address, String SchoolName, double Pay) {
		super(Name, Address);
		person = new Person(Name,Address);
		this.SchoolName = SchoolName;
		this.Pay = Pay;
	}
	@Override
	public String toString() {
		return "Staff [ " + person + ", School Name= " +SchoolName+ ", Pay=
" +Pay+ " ]";
	}
}
public class PersonClass {
```

```
        public static void main(String[] args) {

                Student student = new Student("Deepika", "Honnavar", "Java", 1 ,
5000);

                Staff staff = new Staff("Rukmini", "Honnavar", "Srinivas Institute Of
Technology", 10000);

                System.out.println(student);

                System.out.println(staff);

        }

}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac PersonClass.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java PersonClass
Student [ Person [ name= Deepika , address= Honnavar ], Program= Java, Year= 1Fee= 5000.0 ]
Staff [ Person [ name= Rukmini , address= Honnavar ], School Name= Srinivas Institute Of Technology, Pay= 10000.0 ]
```

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.



Answer:

```
 interface Movable {
```

```
    void moveUp();
    void moveDown();
    void moveLeft();
    void moveRight();
}
class MovablePoint implements Movable {
    private int x;
    private int y;
    private int xSpeed;
    private int ySpeed;
    public MovablePoint(int x, int y, int xSpeed, int ySpeed) {
        this.x = x;
        this.y = y;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
    }
    @Override
    public void moveUp() {
        y += ySpeed;
    }

    @Override
    public void moveDown() {
        y -= ySpeed;
    }
```

```
    @Override
    public void moveLeft() {
        x -= xSpeed;
    }
    @Override
    public void moveRight() {
        x += xSpeed;
    }
    @Override
    public String toString() {
        return "MovablePoint Speed = (" + x + ", " + y + ")";
    }
}

class MovableCircle implements Movable {
    private MovablePoint center;
    private int radius;
    public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {
        this.center = new MovablePoint(x, y, xSpeed, ySpeed);
        this.radius = radius;
    }
    @Override
    public void moveUp() {
        center.moveUp();
    }
```

```
    @Override
    public void moveDown() {
        center.moveDown();
    }
    @Override
    public void moveLeft() {
        center.moveLeft();
    }

    @Override
    public void moveRight() {
        center.moveRight();
    }
    @Override
    public String toString() {
        return "MovableCircle with center at " + center + " and radius " + radius;
    }
}
public class MethodOverride {
            public static void main(String[] args) {
        MovablePoint point = new MovablePoint(0, 0, 1, 1);
        point.moveUp();
        System.out.println(point);
        MovableCircle circle = new MovableCircle(0, 0, 1, 1, 5);
        circle.moveRight();
```

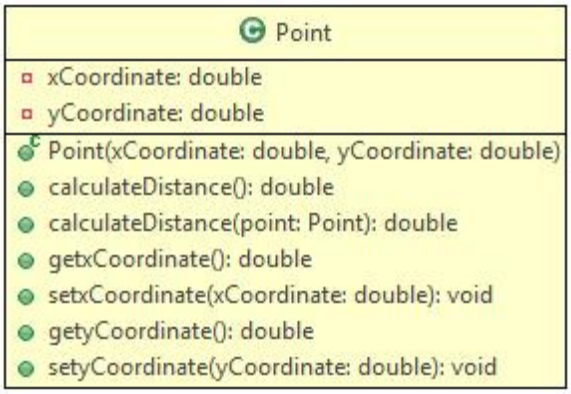
```
        System.out.println(circle);


    }

}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac MethodOverride.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java MethodOverride
MovablePoint Speed = (0, 1)
MovableCircle with center at MovablePoint Speed = (1, 0) and radius 5
```

Problem-11

The Point class is used for representing a point with two coordinates.

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



Method Description

Point(double xCoordinate , double yCoordinate)

Initialize the instance variables xCoordinate and yCoordinate appropriately with the values passed to the constructor.

calculateDistance()

Calculate and return the distance of the point from the origin (0,0). The distance can be calculated using the formula given below. The distance should be rounded off to 2 decimal digits.

distance= √ ((x2-x1)2+(y2-y1)2), where x1 and x2 are values of x-coordinates of two points and y1 and y2 are values of y-coordinates of two points

calculateDistance(Point point)

Calculate and return the distance of the point from the 'point' passed to the method. The distance should be rounded off to 2 decimal digits.

Hints

Use Math.sqrt(double d) method to calculate the square root

Use Math.round(double d) method to round off the values

Implement the getter and setter methods appropriately.

Test the functionalities using the provided Tester class.

Sample Input and Output

Input

Point object - point1

Instance variables	Values
xCoordinate	3.5
yCoordinate	1.5

Point object - point2

Instance variables	Values
xCoordinate	6
yCoordinate	4

Output

```
Distance of point1 from origin is 3.81

Distance of point2 from origin is 7.21

Distance of point1 from point2 is 3.54
```

Answer:

```java
package com.capstone2;
public class Point {
    private double xCoordinate;
    private double yCoordinate;

    public Point(double xCoordinate, double yCoordinate) {
        this.xCoordinate = xCoordinate;
        this.yCoordinate = yCoordinate;
    }

    public double calculateDistance() {
        double distance = Math.sqrt(xCoordinate * xCoordinate
+ yCoordinate * yCoordinate);
        return Math.round(distance * 100.0) / 100.0;
    }

    public double calculateDistance(Point point) {
        double distance = Math.sqrt(Math.pow(point.xCoordinate
- this.xCoordinate, 2) + Math.pow(point.yCoordinate -
this.yCoordinate, 2));
        return Math.round(distance * 100.0) / 100.0;
    }

    public double getxCoordinate() {
        return xCoordinate;
    }

    public void setxCoordinate(double xCoordinate) {
        this.xCoordinate = xCoordinate;
    }

    public double getyCoordinate() {
        return yCoordinate;
    }

    public void setyCoordinate(double yCoordinate) {
        this.yCoordinate = yCoordinate;
    }

    public static void main(String[] args) {
        java.util.Scanner scanner = new
java.util.Scanner(System.in);

        // Input for point1
        System.out.println("Enter xCoordinate for point1:");
        double x1 = scanner.nextDouble();
        System.out.println("Enter yCoordinate for point1:");
        double y1 = scanner.nextDouble();
        Point point1 = new Point(x1, y1);
```

```
        // Input for point2
        System.out.println("Enter xCoordinate for point2:");
        double x2 = scanner.nextDouble();
        System.out.println("Enter yCoordinate for point2:");
        double y2 = scanner.nextDouble();
        Point point2 = new Point(x2, y2);

        scanner.close();

        // Output
        System.out.println("Distance of point1 from origin is
" + point1.calculateDistance());
        System.out.println("Distance of point2 from origin is
" + point2.calculateDistance());
        System.out.println("Distance of point1 from point2 is
" + point1.calculateDistance(point2));
    }
}
```

```
C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>javac Point.java

C:\Users\Administrator\Documents\Capstone_Project_Day2_Assign2>java Point
Enter xCoordinate for point1:
3.5
Enter yCoordinate for point1:
1.5
Enter xCoordinate for point2:
6
Enter yCoordinate for point2:
4
Distance of point1 from origin is 3.81
Distance of point2 from origin is 7.21
Distance of point1 from point2 is 3.54
```