# CONSTRUCTOR

**1. Write a Java program to create a class called "Cat" with instance variables name and age. Implement a default constructor that initializes the name to "Unknown" and the age to 0. Print the values of the variables.**

**2. Write a Java program to create a class called Dog with instance variables name and color. Implement a parameterized constructor that takes name and color as parameters and initializes the instance variables. Print the values of the variables.**

**3. Write a Java program to create a class called "Book" with instance variables title, author, and price. Implement a default constructor and two parameterized constructors:**

- One constructor takes title and author as parameters.
- The other constructor takes title, author, and price as parameters.
- Print the values of the variables for each constructor.

**4. Write a Java program to create a class called Student with instance variables studentId, studentName, and grade. Implement a default constructor and a parameterized constructor that takes all three instance variables. Use constructor chaining to initialize the variables. Print the values of the variables.**



**5. Write a Java program to create a class called Rectangle with instance variables length and width. Implement a parameterized constructor and a copy constructor that initializes a new object using the values of an existing object. Print the values of the variables.**

**6. Write a Java program to create a class called Account with instance variables accountNumber and balance. Implement a parameterized constructor that initializes these variables with validation:**

- accountNumber should be non-null and non-empty.
- balance should be non-negative.
- Print an error message if the validation fails.

**7. Write a Java program to create a class called Singleton that ensures only one instance of the class can be created. Implement a private constructor and a public static method to get the single instance of the class. Print a message indicating the creation of the instance.**

**8. Write a Java program to create a class called Classroom with instance variables className and students (an array of strings). Implement a parameterized constructor that initializes these variables. Print the values of the variables.**

**9. Write a Java program to create a class called Point with instance variables x and y. Implement overloaded constructors:**

- One constructor takes int parameters.
- Another constructor takes double parameters.
- Print the values of the variables for each constructor.

# ENCAPSULATION

**1. Write a Java program to create a class called Person with private instance variables name, age. and country. Provide public getter and setter methods to access and modify these variables.**

**2. Write a Java program to create a class called BankAccount with private instance variables accountNumber and balance. Provide public getter and setter methods to access and modify these variables.**

**3. Write a Java program to create a class called Rectangle with private instance variables length and width. Provide public getter and setter methods to access and modify these variables.**

**4. Write a Java program to create a class called Employee with private instance variables employee_id, employee_name, and employee_salary. Provide public getter and setter methods to access and modify the id and name variables, but provide a getter method for the salary variable that returns a formatted string.**

**5. Write a Java program to create a class called Circle with a private instance variable radius. Provide public getter and setter methods to access and modify the radius variable. However, provide two methods called calculateArea() and calculatePerimeter() that return the calculated area and perimeter based on the current radius value.**



**6. Write a Java program to create a class called Car with private instance variables company_name, model_name, year, and mileage. Provide public getter and setter methods to access and modify the company_name, model_name, and year variables. However, only provide a getter method for the mileage variable.**

**7. Write a Java program to create a class called Student with private instance variables student_id, student_name, and grades. Provide public getter and setter methods to access and modify the student_id and student_name variables. However, provide a method called addGrade() that allows adding a grade to the grades variable while performing additional validation.**

**8. Write a Java program to create a class called "Book" with private instance variables title, author, and price. Provide public getter and setter methods to access and modify these variables. Add a method called applyDiscount() that takes a percentage as a parameter and reduces the price by that percentage.**

**9. Write a Java program to create a class called Smartphone with private instance variables brand, model, and storageCapacity. Provide public getter and setter methods to access and modify these variables. Add a method called increaseStorage() that takes an integer value and increases the storageCapacity by that value.**

**10. Write a Java program to create a class called Desktop with private instance variables brand, processor, and ramSize. Provide public getter and setter methods to access and modify these variables. Add a method called upgradeRam() that takes an integer value and increases the ramSize by that value.**

**11. Write a Java program to create a class called House with private instance variables address, numberOfRooms, and area. Provide public getter and setter methods to access and modify these variables. Add a method called calculatePrice() that returns the price of the house based on its area and a price per square meter.**

**12. Write a Java program to create a class called Account with private instance variables accountNumber, accountHolder, and balance. Provide public getter and setter methods to access and modify these variables. Add a method called deposit() that takes an amount and increases the balance by that amount, and a method called withdraw() that takes an amount and decreases the balance by that amount.**



**13. Write a Java program to create a class called Movie with private instance variables title, director, and duration. Provide public getter and setter methods to access and modify these variables. Add a method called getMovieDetails() that returns a formatted string containing the movie details.**

**14. Write a Java program to create a class called Product with private instance variables productName, productCode, and price. Provide public getter and setter methods to access and modify these variables. Add a method called applyDiscount() that takes a percentage and reduces the price by that percentage.**

# INHERITANCE

**1. Write a Java program to create a class called Animal with a method called makeSound(). Create a subclass called Cat that overrides the makeSound() method to bark.**

**2. Write a Java program to create a class called Vehicle with a method called drive(). Create a subclass called Car that overrides the drive() method to print "Repairing a car".**

**3. Write a Java program to create a class called Shape with a method called getArea(). Create a subclass called Rectangle that overrides the getArea() method to calculate the area of a rectangle.**

**4. Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().**



**5. Write a Java program to create a class known as "BankAccount" with methods called deposit() and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.**

**6. Write a Java program to create a class called Animal with a method named move(). Create a subclass called Cheetah that overrides the move() method to run.**

**7. Write a Java program to create a class known as Person with methods called getFirstName() and getLastName(). Create a subclass called Employee that adds a new method named getEmployeeId() and overrides the getLastName() method to include the employee's job title.**

**8. Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle.**

**9. Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed.**

**10. Write a Java program that creates a class hierarchy for employees of a company. The base class should be Employee, with subclasses Manager, Developer, and Programmer. Each subclass should have properties such as name, address, salary, and job title. Implement methods for calculating bonuses, generating performance reports, and managing projects.**

# POLYMORPHISM

**1. Write a Java program to create a base class Animal (Animal Family) with a method called Sound(). Create two subclasses Bird and Cat. Override the Sound() method in each subclass to make a specific sound for each animal.**

**2. Write a Java program to create a class Vehicle with a method called speedUp(). Create two subclasses Car and Bicycle. Override the speedUp() method in each subclass to increase the vehicle's speed differently.**

**3. Write a Java program to create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.**

**4. Write a Java program to create a class Employee with a method called calculateSalary(). Create two subclasses Manager and Programmer. In each subclass, override the calculateSalary() method to calculate and return the salary based on their specific roles.**

**5. Write a Java program to create a base class Sports with a method called play(). Create three subclasses: Football, Basketball, and Rugby. Override the play() method in each subclass to play a specific statement for each sport.**



**6. Write a Java program to create a class Shape with methods getArea() and getPerimeter(). Create three subclasses: Circle, Rectangle, and Triangle. Override the getArea() and getPerimeter() methods in each subclass to calculate and return the area and perimeter of the respective shapes.**

**7. Write a Java program to create a base class Animal with methods move() and makeSound(). Create two subclasses Bird and Panthera. Override the move() method in each subclass to describe how each animal moves. Also, override the makeSound() method in each subclass to make a specific sound for each animal.**

**8. Write a Java program to create a base class Shape with methods draw() and calculateArea(). Create three subclasses: Circle, Square, and Triangle. Override the draw() method in each subclass to draw the respective shape, and override the calculateArea() method to calculate and return the area of each shape.**

**9. Write a Java program to create a base class BankAccount with methods deposit() and withdraw(). Create two subclasses SavingsAccount and CheckingAccount.**

**Override the withdraw() method in each subclass to impose different withdrawal limits and fees.**

**10. Write a Java program to create a base class Animal with methods eat() and sound(). Create three subclasses: Lion, Tiger, and Panther. Override the eat() method in each subclass to describe what each animal eats. In addition, override the sound() method to make a specific sound for each animal.**

**11. Write a Java program to create a base class Vehicle with methods startEngine() and stopEngine(). Create two subclasses Car and Motorcycle. Override the startEngine() and stopEngine() methods in each subclass to start and stop the engines differently.**

**12. Write a Java program to create a base class Shape with methods draw() and calculateArea(). Create two subclasses Circle and Cylinder. Override the draw() method in each subclass to draw the respective shape. In addition, override the calculateArea() method in the Cylinder subclass to calculate and return the total surface area of the cylinder.**

# INTERFACE

**1. Write a Java program to create an interface Shape with the getArea() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getArea() method for each of the three classes.**

**2. Write a Java program to create a Animal interface with a method called bark() that takes no arguments and returns void. Create a Dog class that implements Animal and overrides speak() to print "Dog is barking".**

**3. Write a Java program to create an interface Flyable with a method called fly_obj(). Create three classes Spacecraft, Airplane, and Helicopter that implement the Flyable interface. Implement the fly_obj() method for each of the three classes.**

**4. Write a Java programming to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw, calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods.**



**5. Write a Java program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.**

**6. Write a Java program to create an interface Drawable with a method draw() that takes no arguments and returns void. Create three classes Circle, Rectangle, and Triangle that implement the Drawable interface and override the draw() method to draw their respective shapes.**

**7. Write a Java program to create an interface Sortable with a method sort() that sorts an array of integers in ascending order. Create two classes BubbleSort and SelectionSort that implement the Sortable interface and provide their own implementations of the sort() method.**

**8. Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.**

**9. Write a Java program to create an interface Searchable with a method search(String keyword) that searches for a given keyword in a text document. Create two classes Document and WebPage that implement the Searchable interface and provide their own implementations of the search() method.**

**10. Write a Java program to create an interface Encryptable with methods encrypt (String data) and decrypt (String encryptedData) that define encryption and decryption operations. Create two classes AES and RSA that implement the Encryptable interface and provide their own encryption and decryption algorithms.**

**11. Write a Java program to create an interface Sortable with a method sort (int[] array) that sorts an array of integers in descending order. Create two classes QuickSort and MergeSort that implement the Sortable interface and provide their own implementations of the sort() method.**

# ABSTRACT CLASS

**1. Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion and Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal.**

**2. Write a Java program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.**

**3. Write a Java program to create an abstract class BankAccount with abstract methods deposit() and withdraw(). Create subclasses: SavingsAccount and CurrentAccount that extend the BankAccount class and implement the respective methods to handle deposits and withdrawals for each account type.**

**4. Write a Java program to create an abstract class Animal with abstract methods eat() and sleep(). Create subclasses Lion, Tiger, and Deer that extend the Animal class and implement the eat() and sleep() methods differently based on their specific behavior.**



**5. Write a Java program to create an abstract class Employee with abstract methods calculateSalary() and displayInfo(). Create subclasses Manager and Programmer that extend the Employee class and implement the respective methods to calculate salary and display information for each role.**

**6. Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.**

**7. Write a Java program to create an abstract class Vehicle with abstract methods startEngine() and stopEngine(). Create subclasses Car and Motorcycle that extend the Vehicle class and implement the respective methods to start and stop the engines for each vehicle type.**

**8. Write a Java program to create an abstract class Person with abstract methods eat() and exercise(). Create subclasses Athlete and LazyPerson that extend the Person class and implement the respective methods to describe how each person eats and exercises.**

**9. Write a Java program to create an abstract class Instrument with abstract methods play() and tune(). Create subclasses for Glockenspiel and Violin that extend the Instrument class and implement the respective methods to play and tune each instrument.**

**10. Write a Java program to create an abstract class Shape2D with abstract methods draw() and resize(). Create subclasses Rectangle and Circle that extend the Shape2D class and implement the respective methods to draw and resize each shape.**

**11. Write a Java program to create an abstract class Bird with abstract methods fly() and makeSound(). Create subclasses Eagle and Hawk that extend the Bird class and implement the respective methods to describe how each bird flies and makes a sound.**

**12. Write a Java program to create an abstract class GeometricShape with abstract methods area() and perimeter(). Create subclasses Triangle and Square that extend the GeometricShape class and implement the respective methods to calculate the area and perimeter of each shape.**