**Exercise 1:**

Create a class **Animal** with a method **makeSound()**. Then, create two subclasses **Dog** and **Cat** that override the **makeSound()** method to bark and meow, respectively.

**Animal.java:**

public class Animal {

public void makeSound() {

System.out.println("Generic animal sound");

}

}

**Dog.java:**

public class Dog extends Animal {

public void makeSound() {

System.out.println("Bark");

}

}

**Cat.java:**

public class Cat extends Animal {

public void makeSound() {

System.out.println("Meow");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Animal dog = new Dog();

Animal cat = new Cat();

dog.makeSound(); // Outputs: Bark

cat.makeSound(); // Outputs: Meow

}

}

**Exercise 3:**

Create a class Vehicle with methods **startEngine()** and **stopEngine().** Then, create two subclasses Car and Motorcycle that extend the Vehicle class and implement these methods.

**Vehicle.java:**

public class Vehicle {

public void startEngine() {

System.out.println("Engine started");

}

public void stopEngine() {

System.out.println("Engine stopped");

}

}

**Car.java:**

public class Car extends Vehicle {

@Override

public void startEngine() {

System.out.println("Car engine started");

}

@Override

public void stopEngine() {

System.out.println("Car engine stopped");

}

}

**Motorcycle.java:**

public class Motorcycle extends Vehicle {

@Override

public void startEngine() {

System.out.println("Motorcycle engine started");

}

@Override

public void stopEngine() {

System.out.println("Motorcycle engine stopped");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Vehicle car = new Car();

Vehicle motorcycle = new Motorcycle();

car.startEngine(); // Outputs: Car engine started

car.stopEngine(); // Outputs: Car engine stopped

motorcycle.startEngine(); // Outputs: Motorcycle engine started

motorcycle.stopEngine(); // Outputs: Motorcycle engine stopped

}

}

**Exercise 4:**

Create a class Person with a method displayInfo(). Then, create two subclasses Student and Teacher that extend the Person class and implement the displayInfo() method to show student and teacher information.

**Person.java:**

public class Person {

public void displayInfo() {

System.out.println("Person information");

}

}

**Student.java:**

public class Student extends Person {

@Override

public void displayInfo() {

System.out.println("Student information");

}

}

**Teacher.java:**

public class Teacher extends Person {

@Override

public void displayInfo() {

System.out.println("Teacher information");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Person student = new Student();

Person teacher = new Teacher();

student.displayInfo(); // Outputs: Student information

teacher.displayInfo(); // Outputs: Teacher information

}

}

**Exercise 5:**

Create a class **BankAccount** with methods **deposit(double amount)** and **withdraw(double amount)**. Then, create two subclasses **SavingsAccount** and **CheckingAccount** that extend the **BankAccount** class and implement interest calculation for savings accounts and overdraft protection for checking accounts.

**BankAccount.java:**

public class BankAccount {

protected double balance;

public void deposit(double amount) {

balance += amount;

System.out.println("Deposited: $" + amount);

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: $" + amount);

} else {

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

}

}

}

**SavingsAccount.java:**

public class SavingsAccount extends BankAccount {

private double interestRate;

public SavingsAccount(double interestRate) {

this.interestRate = interestRate;

}

public void addInterest() {

double interest = balance \* interestRate / 100;

balance += interest;

System.out.println("Interest added: $" + interest);

}

}

**CheckingAccount.java:**

public class CheckingAccount extends BankAccount {

private double overdraftLimit;

public CheckingAccount(double overdraftLimit) {

this.overdraftLimit = overdraftLimit;

}

@Override

public void withdraw(double amount) {

if (amount <= balance + overdraftLimit) {

balance -= amount;

System.out.println("Withdrawn: $" + amount);

} else {

System.out.println("Overdraft limit exceeded!");

}

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

SavingsAccount savingsAccount = new SavingsAccount(2.5);

CheckingAccount checkingAccount = new CheckingAccount(1000);

savingsAccount.deposit(5000);

savingsAccount.addInterest();

savingsAccount.withdraw(2000);

checkingAccount.deposit(2000);

checkingAccount.withdraw(2500);

System.out.println("Savings Account Balance: $" + savingsAccount.balance);

System.out.println("Checking Account Balance: $" + checkingAccount.balance);

}

}

**Exercise 6:**

Create a class hierarchy for geometric shapes. Start with a base class **Shape** and create subclasses **Circle**, **Rectangle**, and **Triangle**. Implement methods to calculate area and perimeter for each shape.

**Shape.java:**

public class Shape {

public double calculateArea() {

return 0.0;

}

public double calculatePerimeter() {

return 0.0;

}

}

**Circle.java:**

public class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

@Override

public double calculatePerimeter() {

return 2 \* Math.PI \* radius;

}

}

**Rectangle.java:**

public class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

@Override

public double calculateArea() {

return length \* width;

}

@Override

public double calculatePerimeter() {

return 2 \* (length + width);

}

}

**Triangle.java:**

public class Triangle extends Shape {

private double side1;

private double side2;

private double side3;

public Triangle(double side1, double side2, double side3) {

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

@Override

public double calculateArea() {

// Implement area calculation using Heron's formula

double s = (side1 + side2 + side3) / 2;

return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

@Override

public double calculatePerimeter() {

return side1 + side2 + side3;

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Circle circle = new Circle(5);

Rectangle rectangle = new Rectangle(4, 6);

Triangle triangle = new Triangle(3, 4, 5);

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

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

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

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

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

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

}

}

**Exercise 7:**

Create a class hierarchy for employees in a company. Start with a base class **Employee** and create subclasses **Manager**, **Developer**, and **Intern**. Each subclass should have specific attributes such as salary, role, and additional methods related to their roles.

**Employee.java:**

public class Employee {

private String name;

private double salary;

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

public void displayInfo() {

System.out.println("Name: " + name);

System.out.println("Salary: $" + salary);

}

}

**Manager.java:**

public class Manager extends Employee {

private String department;

public Manager(String name, double salary, String department) {

super(name, salary);

this.department = department;

}

public void organizeMeeting() {

System.out.println("Manager is organizing a meeting.");

}

}

**Developer.java:**

public class Developer extends Employee {

private String programmingLanguage;

public Developer(String name, double salary, String programmingLanguage) {

super(name, salary);

this.programmingLanguage = programmingLanguage;

}

public void writeCode() {

System.out.println("Developer is writing code in " + programmingLanguage);

}

}

**Intern.java:**

public class Intern extends Employee {

private String mentor;

public Intern(String name, double salary, String mentor) {

super(name, salary);

this.mentor = mentor;

}

public void askForHelp() {

System.out.println("Intern is asking for help from mentor: " + mentor);

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Manager manager = new Manager("John Doe", 80000, "IT");

Developer developer = new Developer("Alice Smith", 60000, "Java");

Intern intern = new Intern("Bob Johnson", 30000, "Senior Developer");

manager.displayInfo();

manager.organizeMeeting();

developer.displayInfo();

developer.writeCode();

intern.displayInfo();

intern.askForHelp();

}

}

**Exercise 8:**

Create a class hierarchy for a simple zoo simulation. Start with a base class **Animal** and create subclasses **Mammal**, **Bird**, and **Fish**. Each subclass should have specific attributes and methods, and you can add more specific animal classes as needed.

**Animal.java:**

public class Animal {

private String name;

public Animal(String name) {

this.name = name;

}

public void makeSound() {

System.out.println("Generic animal sound");

}

}

**Mammal.java:**

public class Mammal extends Animal {

private boolean isWarmBlooded;

public Mammal(String name, boolean isWarmBlooded) {

super(name);

this.isWarmBlooded = isWarmBlooded;

}

@Override

public void makeSound() {

System.out.println("Mammal sound");

}

public void nurseYoung() {

System.out.println("Mammal is nursing its young.");

}

}

**Bird.java:**

public class Bird extends Animal {

private boolean canFly;

public Bird(String name, boolean canFly) {

super(name);

this.canFly = canFly;

}

@Override

public void makeSound() {

System.out.println("Bird sound");

}

public void buildNest() {

System.out.println("Bird is building a nest.");

}

}

**Fish.java:**

public class Fish extends Animal {

private boolean isSaltwater;

public Fish(String name, boolean isSaltwater) {

super(name);

this.isSaltwater = isSaltwater;

}

@Override

public void makeSound() {

System.out.println("Fish sound");

}

public void swim() {

System.out.println("Fish is swimming.");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Mammal lion = new Mammal("Lion", true);

Bird eagle = new Bird("Eagle", true);

Fish shark = new Fish("Shark", false);

lion.makeSound();

lion.nurseYoung();

eagle.makeSound();

eagle.buildNest();

shark.makeSound();

shark.swim();

}

}

**Exercise 9:**

Create a class hierarchy for a university system. Start with a base class **Person** and create subclasses **Student** and **Professor**. Add more fields and methods to capture additional information.

**Person.java:**

public class Person {

private String name;

private int age;

private String address;

private String phoneNumber;

public Person(String name, int age, String address, String phoneNumber) {

this.name = name;

this.age = age;

this.address = address;

this.phoneNumber = phoneNumber;

}

public void displayInfo() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Address: " + address);

System.out.println("Phone Number: " + phoneNumber);

}

}

**Student.java:**

public class Student extends Person {

private int studentId;

private String major;

private double gpa;

public Student(String name, int age, String address, String phoneNumber, int studentId, String major, double gpa) {

super(name, age, address, phoneNumber);

this.studentId = studentId;

this.major = major;

this.gpa = gpa;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Student ID: " + studentId);

System.out.println("Major: " + major);

System.out.println("GPA: " + gpa);

}

public void study() {

System.out.println("Student is studying.");

}

}

**Professor.java:**

public class Professor extends Person {

private String department;

private int employeeId;

private String researchArea;

public Professor(String name, int age, String address, String phoneNumber, String department, int employeeId, String researchArea) {

super(name, age, address, phoneNumber);

this.department = department;

this.employeeId = employeeId;

this.researchArea = researchArea;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Department: " + department);

System.out.println("Employee ID: " + employeeId);

System.out.println("Research Area: " + researchArea);

}

public void conductResearch() {

System.out.println("Professor is conducting research.");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Student student = new Student("Alice", 20, "123 Main St", "555-1234", 12345, "Computer Science", 3.8);

Professor professor = new Professor("Dr. Smith", 45, "456 Oak St", "555-5678", "Engineering", 9876, "Machine Learning");

student.displayInfo();

student.study();

professor.displayInfo();

professor.conductResearch();

}

}

**Exercise 10:**

Create a class hierarchy for a multimedia library. Start with a base class **MediaItem** and create subclasses **Book**, **Movie**, and **MusicAlbum**. Add more fields and methods to represent details specific to each type of media.

**MediaItem.java:**

public class MediaItem {

private String title;

private int year;

private String genre;

public MediaItem(String title, int year, String genre) {

this.title = title;

this.year = year;

this.genre = genre;

}

public void displayInfo() {

System.out.println("Title: " + title);

System.out.println("Year: " + year);

System.out.println("Genre: " + genre);

}

}

**Book.java:**

public class Book extends MediaItem {

private String author;

private int pageCount;

private String publisher;

public Book(String title, int year, String genre, String author, int pageCount, String publisher) {

super(title, year, genre);

this.author = author;

this.pageCount = pageCount;

this.publisher = publisher;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Author: " + author);

System.out.println("Page Count: " + pageCount);

System.out.println("Publisher: " + publisher);

}

public void read() {

System.out.println("Reading the book.");

}

}

**Movie.java:**

public class Movie extends MediaItem {

private String director;

private double duration;

private String rating;

public Movie(String title, int year, String genre, String director, double duration, String rating) {

super(title, year, genre);

this.director = director;

this.duration = duration;

this.rating = rating;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Director: " + director);

System.out.println("Duration: " + duration + " minutes");

System.out.println("Rating: " + rating);

}

public void watch() {

System.out.println("Watching the movie.");

}

}

**MusicAlbum.java:**

public class MusicAlbum extends MediaItem {

private String artist;

private int numberOfTracks;

private String recordLabel;

public MusicAlbum(String title, int year, String genre, String artist, int numberOfTracks, String recordLabel) {

super(title, year, genre);

this.artist = artist;

this.numberOfTracks = numberOfTracks;

this.recordLabel = recordLabel;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Artist: " + artist);

System.out.println("Number of Tracks: " + numberOfTracks);

System.out.println("Record Label: " + recordLabel);

}

public void listen() {

System.out.println("Listening to the music album.");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Book book = new Book("The Catcher in the Rye", 1951, "Fiction", "J.D. Salinger", 224, "Little, Brown and Company");

Movie movie = new Movie("Inception", 2010, "Sci-Fi", "Christopher Nolan", 148.5, "PG-13");

MusicAlbum musicAlbum = new MusicAlbum("Abbey Road", 1969, "Rock", "The Beatles", 17, "Apple Records");

book.displayInfo();

book.read();

movie.displayInfo();

movie.watch();

musicAlbum.displayInfo();

musicAlbum.listen();

}

}

**Exercise 11:**

Create a class hierarchy for a retail store system. Start with a base class **Product** and create subclasses **Electronics**, **Clothing**, and **Food**. Each subclass should have 7 to 10 fields representing different product attributes.

**Product.java:**

public class Product {

private String productName;

private double price;

private String brand;

private int quantity;

private boolean inStock;

private String description;

public Product(String productName, double price, String brand, int quantity, boolean inStock, String description) {

this.productName = productName;

this.price = price;

this.brand = brand;

this.quantity = quantity;

this.inStock = inStock;

this.description = description;

}

public void displayInfo() {

System.out.println("Product: " + productName);

System.out.println("Price: $" + price);

System.out.println("Brand: " + brand);

System.out.println("Quantity: " + quantity);

System.out.println("In Stock: " + inStock);

System.out.println("Description: " + description);

}

}

**Electronics.java:**

public class Electronics extends Product {

private String type;

private String model;

private String warranty;

public Electronics(String productName, double price, String brand, int quantity, boolean inStock, String description, String type, String model, String warranty) {

super(productName, price, brand, quantity, inStock, description);

this.type = type;

this.model = model;

this.warranty = warranty;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Type: " + type);

System.out.println("Model: " + model);

System.out.println("Warranty: " + warranty);

}

public void powerOn() {

System.out.println("Electronics is powering on.");

}

}

**Clothing.java:**

public class Clothing extends Product {

private String size;

private String color;

private String material;

public Clothing(String productName, double price, String brand, int quantity, boolean inStock, String description, String size, String color, String material) {

super(productName, price, brand, quantity, inStock, description);

this.size = size;

this.color = color;

this.material = material;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Size: " + size);

System.out.println("Color: " + color);

System.out.println("Material: " + material);

}

public void tryOn() {

System.out.println("Customer is trying on the clothing.");

}

}

**Food.java:**

public class Food extends Product {

private String expirationDate;

private boolean isPerishable;

private String allergens;

public Food(String productName, double price, String brand, int quantity, boolean inStock, String description, String expirationDate, boolean isPerishable, String allergens) {

super(productName, price, brand, quantity, inStock, description);

this.expirationDate = expirationDate;

this.isPerishable = isPerishable;

this.allergens = allergens;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Expiration Date: " + expirationDate);

System.out.println("Is Perishable: " + isPerishable);

System.out.println("Allergens: " + allergens);

}

public void cook() {

System.out.println("Cooking the food.");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Electronics phone = new Electronics("Smartphone", 499.99, "Samsung", 50, true, "Latest model", "Mobile", "Galaxy S21", "1 year");

Clothing shirt = new Clothing("T-Shirt", 19.99, "Nike", 100, true, "Casual shirt", "Large", "Red", "Cotton");

Food chocolate = new Food("Dark Chocolate", 5.99, "Lindt", 200, true, "70% cocoa", "2023-12-31", false, "May contain nuts");

phone.displayInfo();

phone.powerOn();

shirt.displayInfo();

shirt.tryOn();

chocolate.displayInfo();

chocolate.cook();

}

}

**Exercise 12:**

Create a class hierarchy for a transportation system. Start with a base class **Vehicle** and create subclasses **Car**, **Bus**, and **Bicycle**. Each subclass should have 7 to 10 fields representing different vehicle attributes.

**Vehicle.java:**

public class Vehicle {

private String make;

private String model;

private int year;

private double price;

private boolean isAvailable;

private String color;

private String fuelType;

private int seatingCapacity;

private double mileage;

public Vehicle(String make, String model, int year, double price, boolean isAvailable, String color, String fuelType, int seatingCapacity, double mileage) {

this.make = make;

this.model = model;

this.year = year;

this.price = price;

this.isAvailable = isAvailable;

this.color = color;

this.fuelType = fuelType;

this.seatingCapacity = seatingCapacity;

this.mileage = mileage;

}

public void displayInfo() {

System.out.println("Make: " + make);

System.out.println("Model: " + model);

System.out.println("Year: " + year);

System.out.println("Price: $" + price);

System.out.println("Is Available: " + isAvailable);

System.out.println("Color: " + color);

System.out.println("Fuel Type: " + fuelType);

System.out.println("Seating Capacity: " + seatingCapacity);

System.out.println("Mileage: " + mileage + " mpg");

}

}

**Car.java:**

public class Car extends Vehicle {

private int numberOfDoors;

private boolean isConvertible;

private String transmissionType;

public Car(String make, String model, int year, double price, boolean isAvailable, String color, String fuelType, int seatingCapacity, double mileage, int numberOfDoors, boolean isConvertible, String transmissionType) {

super(make, model, year, price, isAvailable, color, fuelType, seatingCapacity, mileage);

this.numberOfDoors = numberOfDoors;

this.isConvertible = isConvertible;

this.transmissionType = transmissionType;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Number of Doors: " + numberOfDoors);

System.out.println("Is Convertible: " + isConvertible);

System.out.println("Transmission Type: " + transmissionType);

}

public void startEngine() {

System.out.println("Car engine started.");

}

}

**Bus.java:**

public class Bus extends Vehicle {

private int numberOfSeats;

private String busType;

private boolean isDoubleDecker;

public Bus(String make, String model, int year, double price, boolean isAvailable, String color, String fuelType, int seatingCapacity, double mileage, int numberOfSeats, String busType, boolean isDoubleDecker) {

super(make, model, year, price, isAvailable, color, fuelType, seatingCapacity, mileage);

this.numberOfSeats = numberOfSeats;

this.busType = busType;

this.isDoubleDecker = isDoubleDecker;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Number of Seats: " + numberOfSeats);

System.out.println("Bus Type: " + busType);

System.out.println("Is Double Decker: " + isDoubleDecker);

}

public void openDoors() {

System.out.println("Opening bus doors.");

}

}

**Bicycle.java:**

public class Bicycle extends Vehicle {

private int numberOfGears;

private String bicycleType;

private boolean hasBasket;

public Bicycle(String make, String model, int year, double price, boolean isAvailable, String color, String fuelType, int seatingCapacity, double mileage, int numberOfGears, String bicycleType, boolean hasBasket) {

super(make, model, year, price, isAvailable, color, fuelType, seatingCapacity, mileage);

this.numberOfGears = numberOfGears;

this.bicycleType = bicycleType;

this.hasBasket = hasBasket;

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Number of Gears: " + numberOfGears);

System.out.println("Bicycle Type: " + bicycleType);

System.out.println("Has Basket: " + hasBasket);

}

public void ride() {

System.out.println("Riding the bicycle.");

}

}

**Main.java:**

public class Main {

public static void main(String[] args) {

Car sedan = new Car("Toyota", "Camry", 2022, 25000, true, "Blue", "Gasoline", 5, 30.5, 4, false, "Automatic");

Bus cityBus = new Bus("Mercedes", "Sprinter", 2023, 80000, true, "Yellow", "Diesel", 30, 10.5, 40, "City Bus", true);

Bicycle mountainBike = new Bicycle("Schwinn", "High Timber", 2021, 300, true, "Black", "Manual", 1, 25, 21, "Mountain Bike", false);

sedan.displayInfo();

sedan.startEngine();

cityBus.displayInfo();

cityBus.openDoors();

mountainBike.displayInfo();

mountainBike.ride();

}

}