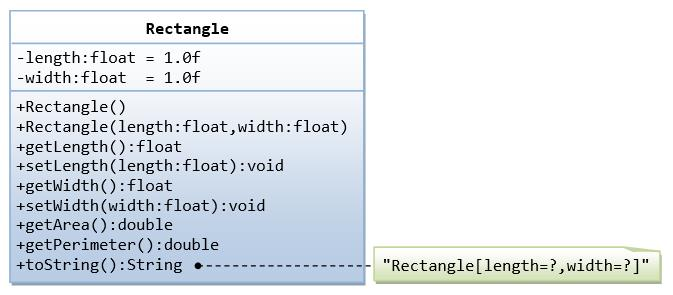
**Capstone project Day2 Asiignment2**

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.



Solution:-

public class Rectangle {

private float length;

private float width;

public Rectangle(float length, float width) {

this.length = length;

this.width = width;

}

public float getLength() {

return length;

}

public void setLength(float length) {

this.length = length;

}

public float getWidth() {

return width;

}

public void setWidth(float width) {

this.width = width;

}

public float calculateArea() {

return length \* width;

}

public float calculatePerimeter() {

return 2 \* (length + width);

}

}

 //main class

public class Main {

public static void main(String[] args) {

// Create a Rectangle object

Rectangle rectangle = new Rectangle(5.0f, 3.0f);

// Print the rectangle's length and width

System.out.println("Length: " + rectangle.getLength());

System.out.println("Width: " + rectangle.getWidth());

// Calculate and print the rectangle's area and perimeter

System.out.println("Area: " + rectangle.calculateArea());

System.out.println("Perimeter: " + rectangle.calculatePerimeter());

// Change the rectangle's length and width

rectangle.setLength(8.0f);

rectangle.setWidth(4.0f);

// Print the updated rectangle's length and width

System.out.println("Updated Length: " + rectangle.getLength());

System.out.println("Updated Width: " + rectangle.getWidth());

// Calculate and print the updated rectangle's area and perimeter

System.out.println("Updated Area: " + rectangle.calculateArea());

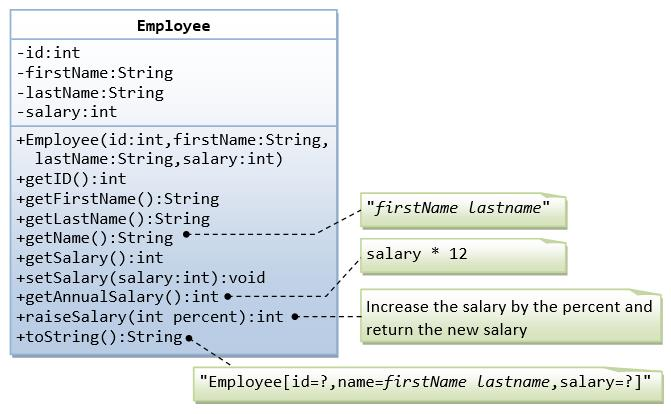
System.out.println("Updated Perimeter: " + rectangle.calculatePerimeter());

}

}

**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.



**Solution:-**

package employee;

public class Employee {

private int id;

private String name;

private double salary;

public Employee(int id, String name, double salary) {

super();

this.id = id;

this.name = name;

this.salary = salary;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public void raiseSalary(double percent) {

salary \*= (1 + percent / 100);

}

}

//main class

package employee;

public class Main {

public static void main(String[] args) {

// Create an Employee object

Employee employee = new Employee(1, "punith", 50000.0);

// Print the employee's details

System.***out***.println("ID: " + employee.getId());

System.***out***.println("Name: " + employee.getName());

System.***out***.println("Salary: " + employee.getSalary());

// Raise the employee's salary by 10%

employee.raiseSalary(10);

// Print the updated employee's details

System.***out***.println("Updated ID: " + employee.getId());

System.***out***.println("Updated Name: " + employee.getName());

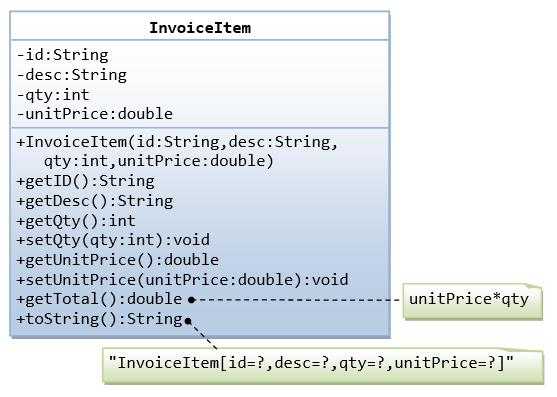
System.***out***.println("Updated Salary: " + employee.getSalary());

}

}

**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.



Solution:-

package invoice;

public class InvoiceItem {

private int id;

private String description;

private int quantity;

private double unitPrice;

public InvoiceItem(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;

}

}

//main class

package invoice;

public class Main {

public static void main(String[] args) {

// Create an array of InvoiceItem objects

InvoiceItem[] items = new InvoiceItem[3];

items[0] = new InvoiceItem(1, "Widget", 2, 15.99);

items[1] = new InvoiceItem(2, "Gadget", 3, 24.99);

items[2] = new InvoiceItem(3, "Thingamajig", 1, 9.99);

// Print the details of each InvoiceItem

for (InvoiceItem 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();

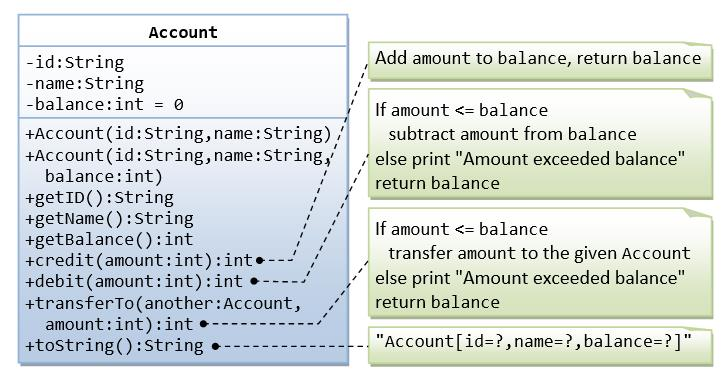
}

}

}

**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.



Solution:-

public class Account {

private int accountNumber;

private double balance;

public Account(int accountNumber, double balance) {

this.accountNumber = accountNumber;

this.balance = balance;

}

public int getAccountNumber() {

return accountNumber;

}

public void setAccountNumber(int accountNumber) {

this.accountNumber = accountNumber;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public void credit(double amount) {

balance += amount;

}

public void debit(double amount) {

if (balance >= amount) {

balance -= amount;

} else {

System.out.println("Insufficient funds");

}

}

public void transferTo(Account anotherAccount, double amount) {

if (balance >= amount) {

debit(amount);

anotherAccount.credit(amount);

} else {

System.out.println("Insufficient funds");

}

}

}

//main class

package account;

public class Main {

public static void main(String[] args) {

// Create two Account objects

Account account1 = new Account(12345, 1000.0);

Account account2 = new Account(67890, 500.0);

// Print the initial balances

System.***out***.println("Initial balance of account 1: " + account1.getBalance());

System.***out***.println("Initial balance of account 2: " + account2.getBalance());

// Credit account 1

account1.credit(500.0);

System.***out***.println("After crediting account 1: " + account1.getBalance());

// Debit account 1

account1.debit(200.0);

System.***out***.println("After debiting account 1: " + account1.getBalance());

// Transfer funds from account 1 to account 2

account1.transferTo(account2, 300.0);

System.***out***.println("After transferring funds: ");

System.***out***.println("Account 1 balance: " + account1.getBalance());

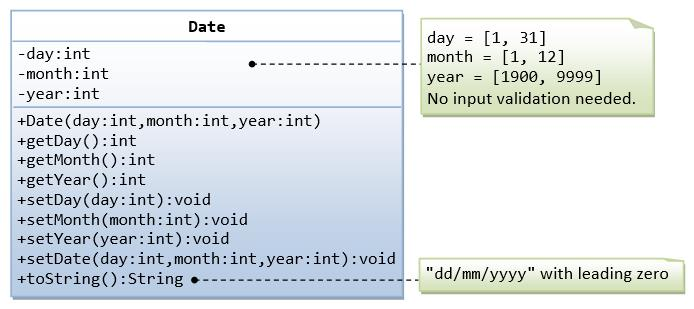
System.***out***.println("Account 2 balance: " + account2.getBalance());

}

}

**Problem-5**

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



Solution:-

public 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);

}

}

 //main class

public class Main {

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

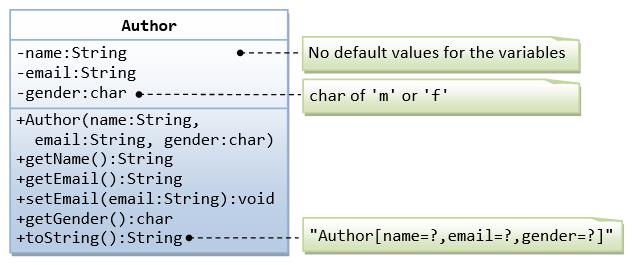
System.out.println("Updated Date: ");

date.displayDate();

}

}

**Problem-6**



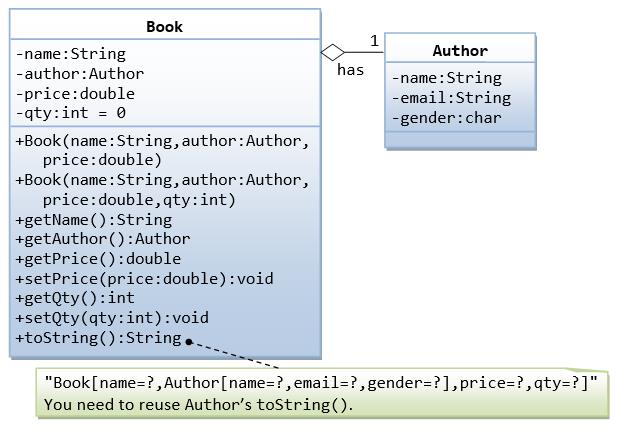
A class called Author (as shown in the class diagram) is designed to model a book's author. It contains:

* 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().

 Solution:-

package book;

public class Author {

// Private instance variables

private String name;

private String email;

private char gender; // 'm' or 'f'

// Constructor to initialize name, email, and gender

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

this.name = name;

this.email = email;

this.gender = gender;

}

// Getters for name, email, and gender (no setters for name and gender)

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public char getGender() {

return gender;

}

// toString method to return string representation of Author

public String toString() {

return "Author[name=" + name + ",email=" + email + ",gender=" + gender + "]";

}

}

// book class

package book;

public class Book {

// Private instance variables

private String name;

private Author author;

private double price;

private int qty;

// Constructor with three parameters (no qty)

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

this.name = name;

this.author = author;

this.price = price;

}

// Constructor with all four parameters

public Book(String name, Author author, double price, int qty) {

this.name = name;

this.author = author;

this.price = price;

this.qty = qty;

}

// Getters and setters for name, author, price, and qty

public String getName() {

return name;

}

public Author getAuthor() {

return author;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getQty() {

return qty;

}

public void setQty(int qty) {

this.qty = qty;

}

// toString method to return string representation of Book

public String toString() {

return "Book[name=" + name + "," + author.toString() + ",price=" + price + ",qty=" + qty + "]";

}

}

//main class

package book;

public class Main {

public static void main(String[] args) {

// Create an Author

Author author = new Author("Tan Ah Teck", "ahTeck@somewhere.com", 'm');

// Create a Book using the first constructor

Book book1 = new Book("Java Programming", author, 29.99);

// Create a Book using the second constructor

Book book2 = new Book("Python Basics", author, 19.99, 50);

// Print details of the books using toString()

System.***out***.println(book1.toString());

System.***out***.println(book2.toString());

// Example of modifying price and quantity

book1.setPrice(39.99);

book2.setQty(60);

// Print modified details of the books

System.***out***.println("Modified details:");

System.***out***.println(book1.toString());

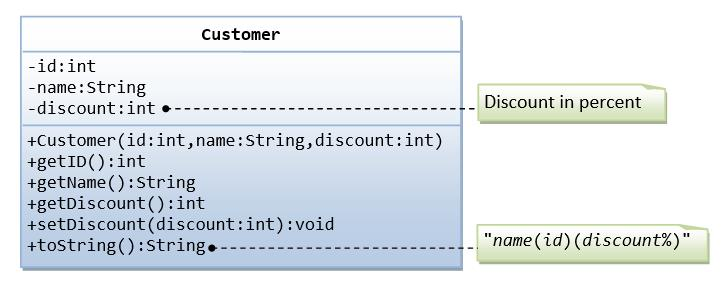
System.***out***.println(book2.toString());

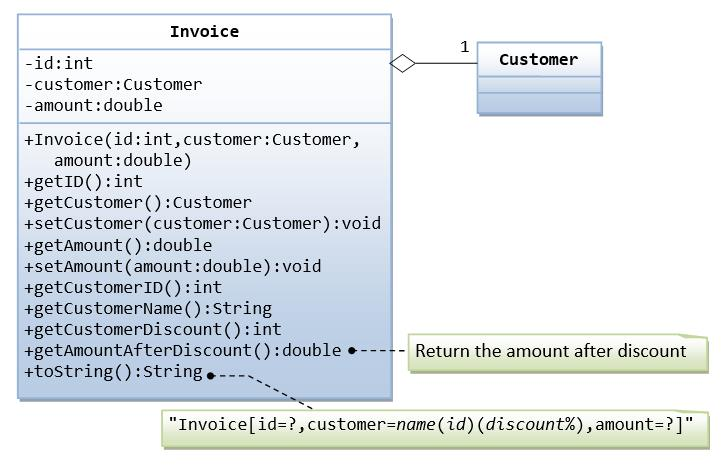
}

}

**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.





Soluion:-

package customerinvoice;

public class Customer {

// Private instance variables

private int id;

private String name;

private int discount; // discount percentage

// Constructor to initialize id, name, and discount

public Customer(int id, String name, int discount) {

this.id = id;

this.name = name;

this.discount = discount;

}

// Getters for id, name, and discount (no setters for id)

public int getId() {

return id;

}

public String getName() {

return name;

}

public int getDiscount() {

return discount;

}

// toString method to return string representation of Customer

public String toString() {

return "Customer[id=" + id + ",name=" + name + ",discount=" + discount + "]";

}

}

//Invoice class

package customerinvoice;

public class Invoice {

// Private instance variables

private int id;

private Customer customer; // composition: Invoice has a Customer

private double amount;

// Constructor to initialize id, customer, and amount

public Invoice(int id, Customer customer, double amount) {

this.id = id;

this.customer = customer;

this.amount = amount;

}

// Getters for id, customer, and amount (no setters for id)

public int getId() {

return id;

}

public Customer getCustomer() {

return customer;

}

public double getAmount() {

return amount;

}

// Method to get the customer's name

public String getCustomerName() {

return customer.getName();

}

// Method to get the amount after applying customer's discount

public double getAmountAfterDiscount() {

return amount - (amount \* customer.getDiscount() / 100);

}

// toString method to return string representation of Invoice

public String toString() {

return "Invoice[id=" + id + "," + customer.toString() + ",amount=" + amount + "]";

}

}

//Main class

package customerinvoice;

public class Main {

public static void main(String[] args) {

// Create a Customer

Customer customer = new Customer(1, "punith", 10);

// Create an Invoice for the customer

Invoice invoice = new Invoice(1001, customer, 500.0);

// Print details of the invoice

System.***out***.println(invoice.toString());

// Example of using getters

System.***out***.println("Customer Name: " + invoice.getCustomerName());

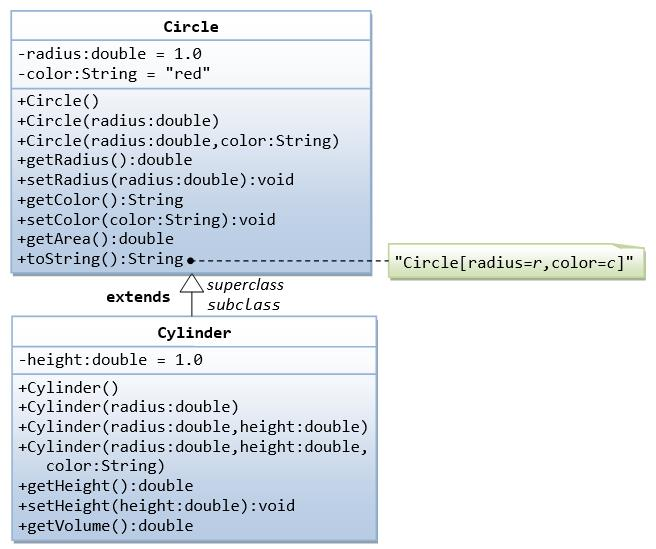
System.***out***.println("Invoice Amount: $" + invoice.getAmount());

System.***out***.println("Amount after Discount: $" + invoice.getAmountAfterDiscount());

}

}

**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.

 Solution:-

package circle;

public class Circle {

private double radius;

public Circle() {

this.radius = 1.0;

}

public Circle(double radius) {

this.radius = radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

public double getArea() {

return Math.***PI*** \* radius \* radius;

}

public String toString() {

return "Circle[radius=" + radius + "]";

}

}

//Cylinder class

package circle;

public class Cylinder extends Circle {

private double height;

public Cylinder() {

super();

this.height = 1.0;

}

public Cylinder(double height) {

super();

this.height = height;

}

public Cylinder(double radius, double height) {

super(radius);

this.height = height;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

}

public double getVolume() {

return getArea() \* height;

}

*@Override*

public String toString() {

return "Cylinder[" + super.toString() + ",height=" + height + "]";

}

}

 //main class

package circle;

public class Main {

public static void main(String[] args) {

// Create a Circle

Circle circle = new Circle(2.0);

System.***out***.println("Circle details:");

System.***out***.println(circle.toString());

System.***out***.println("Area: " + circle.getArea());

// Create a Cylinder

Cylinder cylinder = new Cylinder(3.0, 4.0);

System.***out***.println("\nCylinder details:");

System.***out***.println(cylinder.toString());

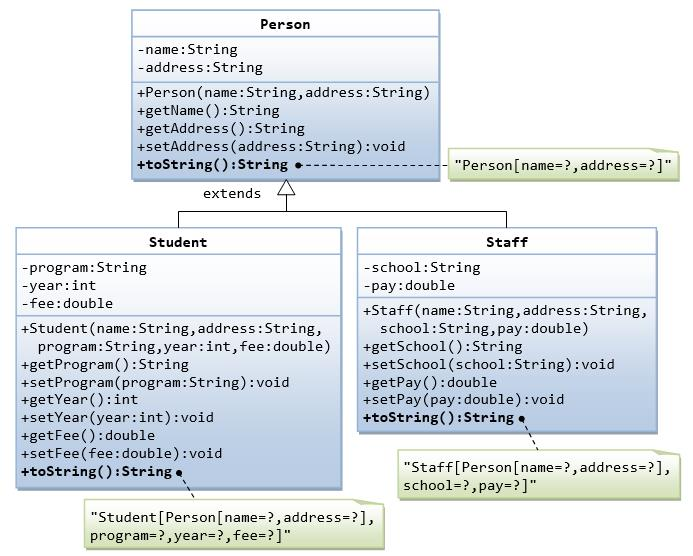
System.***out***.println("Volume: " + cylinder.getVolume());

}

}

**Problem-9**

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



Solution:-

package person;

public class Person {

// Instance variables

private String name;

private String address;

// Constructor

public Person(String name, String address) {

this.name = name;

this.address = address;

}

// Getter and setter methods

public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

// toString method

*@Override*

public String toString() {

return "Person [name=" + name + ", address=" + address + "]";

}

}

//Student class

package person;

public class Student extends Person {

// Instance variables specific to Student

private String program;

private int year;

private double fee;

// Constructor

public Student(String name, String address, String program, int year, double fee) {

super(name, address);

this.program = program;

this.year = year;

this.fee = fee;

}

// Getter and setter methods

public String getProgram() {

return program;

}

public void setProgram(String program) {

this.program = program;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

public double getFee() {

return fee;

}

public void setFee(double fee) {

this.fee = fee;

}

// toString method overriding Person's toString

*@Override*

public String toString() {

return "Student " + super.toString() + ", program=" + program + ", year=" + year + ", fee=" + fee + "]";

}

}

//Staff class

package person;

public class Staff extends Person {

// Instance variables specific to Staff

private String school;

private double pay;

// Constructor

public Staff(String name, String address, String school, double pay) {

super(name, address);

this.school = school;

this.pay = pay;

}

// Getter and setter methods

public String getSchool() {

return school;

}

public void setSchool(String school) {

this.school = school;

}

public double getPay() {

return pay;

}

public void setPay(double pay) {

this.pay = pay;

}

// toString method overriding Person's toString

*@Override*

public String toString() {

return "Staff " + super.toString() + ", school=" + school + ", pay=" + pay + "]";

}

}

 //main class

package person;

public class Main {

public static void main(String[] args) {

// Create a Student

Student student = new Student("punith", "hassan", "Computer Science", 2023, 12000.0);

System.***out***.println(student);

// Create a Staff

Staff staff = new Staff("punith nr", "sakaleshpura", "malnad School", 45000.0);

System.***out***.println(staff);

// Test modifying student and staff details

student.setFee(12500.0);

staff.setPay(47000.0);

// Print updated details

System.***out***.println("\nUpdated Details:");

System.***out***.println(student);

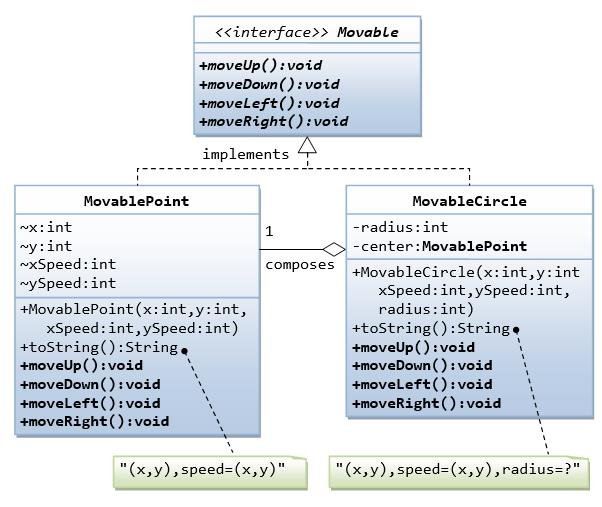
System.***out***.println(staff);

}

}

**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.



**Solution:-**

package moveable;

public interface Moveable {

void moveUp();

void moveDown();

void moveLeft();

void moveRight();

}

//movablepoint class

package moveable;

public class MovablePoint implements Moveable {

// Instance variables

private int x;

private int y;

// Constructor

public MovablePoint(int x, int y) {

this.x = x;

this.y = y;

}

// Implementing interface methods

*@Override*

public void moveUp() {

y++;

}

*@Override*

public void moveDown() {

y--;

}

*@Override*

public void moveLeft() {

x--;

}

*@Override*

public void moveRight() {

x++;

}

// toString method to print current coordinates

*@Override*

public String toString() {

return "MovablePoint[x=" + x + ",y=" + y + "]";

}

}

//movable circle class

package moveable;

public class MovableCircle implements Moveable {

// Instance variables

private MovablePoint center;

private int radius;

// Constructor

public MovableCircle(int x, int y, int radius) {

this.center = new MovablePoint(x, y);

this.radius = radius;

}

// Implementing interface methods

*@Override*

public void moveUp() {

center.moveUp();

}

*@Override*

public void moveDown() {

center.moveDown();

}

*@Override*

public void moveLeft() {

center.moveLeft();

}

*@Override*

public void moveRight() {

center.moveRight();

}

// toString method to print current position and radius

*@Override*

public String toString() {

return "MovableCircle[center=" + center.toString() + ",radius=" + radius + "]";

}

}

//main class

package moveable;

public class Main {

public static void main(String[] args) {

// Create a MovablePoint

MovablePoint point = new MovablePoint(1, 2);

System.***out***.println("Original Point: " + point);

// Move the point

point.moveUp();

point.moveLeft();

System.***out***.println("Point after moving up and left: " + point);

// Create a MovableCircle

MovableCircle circle = new MovableCircle(3, 4, 5);

System.***out***.println("Original Circle: " + circle);

// Move the circle

circle.moveDown();

circle.moveRight();

System.***out***.println("Circle after moving down and right: " + circle);

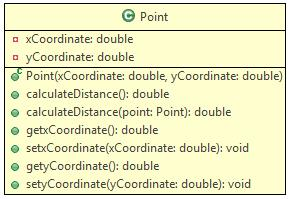
}

}

**Problem statement: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

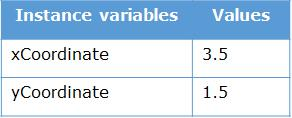
Implement the getter and setter methods appropriately.

Test the functionalities using the provided Tester class.

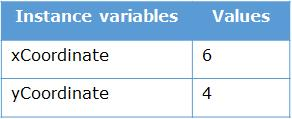
**Sample Input and Output**

**Input**

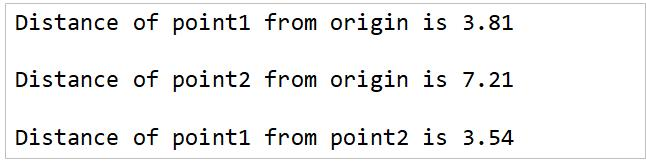
Point object - point1



Point object - point2



**Output**



Solution:-

package point;

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.getXCoordinate() - this.xCoordinate, 2) +

Math.*pow*(point.getYCoordinate() - 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;

}

}

//main class

package point;

public class Main {

public static void main(String[] args) {

// Create two Point objects

Point point1 = new Point(3.5, 1.5);

Point point2 = new Point(6, 4);

// Calculate and print distance of point1 from origin

System.***out***.println("Distance of point1 from origin: " + point1.calculateDistance());

// Calculate and print distance of point2 from origin

System.***out***.println("Distance of point2 from origin: " + point2.calculateDistance());

// Calculate and print distance of point1 from point2

System.***out***.println("Distance between point1 and point2: " + point1.calculateDistance(point2));

}

}