Week 1 : (ALGORITHMS_DATASTRUCTURES)

Task 1 : (Inventory management system)

Product.java :

```java
package inventory;

public class Product {
   private int productId;
   private String productName;
   private int quantity;
   private double price;

   public Product(int productId, String productName, int quantity, double price) {
      this.productId = productId;
      this.productName = productName;
      this.quantity = quantity;
      this.price = price;
   }

   public int getProductId() { return productId; }
   public String getProductName() { return productName; }
   public int getQuantity() { return quantity; }
   public double getPrice() { return price; }

   public void setQuantity(int quantity) { this.quantity = quantity; }
   public void setPrice(double price) { this.price = price; }

   @Override
   public String toString() {
      return productId + " | " + productName + " | Qty: " + quantity + " | ₹" + price;
   }
}
```

InventoryManager.java :

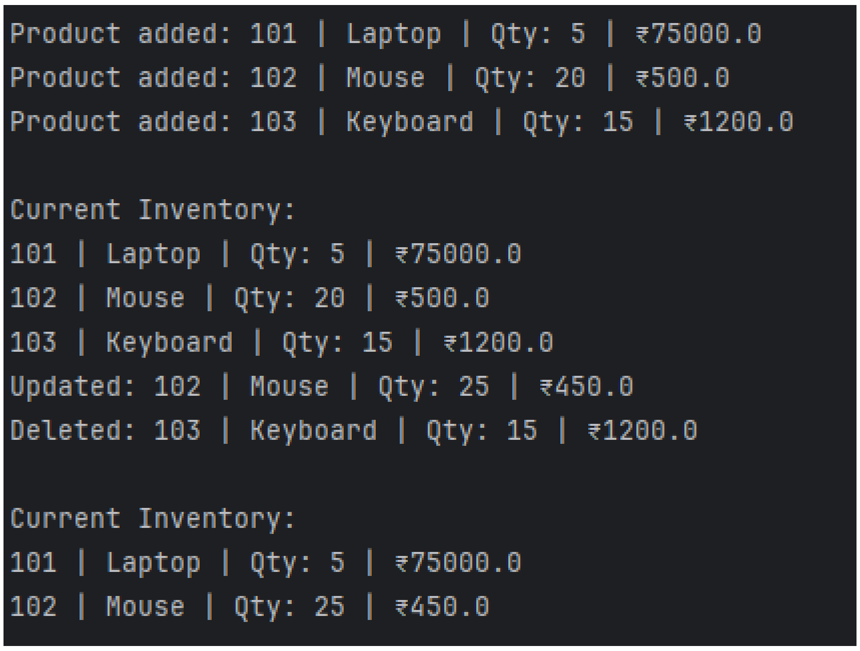
```java
package inventory;

import java.util.HashMap;

public class InventoryManager {
   private HashMap<Integer, Product> inventory = new HashMap<>();

   public void addProduct(Product product) {
      inventory.put(product.getProductId(), product);
      System.out.println("Product added: " + product);
```

```java
    }

    public void updateProduct(int productId, int newQty, double newPrice) {
        Product product = inventory.get(productId);
        if (product != null) {
            product.setQuantity(newQty);
            product.setPrice(newPrice);
            System.out.println("Updated: " + product);
        } else {
            System.out.println("Product not found.");
        }
    }

    public void deleteProduct(int productId) {
        Product removed = inventory.remove(productId);
        if (removed != null) {
            System.out.println("Deleted: " + removed);
        } else {
            System.out.println("Product not found.");
        }
    }

    public void showAllProducts() {
        System.out.println("\nCurrent Inventory:");
        for (Product p : inventory.values()) {
            System.out.println(p);
        }
    }
}
```

InventoryTest.java :

```java
package inventory;

public class InventoryTest {
    public static void main(String[] args) {
        InventoryManager manager = new InventoryManager();


        manager.addProduct(new Product(101, "Laptop", 5, 75000));
        manager.addProduct(new Product(102, "Mouse", 20, 500));
        manager.addProduct(new Product(103, "Keyboard", 15, 1200));

        manager.showAllProducts();


        manager.updateProduct(102, 25, 450);
        manager.deleteProduct(103);

        manager.showAllProducts();
    }
}
```

OUTPUT :



```
Product added: 101 | Laptop | Qty: 5 | ₹75000.0
Product added: 102 | Mouse | Qty: 20 | ₹500.0
Product added: 103 | Keyboard | Qty: 15 | ₹1200.0

Current Inventory:
101 | Laptop | Qty: 5 | ₹75000.0
102 | Mouse | Qty: 20 | ₹500.0
103 | Keyboard | Qty: 15 | ₹1200.0
Updated: 102 | Mouse | Qty: 25 | ₹450.0
Deleted: 103 | Keyboard | Qty: 15 | ₹1200.0

Current Inventory:
101 | Laptop | Qty: 5 | ₹75000.0
102 | Mouse | Qty: 25 | ₹450.0
```

TASK 2 : (Ecommerence Search Function):

Product.java :

```
package search;

public class Product {
    int productId;
    String productName;
    String category;

    public Product(int productId, String productName, String category) {
        this.productId = productId;
        this.productName = productName;
        this.category = category;
    }

    @Override
    public String toString() {
        return productId + " - " + productName + " (" + category + ")";
    }
}
```

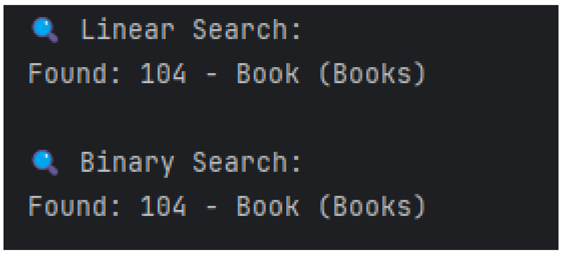
SearchAlgorithms.java:

```
package search;

import java.util.Arrays;
import java.util.Comparator;

public class SearchAlgorithms {


    public static Product linearSearch(Product[] products, int targetId) {
        for (Product p : products) {
            if (p.productId == targetId) {
                return p;
            }
        }
        return null;
    }


    public static Product binarySearch(Product[] products, int targetId) {
        int left = 0;
        int right = products.length - 1;

        while (left <= right) {
            int mid = (left + right) / 2;
            if (products[mid].productId == targetId) {
                return products[mid];
            } else if (products[mid].productId < targetId) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return null;
    }


    public static void sortByProductId(Product[] products) {
        Arrays.sort(products, Comparator.comparingInt(p -> p.productId));
    }
```

EcommerceSearchTest.java:

```java
package search;

public class EcommerceSearchTest {
  public static void main(String[] args) {
    Product[] inventory = {
        new Product(103, "Keyboard", "Electronics"),
        new Product(101, "Laptop", "Electronics"),
        new Product(105, "Shoes", "Fashion"),
        new Product(104, "Book", "Books"),
        new Product(102, "Mouse", "Electronics")
    };

     System.out.println("🔍 Linear Search:");
    Product foundLinear = SearchAlgorithms.linearSearch(inventory, 104);
    System.out.println(foundLinear != null ? "Found: " + foundLinear : "Product not found");
    SearchAlgorithms.sortByProductId(inventory);
    System.out.println("\n🔍 Binary Search:");
    Product foundBinary = SearchAlgorithms.binarySearch(inventory, 104);
    System.out.println(foundBinary != null ? "Found: " + foundBinary : "Product not found");
  }
}
```

OUTPUT :



```
🔍 Linear Search:
Found: 104 - Book (Books)

🔍 Binary Search:
Found: 104 - Book (Books)
```

TASK 3 : (Customer orders sort) :

Order.java:

```java
package sorting;

public class Order {
  int orderId;
  String customerName;
  double totalPrice;

  public Order(int orderId, String customerName, double totalPrice) {
    this.orderId = orderId;
    this.customerName = customerName;
    this.totalPrice = totalPrice;
  }

  @Override
  public String toString() {
    return "Order ID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice;
  }
}
```

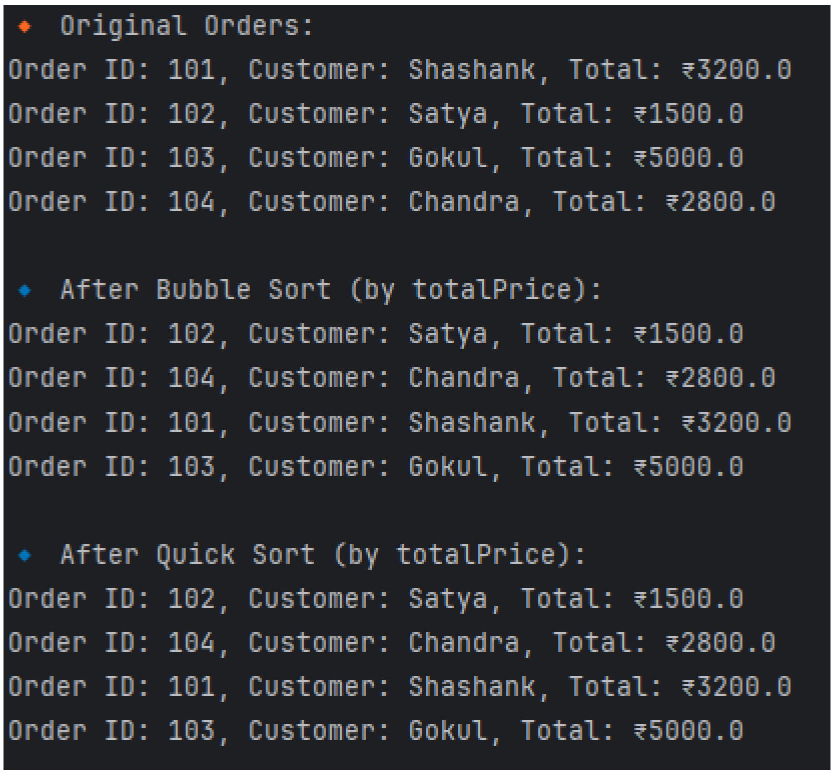
OrderSorter.java:

```
package sorting;

public class Order {
    int orderId;
    String customerName;
    double totalPrice;

    public Order(int orderId, String customerName, double totalPrice) {
        this.orderId = orderId;
        this.customerName = customerName;
        this.totalPrice = totalPrice;
    }

    @Override
    public String toString() {
        return "Order ID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice;
    }
}
```

OrderSortTest.java:

```
package sorting;

public class OrderSortTest {
    public static void main(String[] args) {
        Order[] orders = {
            new Order(101, "Shashank", 3200.0),
            new Order(102, "Satya", 1500.0),
            new Order(103, "Gokul", 5000.0),
            new Order(104, "Chandra", 2800.0)
        };

        System.out.println("🔸 Original Orders:");
        OrderSorter.printOrders(orders);
        Order[] bubbleSorted = orders.clone();
        OrderSorter.bubbleSort(bubbleSorted);
        System.out.println(" 🔹 After Bubble Sort (by totalPrice):");
        OrderSorter.printOrders(bubbleSorted);


        Order[] quickSorted = orders.clone();
        OrderSorter.quickSort(quickSorted, 0, quickSorted.length - 1);
        System.out.println(" 🔹 After Quick Sort (by totalPrice):");
        OrderSorter.printOrders(quickSorted);
    }
}
```

OUTPUT :



```
🔸 Original Orders:
Order ID: 101, Customer: Shashank, Total: ₹3200.0
Order ID: 102, Customer: Satya, Total: ₹1500.0
Order ID: 103, Customer: Gokul, Total: ₹5000.0
Order ID: 104, Customer: Chandra, Total: ₹2800.0

🔹 After Bubble Sort (by totalPrice):
Order ID: 102, Customer: Satya, Total: ₹1500.0
Order ID: 104, Customer: Chandra, Total: ₹2800.0
Order ID: 101, Customer: Shashank, Total: ₹3200.0
Order ID: 103, Customer: Gokul, Total: ₹5000.0

🔹 After Quick Sort (by totalPrice):
Order ID: 102, Customer: Satya, Total: ₹1500.0
Order ID: 104, Customer: Chandra, Total: ₹2800.0
Order ID: 101, Customer: Shashank, Total: ₹3200.0
Order ID: 103, Customer: Gokul, Total: ₹5000.0
```

TASK 4 : (Employee Management System) :

Employee.java:

```
package ems;

public class Employee {
    int employeeId;
    String name;
    String position;
    double salary;
    public Employee(int employeeId, String name, String position, double salary) {
        this.employeeId = employeeId;
        this.name = name;
        this.position = position;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: ₹" +
salary;
    }
}
```

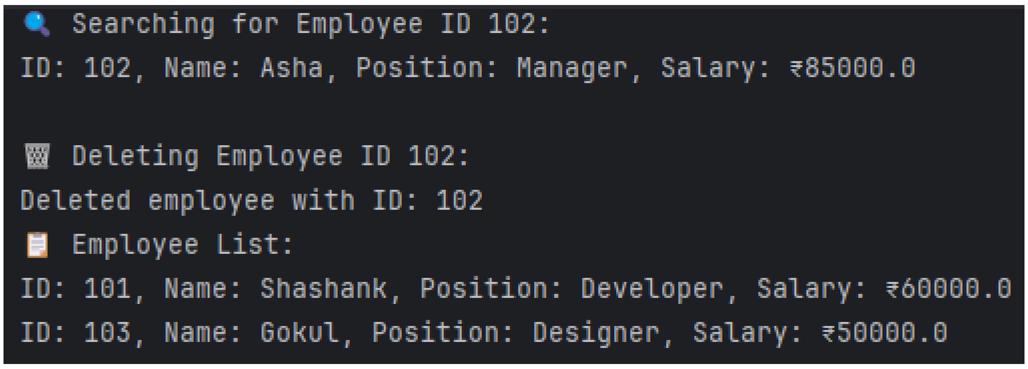
EmployeeManager.java:

```
package ems;

public class EmployeeManager {
    private Employee[] employees;
    private int count;

    public EmployeeManager(int size) {
        employees = new Employee[size];
        count = 0;
    }

    // Add Employee
    public void addEmployee(Employee emp) {
        if (count < employees.length) {
            employees[count++] = emp;
            System.out.println("Added: " + emp);
        } else {
            System.out.println("Employee list full!");
        }
    }

    public Employee searchEmployee(int id) {
        for (int i = 0; i < count; i++) {
            if (employees[i].employeeId == id) {
                return employees[i];
            }
        }
        return null;
    }
    public void listEmployees() {
        System.out.println("📋 Employee List:");
        for (int i = 0; i < count; i++) {
            System.out.println(employees[i]);
        }
    }


    public void deleteEmployee(int id) {
        for (int i = 0; i < count; i++) {
            if (employees[i].employeeId == id) {

                for (int j = i; j < count - 1; j++) {
                    employees[j] = employees[j + 1];
                }
                employees[--count] = null;
                System.out.println("Deleted employee with ID: " + id);
                return;
            }
        }
        System.out.println("Employee not found.");
    }
}
```

EmployeeTest.java:

```
package ems;

public class EmployeeTest {
    public static void main(String[] args) {
        EmployeeManager manager = new EmployeeManager(5);

        manager.addEmployee(new Employee(101, "Shashank", "Developer", 60000));
        manager.addEmployee(new Employee(102, "Asha", "Manager", 85000));
        manager.addEmployee(new Employee(103, "Gokul", "Designer", 50000));

        manager.listEmployees();

        System.out.println("\n🔍 Searching for Employee ID 102:");
        Employee found = manager.searchEmployee(102);
        System.out.println(found != null ? found : "Not found");

        System.out.println("\n🗑️ Deleting Employee ID 102:");
        manager.deleteEmployee(102);

        manager.listEmployees();
    }
}
```

OUTPUT :



```
🔍 Searching for Employee ID 102:
ID: 102, Name: Asha, Position: Manager, Salary: ₹85000.0

🗑 Deleting Employee ID 102:
Deleted employee with ID: 102
📋 Employee List:
ID: 101, Name: Shashank, Position: Developer, Salary: ₹60000.0
ID: 103, Name: Gokul, Position: Designer, Salary: ₹50000.0
```

TASK 4 : (Task Management System):

Task.java:

```java
package tasks;

public class Task {
    int taskId;
    String taskName;
    String status;

    public Task(int taskId, String taskName, String status) {
        this.taskId = taskId;
        this.taskName = taskName;
        this.status = status;
    }

    @Override
    public String toString() {
        return "Task ID: " + taskId + ", Name: " + taskName + ", Status: " + status;
    }
}
```

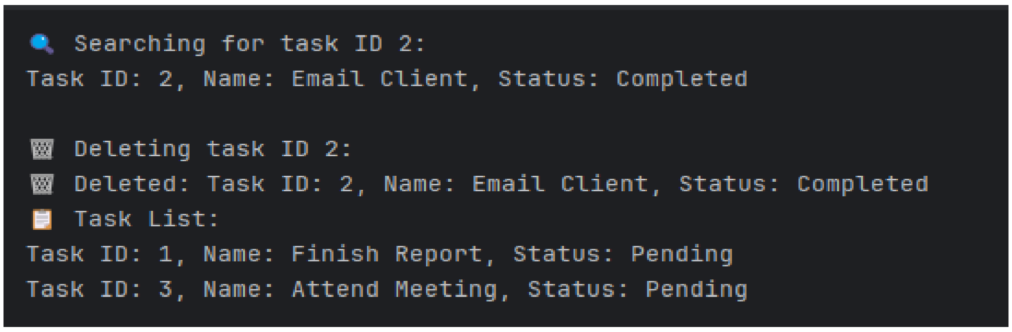
TaskManager.java:

```java
package tasks;

public class Task {
    int taskId;
    String taskName;
    String status;
    public Task(int taskId, String taskName, String status) {
        this.taskId = taskId;
        this.taskName = taskName;
        this.status = status;
    }

    @Override
    public String toString() {
        return "Task ID: " + taskId + ", Name: " + taskName + ", Status: " + status;
    }
}
```

TaskTest.java:

```java
package tasks;

public class TaskTest {
    public static void main(String[] args) {
        TaskManager manager = new TaskManager();

        manager.addTask(new Task(1, "Finish Report", "Pending"));
        manager.addTask(new Task(2, "Email Client", "Completed"));
        manager.addTask(new Task(3, "Attend Meeting", "Pending"));

        manager.listTasks();

        System.out.println("\n🔍 Searching for task ID 2:");
        Task found = manager.searchTask(2);
        System.out.println(found != null ? found : "Task not found");

        System.out.println("\n🗑️ Deleting task ID 2:");
        manager.deleteTask(2);

        manager.listTasks();
    }
}
```

OUTPUT :



```
🔍 Searching for task ID 2:
Task ID: 2, Name: Email Client, Status: Completed

🗑️ Deleting task ID 2:
🗑️ Deleted: Task ID: 2, Name: Email Client, Status: Completed
📋 Task List:
Task ID: 1, Name: Finish Report, Status: Pending
Task ID: 3, Name: Attend Meeting, Status: Pending
```

TASK 6 : (Library Management System) :

Book.java:

```
package library;

public class Book {
  int bookId;
  String title;
  String author;

  public Book(int bookId, String title, String author) {
    this.bookId = bookId;
    this.title = title;
    this.author = author;
  }

  @Override
  public String toString() {
    return "Book ID: " + bookId + ", Title: \"" + title + "\", Author: " + author;
  }
}
```

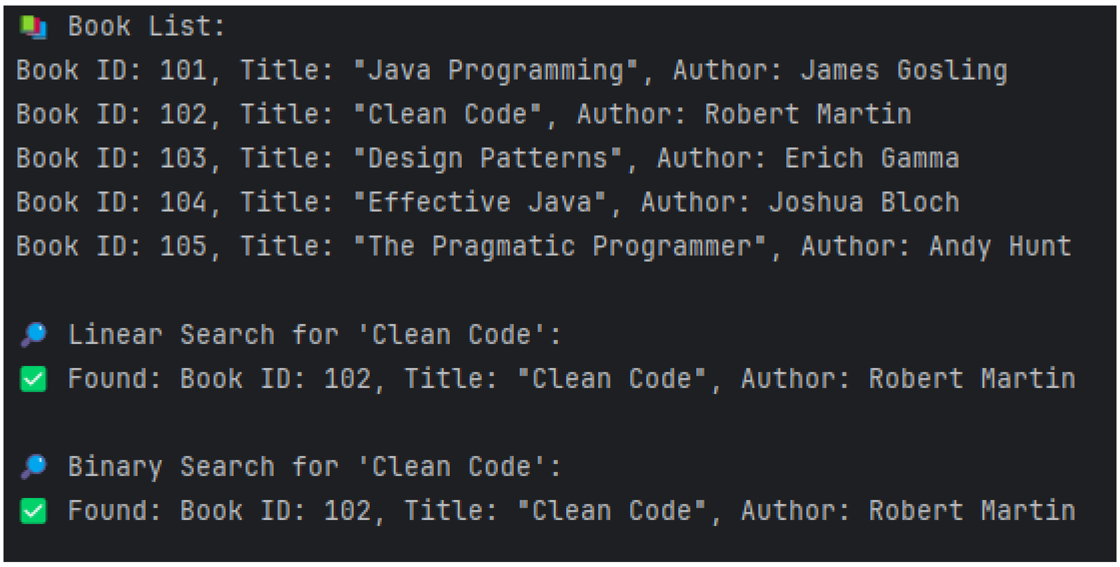
BookSearch.java:

```
package library;

public class Book {
  int bookId;
  String title;
  String author;

  public Book(int bookId, String title, String author) {
    this.bookId = bookId;
    this.title = title;
    this.author = author;
  }

  @Override
  public String toString() {
    return "Book ID: " + bookId + ", Title: \"" + title + "\", Author: " + author;
  }
}
```

LibraryTest.java:

```java
package library;

public class Book {
    int bookId;
    String title;
    String author;

    public Book(int bookId, String title, String author) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
    }

    @Override
    public String toString() {
        return "Book ID: " + bookId + ", Title: \"" + title + "\", Author: " + author;
    }
}
```

OUTPUT :



```
📚 Book List:
Book ID: 101, Title: "Java Programming", Author: James Gosling
Book ID: 102, Title: "Clean Code", Author: Robert Martin
Book ID: 103, Title: "Design Patterns", Author: Erich Gamma
Book ID: 104, Title: "Effective Java", Author: Joshua Bloch
Book ID: 105, Title: "The Pragmatic Programmer", Author: Andy Hunt

🔎 Linear Search for 'Clean Code':
✅ Found: Book ID: 102, Title: "Clean Code", Author: Robert Martin

🔎 Binary Search for 'Clean Code':
✅ Found: Book ID: 102, Title: "Clean Code", Author: Robert Martin
```

TASK 7 : (Financial Forecasting) :

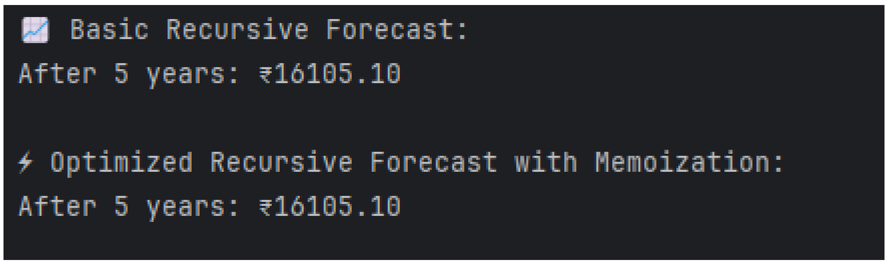
ForecastCalculator.java:

```java
package forecast;

public class ForecastCalculator {


    public static double forecastRecursive(double amount, double rate, int years) {
        if (years == 0) {
            return amount;
    }
    return forecastRecursive(amount * (1 + rate), rate, years - 1);
  }


  public static double forecastMemo(double amount, double rate, int years, double[] memo) {
    if (years == 0) return amount;

    if (memo[years] != 0) {
      return memo[years];
    }

    memo[years] = forecastMemo(amount, rate, years - 1, memo) * (1 + rate);
    return memo[years];
  }
}
```

ForecastTest.java:

```
package forecast;

public class ForecastTest {
  public static void main(String[] args) {
    double initialAmount = 10000;
    double growthRate = 0.10;  // 10% growth
    int years = 5;

    System.out.println("📈 Basic Recursive Forecast:");
    double futureValue1 = ForecastCalculator.forecastRecursive(initialAmount, growthRate,
years);
    System.out.printf("After %d years: ₹%.2f\n", years, futureValue1);

    System.out.println("\n ⚡ Optimized Recursive Forecast with Memoization:");
    double[] memo = new double[years + 1];
    double futureValue2 = ForecastCalculator.forecastMemo(initialAmount, growthRate, years,
memo);
    System.out.printf("After %d years: ₹%.2f\n", years, futureValue2);
  }
}
```

OUTPUT :



```
📈 Basic Recursive Forecast:
After 5 years: ₹16105.10

⚡ Optimized Recursive Forecast with Memoization:
After 5 years: ₹16105.10
```