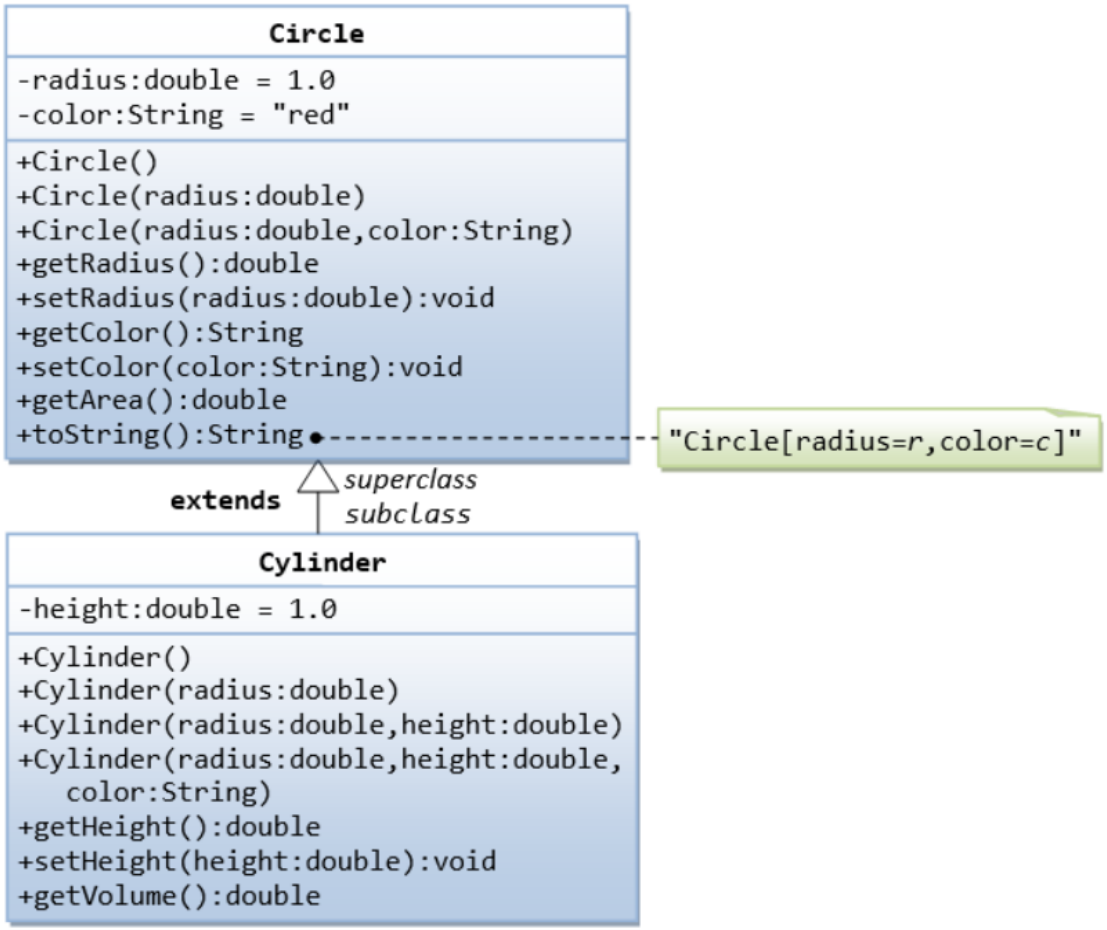
Hands-on No. : 3

Topic : OOPs- Inhertiance,Polymorphism,Abstraction

Date : 31.07.2025

Solve the following problems

Question No.	Question Detail
1	You are required to design and implement two classes, **Circle** and **Cylinder**, that model the behavior of a circle and a cylinder respectively. The **Cylinder** class should be a subclass of **Circle** and inherit its properties and methods. This task is aimed at demonstrating the concepts of **inheritance**, **constructor overloading**, **method overriding**, and **code reuse** in object-oriented programming.



2	You are tasked with demonstrating **method overloading** in the Employee class, which models an employee in an organization with attributes such as ID, name, and salary. You must implement multiple versions of the method calculateYearlySalary using **method overloading**, each accepting a different set of parameters to calculate the employee's annual salary. **Method Requirements:** 1. **calculateYearlySalary(double monthlySalary)** ○ Calculates and returns the yearly salary based on a given **monthly salary**. 2. **calculateYearlySalary(double dailySalary, int workingDays)** ○ Calculates and returns the yearly salary using **daily salary** and **number of working days in a year**. 3. **calculateYearlySalary(int workingDays, double hourlySalary, int hoursPerDay)** ○ Calculates and returns the yearly salary using **hourly wage**, **working days**, and **hours worked per day**. 4. **toString() method** ○ Returns a string containing the **employee's ID, name, and salary**.
3	You have been hired to build a basic flight booking system, similar to **MakeMyTrip**, where users can book flights from different airlines such as **Indigo**, **Air India**, and **SpiceJet**. Even though each airline might have its own way of handling bookings and cancellations, all airlines must follow a **common structure** for the system to work uniformly. This can be achieved using **interfaces** and **polymorphism** in Java. Requirements: **Create an interface** named FlightBooking with the following methods: • void bookTicket() – to book a ticket. • void cancelTicket() – to cancel a booked ticket. **Create three classes** that implement the FlightBooking interface: • Indigo – provides its own way to book and cancel tickets. • AirIndia – provides its own way to book and cancel tickets. • SpiceJet – provides its own way to book and cancel tickets. **In the main() method** of your program: • Create an object of each class: Indigo, AirIndia, and SpiceJet. • Use each object to:

	○ Call bookTicket() to simulate booking a flight. ○ Call cancelTicket() to simulate cancelling a flight.
4	You are developing an application for an online shopping platform. You are required to create a class representing a "Product" and its subclasses for different types of products. **Product Class:** The **Product** class represents an abstract class with the following attributes and methods: **Attributes:** **productID** (int): A unique identifier for each product. **productName** (String): The name of the product. **price** (double): The price of the product. **Methods:** **Constructor:** Initializes the **productID**, **productName**, and **price** attributes. **getProductId ()**: Returns the product's unique identifier. **getProductName ()**: Returns the name of the product. **getPrice ()**: Returns the price of the product. **displayInfo ()**: Method is declared as abstract. **Subclasses of Product:** **1. ElectronicProduct:** **Attributes: warrantyPeriod** (int): The warranty period of the electronic product in months. **Methods:** **Constructor ():** Initializes the attributes of the **ElectronicProduct** class, Including calling the superclass constructor. **getWarrantyPeriod ()**: Returns the warranty period of the electronic product. **displayInfo ()**: Overrides the **displayInfo ()** method of the superclass to display information about the product along with the warranty. **2. ClothingProduct:** **Attributes:size** (String): The size of the clothing product (e.g., S, M, L, XL). **material** (String): The material of the clothing product (e.g., Cotton, Silk, Polyester). **Methods:** **Constructor:** Initializes the attributes of the **ClothingProduct** class, including calling the superclass constructor. **getSize ():** Returns the size of the clothing product. **getMaterial ():** Returns the material of the clothing product. **displayInfo ():** Overrides the **displayInfo ()** method of the superclass to display additional information about the size and material. **Application Flow:** 1. Create a few instances of the **Product** class and its subclasses (**ElectronicProduct** and **ClothingProduct**) with different attributes. 2. Display the information of each product using the **displayInfo ()** method.