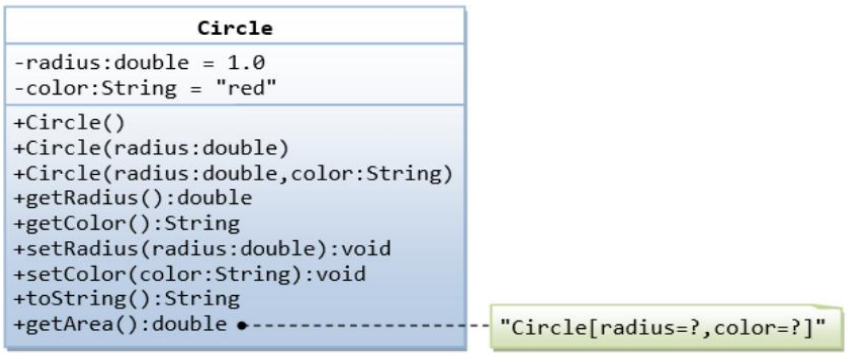
Hands-on No. : 6

Topic : OOPs- Class, Object, Constructor, Encapsulation

Date : 14-05-2024

Solve the following problems

Q. No.	Question Detail	Level
1	Design the Circle class as given below and test the methods of the class by creating objects of type Circle in the driver class. 	Easy
2	Write a program to print the area of a rectangle by creating a class named 'Area' having two methods. First method named as 'setDim' takes length and breadth of rectangle as parameters and the second method named as 'getArea' returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.	Easy
3	Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.	Easy
4	Write a program by creating an 'Employee' class having the following methods and print the final salary. • 'getInfo()' which takes the salary, number of hours of work per day of employee as parameter • 'AddSal()' which adds $10 to salary of the employee if it is less than $500.	Easy

	• 'AddWork()' which adds $5 to salary of employee if the number of hours of work per day is more than 6 hours.	
5	Suppose you are designing a simple class to keep track of the number of students enrolled in a school. Each time a new student is enrolled, the total count of students should increase. Additionally, you want to define a constant variable for the maximum capacity of students that the school can accommodate. Define a Java class named **School** with the following requirements: 1. Implement a static variable named **totalStudents** to keep track of the total number of students enrolled in the school. 2. Define a constant variable named **MAX_CAPACITY** to represent the maximum capacity of students that the school can accommodate. Set its value to 500. 3. Implement a method named **enrollStudent()** that increments the **totalStudents** count each time a new student is enrolled. 4. Implement a method named **getTotalStudents()** that returns the current count of total students. 5. Ensure that the **MAX_CAPACITY** variable cannot be modified after initialization. Write the Java class **School** with the above requirements and demonstrate its usage in the **Main** class by enrolling students and retrieving the total count.	Easy
6	You are developing a simple program to convert temperature measurements between Celsius and Fahrenheit scales. To achieve this, you want to define a Java class with static methods for temperature conversion. Define a Java class named **TemperatureConverter** with the following specifications: 1. Implement a static method named **celsiusToFahrenheit()** that takes a temperature value in Celsius as input and returns the equivalent temperature in Fahrenheit using the formula: Fahrenheit=95×Celsius+32Fahrenheit=59×Celsius+32. 2. Implement another static method named **fahrenheitToCelsius()** that takes a temperature value in Fahrenheit as input and returns the equivalent temperature in	Easy

	Celsius using the formula: Celsius=59×(Fahrenheit−32)Celsius=95×(Fahrenheit−32). Write the Java class **TemperatureConverter** with the given requirements and demonstrate its usage in the **Main** class by converting temperatures between Celsius and Fahrenheit scales.	
7	Write the java implementation for a class named Theatre to display the details of theater. The class Theatre is described as follows: **Attributes:** theatreId : int ,theatreName : String, totalTheatreScreens, theatreLocation **Methods:** (i) A Theatre instance can be created by supplying the value for all the attributes of class. (ii) A method to display the details of a theatre Write class TestTheatre.java which Create n instance of Theatre and display the n instance theatre details.	Medium
8	You are tasked with creating a Java class to represent a student in a school management system. To allow easy access to the student's details, you need to implement public instance variables in the class. Define a Java class named **Student** with the following specifications: 1. Public instance variables to store the student's ID, name, age, and grade. 2. Implement a default constructor that initializes the student's details as follows: • Student ID: 0 • Student name: "Unknown" • Age: 0 • Grade: "Unknown" 3. Implement another constructor that takes parameters for student ID, name, age, and grade, and initializes the corresponding instance variables with the provided values.	Medium

9	Define a class in JAVA with following description: **Private Members** A data member Flight number of type integer A data member Destination of type string A data member Distance of type float A data member Fuel of type float A member function CALFUEL() to calculate the value of Fuel as per the following criteria ***Distance*** ***Fuel*** <=1000 500 more than 1000 and <=2000 1100 more than 2000 2200 **Public Members** A function FEEDINFO() to allow user to enter values for Flight Number, Destination, Distance & call function CALFUEL() to calculate the quantity of Fuel A function SHOWINFO() to allow user to view the content of all the data members	Medium
10	You are designing a simple banking system where you need to create a class to represent a bank account. To ensure data security and prevent unauthorized access to sensitive information, you want to encapsulate the account details by providing access only through getter and setter methods. Define a Java class named **BankAccount** with the following specifications: 1. Private instance variables to store the account number, account holder's name, and account balance. 2. Public getter and setter methods for each instance variable to provide controlled access to the account details. Ensure that the setter methods validate the input before assigning it to the instance variables: • The account number must be a positive integer. • The account holder's name cannot be empty.	Medium

	• The account balance must be a non-negative value. 3. Implement a method named **deposit()** that takes an amount as input and adds it to the account balance. 4. Implement a method named **withdraw()** that takes an amount as input and subtracts it from the account balance, ensuring that the account balance never becomes negative. Write the Java class **BankAccount** with the given requirements and demonstrate its usage in the **Main** class by creating an account, performing deposits and withdrawals, and retrieving account details.	