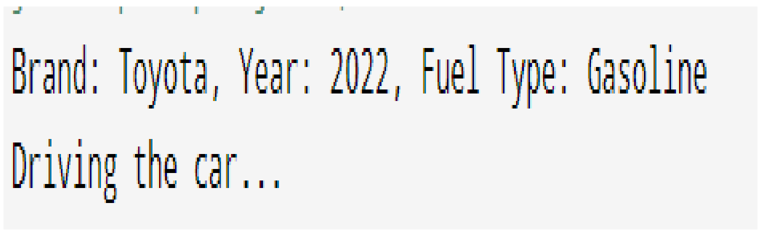
JAVA LAB – WEEK 5 PROGRAMS

INHERITANCE

1.Design a simple single inheritance system for a vehicle management application: create a base class Vehicle with attributes brand and year, and a method displayInfo() that returns the vehicle's details. Extend this class into a Car class that adds an attribute fuelType and a method drive(). Implement and demonstrate these classes by creating an instance of Car, calling displayInfo(), and using the drive() method to simulate driving the car.

```
class Vehicle {
    private String brand;
    private int year;
    public Vehicle(String brand, int year) {
        this.brand = brand;
        this.year = year;
    }
    public String displayInfo() {
        return "Brand: " + brand + ", Year: " + year;
    }
}
class Car extends Vehicle {
    private String fuelType;
    public Car(String brand, int year, String fuelType) {
        super(brand, year);
        this.fuelType = fuelType;
    }
    public void drive() {
        System.out.println("Driving the car...");
    }
@Override
    public String displayInfo() {
        return super.displayInfo() + ", Fuel Type: " + fuelType;
    }
}
public class VehicleTest {
    public static void main(String[] args) {
        Car myCar = new Car("Toyota", 2022, "Gasoline");
        System.out.println(myCar.displayInfo());
        myCar.drive();
    }
}
```

OUTPUT



```
Brand: Toyota, Year: 2022, Fuel Type: Gasoline
Driving the car...
```

2.Create a library management system in Java with two classes: Book and EBook. The Book class features attributes like title, author, ISBN, and pages, along with a method to display its details. The EBook class extends Book by adding an attribute for fileSize in MB and modifies the display method to include this new attribute. In a LibraryTest class, instantiate objects of both Book and EBook, configure their attributes, and demonstrate the use of the enhanced display functionality.

```java
class Book {
    private String title;
    private String author;
    private String ISBN;
    private int pages;
    public Book(String title, String author, String ISBN, int pages) {
        this.title = title;
        this.author = author;
        this.ISBN = ISBN;
        this.pages = pages;
    }
    public String displayDetails() {
        return "Title: " + title + ", Author: " + author + ", ISBN: " + ISBN + ", Pages: " + pages;
    }
}
class EBook extends Book {
    private double fileSize;
    public EBook(String title, String author, String ISBN, int pages, double fileSize) {
        super(title, author, ISBN, pages);
        this.fileSize = fileSize;
    }
    @Override
    public String displayDetails() {
        return super.displayDetails() + ", File Size: " + fileSize + " MB";
    }
}
public class LibraryTest {
    public static void main(String[] args) {
        Book book = new Book("The Great Gatsby", "F. Scott Fitzgerald", "9780743273565", 180);
        EBook eBook = new EBook("Digital Fortress", "Dan Brown", "9780552161233", 350, 2.5);
        System.out.println(book.displayDetails());
        System.out.println(eBook.displayDetails());
    }
}
```

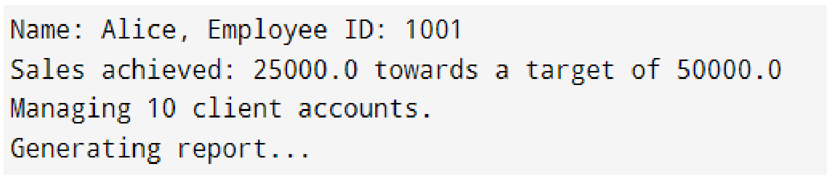
OUTPUT

```
Title: The Great Gatsby, Author: F. Scott Fitzgerald, ISBN: 9780743273565, Pages: 180
Title: Digital Fortress, Author: Dan Brown, ISBN: 9780552161233, Pages: 350, File Size
    : 2.5 MB
```

3.Create a multilevel inheritance hierarchy for a company's employee management system: start with a base class Employee with attributes name and employeeId, and a method getDetails(). Extend it into a SalesEmployee class with an additional attribute salesTarget and a method achieveSales(). Further extend SalesEmployee into an AccountManager class, adding an attribute clientAccounts and methods to manageAccounts() and generateReport(). Implement and demonstrate these classes by creating an instance of AccountManager, calling getDetails(), and using the specific methods for sales and account management tasks.

```
class Employee {
    private String name;
    private int employeeId;

    public Employee(String name, int employeeId) {
        this.name = name;
        this.employeeId = employeeId;
    }
    public String getDetails() {
        return "Name: " + name + ", Employee ID: " + employeeId;
    }
}
class SalesEmployee extends Employee {
    private double salesTarget;

    public SalesEmployee(String name, int employeeId, double salesTarget) {
        super(name, employeeId);
        this.salesTarget = salesTarget;
    }

    public void achieveSales(double amount) {
        System.out.println("Sales achieved: " + amount + " towards a target of " + salesTarget);
    }
}

class AccountManager extends SalesEmployee {
    private int clientAccounts;

    public AccountManager(String name, int employeeId, double salesTarget, int
clientAccounts) {
        super(name, employeeId, salesTarget);
        this.clientAccounts = clientAccounts;
    }
    public void manageAccounts() {
        System.out.println("Managing " + clientAccounts + " client accounts.");
    }
    public void generateReport() {
        System.out.println("Generating report...");
    }
}
public class EmployeeTest {
    public static void main(String[] args) {
        AccountManager manager = new AccountManager("Alice", 1001, 50000.00, 10);
        System.out.println(manager.getDetails());
        manager.achieveSales(25000.00);
        manager.manageAccounts();
        manager.generateReport();
    }
}
```

OUTPUT

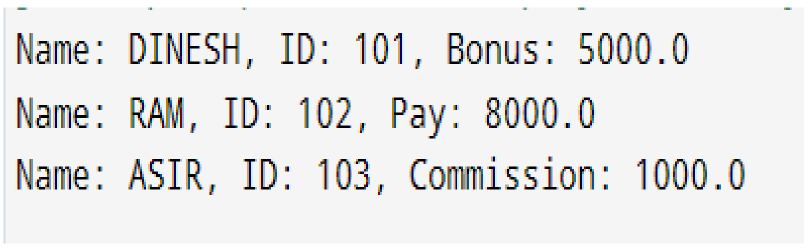


```
Name: Alice, Employee ID: 1001
Sales achieved: 25000.0 towards a target of 50000.0
Managing 10 client accounts.
Generating report...
```

4.Create a base class Employee with common attributes and a getDetails() method. Extend it into Manager, Developer, and Salesperson, each with additional attributes and methods: Manager calculates a bonus, Developer computes pay based on hours worked, and Salesperson calculates commission. Implement and demonstrate these classes by creating instances

```
abstract class Employee {
    private String name;
    private int id;
    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }
    public abstract String getDetails();
    public String getName() {
        return name;
    }
    public int getId() {
        return id;
    }
}
class Manager extends Employee {
    private double bonus;
    public Manager(String name, int id, double bonus) {
        super(name, id);
        this.bonus = bonus;
    }
    public double calculateBonus() {
        return bonus;
    }
    @Override
    public String getDetails() {
        return "Name: " + getName() + ", ID: " + getId() + ", Bonus: " + calculateBonus();
    }
}
class Developer extends Employee {
    private int hoursWorked;
    private double hourlyRate;

    public Developer(String name, int id, int hoursWorked, double hourlyRate) {
        super(name, id);
        this.hoursWorked = hoursWorked;
        this.hourlyRate = hourlyRate;
    }
    public double computePay() {
        return hoursWorked * hourlyRate;
    }
    @Override
    public String getDetails() {
        return "Name: " + getName() + ", ID: " + getId() + ", Pay: " + computePay();
    }
}
class Salesperson extends Employee {
    private double salesAmount;
    private double commissionRate;
    public Salesperson(String name, int id, double salesAmount, double commissionRate) {
        super(name, id);
        this.salesAmount = salesAmount;
        this.commissionRate = commissionRate;
    }
    public double calculateCommission() {
        return salesAmount * commissionRate;
    }
    @Override
    public String getDetails() {
        return "Name: " + getName() + ", ID: " + getId() + ", Commission: " +
calculateCommission();
    }
}
public class EmployeeHierarchyTest {
    public static void main(String[] args) {
        Manager manager = new Manager("DINESH", 101, 5000);
        Developer developer = new Developer("RAM", 102, 160, 50);
        Salesperson salesperson = new Salesperson("ASIR", 103, 20000, 0.05);
        System.out.println(manager.getDetails());
        System.out.println(developer.getDetails());
        System.out.println(salesperson.getDetails());
    }
}
```

OUTPUT



```
Name: DINESH, ID: 101, Bonus: 5000.0
Name: RAM, ID: 102, Pay: 8000.0
Name: ASIR, ID: 103, Commission: 1000.0
```