**WEEK-1(HANDS ON-1)**

**DESIGN PATTERNS AND PRINCIPLES**

**Exercise 1**: Implementing the Singleton Pattern

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Source Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

**A computer screen with white text

AI-generated content may be incorrect.**

**Exercise 2: Implementing the Factory Method Pattern**

**Scenario:**

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

**Source code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Exercise 3: Implementing the Builder Pattern**

**Scenario**: You are developing a system to create complex objects such as a Computer with multiple optional parts. Use the Builder Pattern to manage the construction process**.**

**Source code:**

A screen shot of a computer program

AI-generated content may be incorrect.

**A screenshot of a computer program

AI-generated content may be incorrect.**

A computer screen shot of a program code

AI-generated content may be incorrect.

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

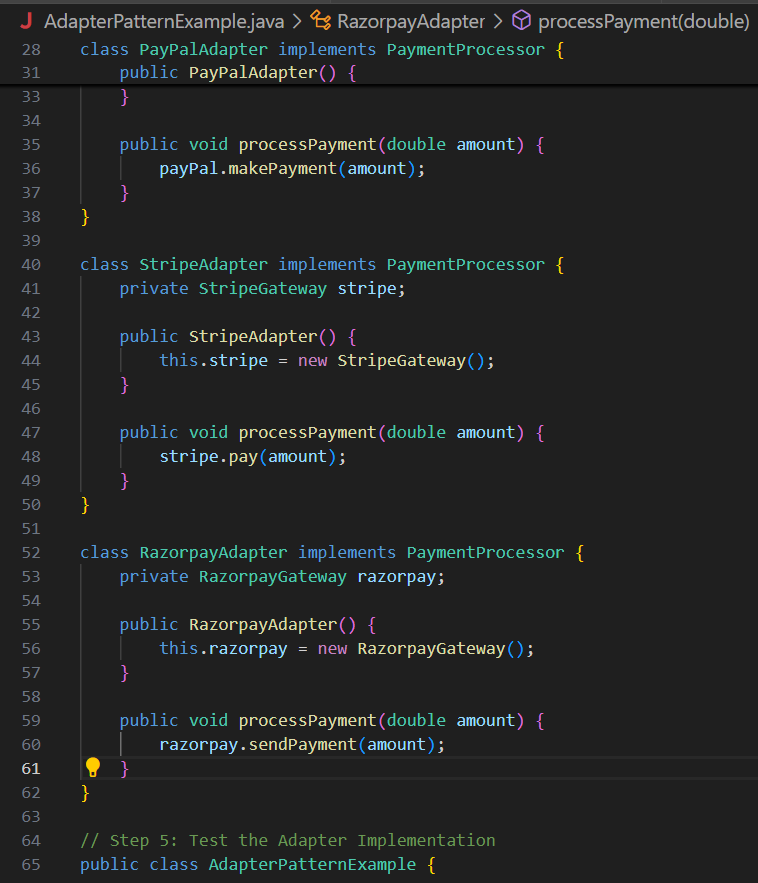
**Exercise 4: Implementing the Adapter Pattern**

**Scenario:** You are developing a payment processing system that needs to integrate with multiple third-party payment gateways with different interfaces. Use the Adapter Pattern to achieve this.

**Source code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**



A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

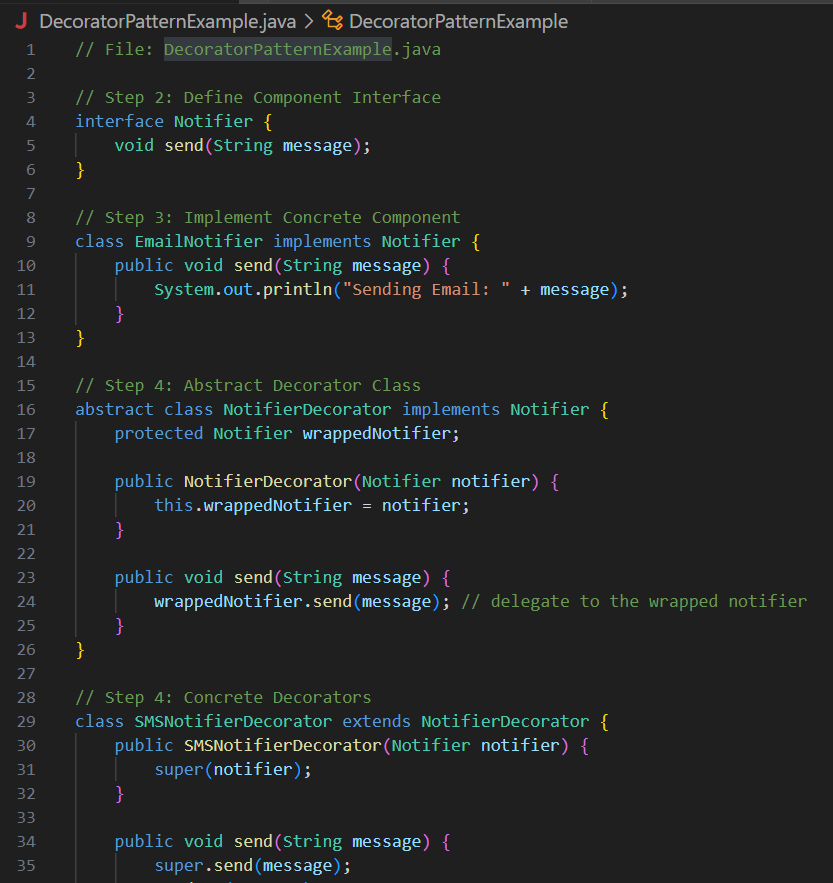
**A screen shot of a computer program

AI-generated content may be incorrect.**

**Exercise 5: Implementing the Decorator Pattern**

**Scenario:** You are developing a notification system where notifications can be sent via multiple channels (e.g., Email, SMS). Use the Decorator Pattern to add functionalities dynamically.

**Source code:**

****

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

**Output**:

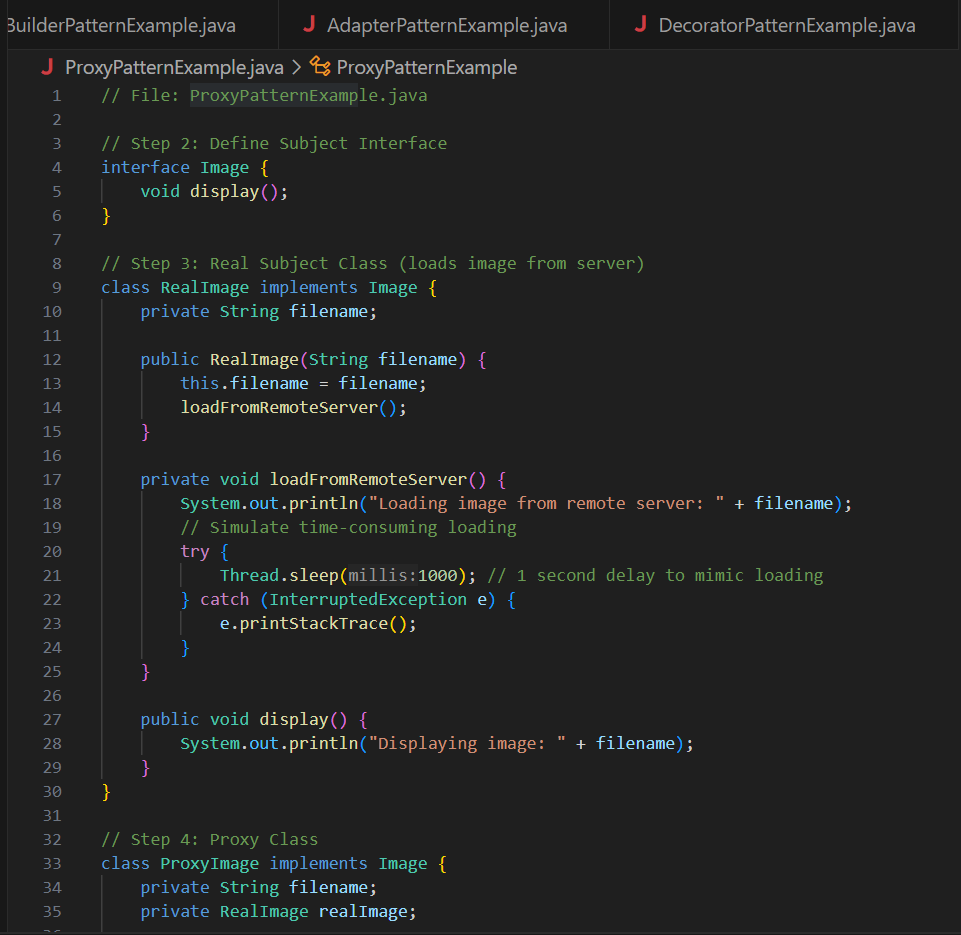
A screen shot of a computer

AI-generated content may be incorrect.

**Exercise 6: Implementing the Proxy Pattern**

**Scenario:** You are developing an image viewer application that loads images from a remote server. Use the Proxy Pattern to add lazy initialization and caching.

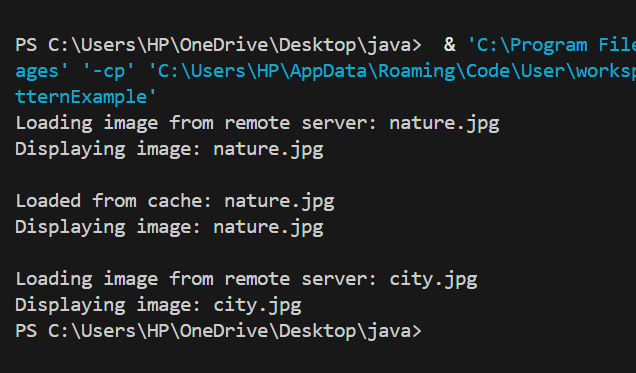
**Source code:**

****

A screen shot of a computer program

AI-generated content may be incorrect.

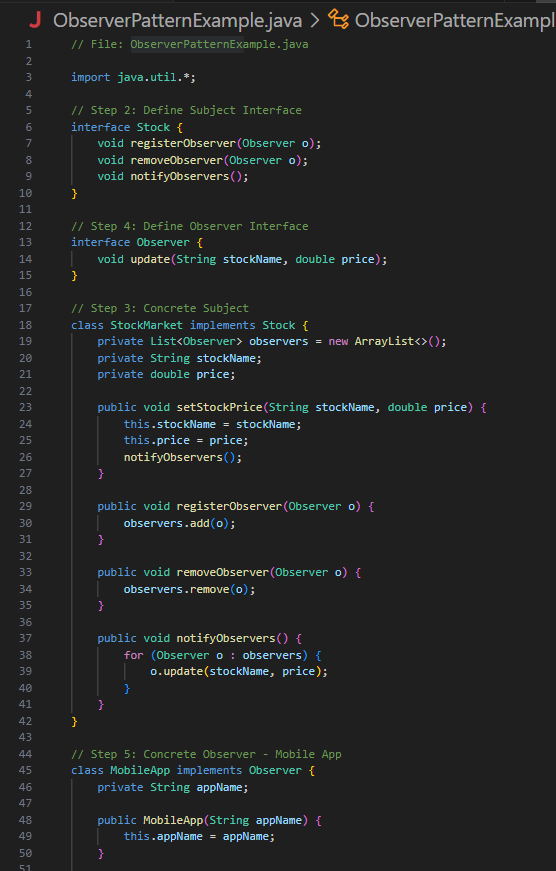
**Output:**



**Exercise 7: Implementing the Observer Pattern**

**Scenario:** You are developing a stock market monitoring application where multiple clients need to be notified whenever stock prices change. Use the Observer Pattern to achieve this.

**Source code:**

****

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

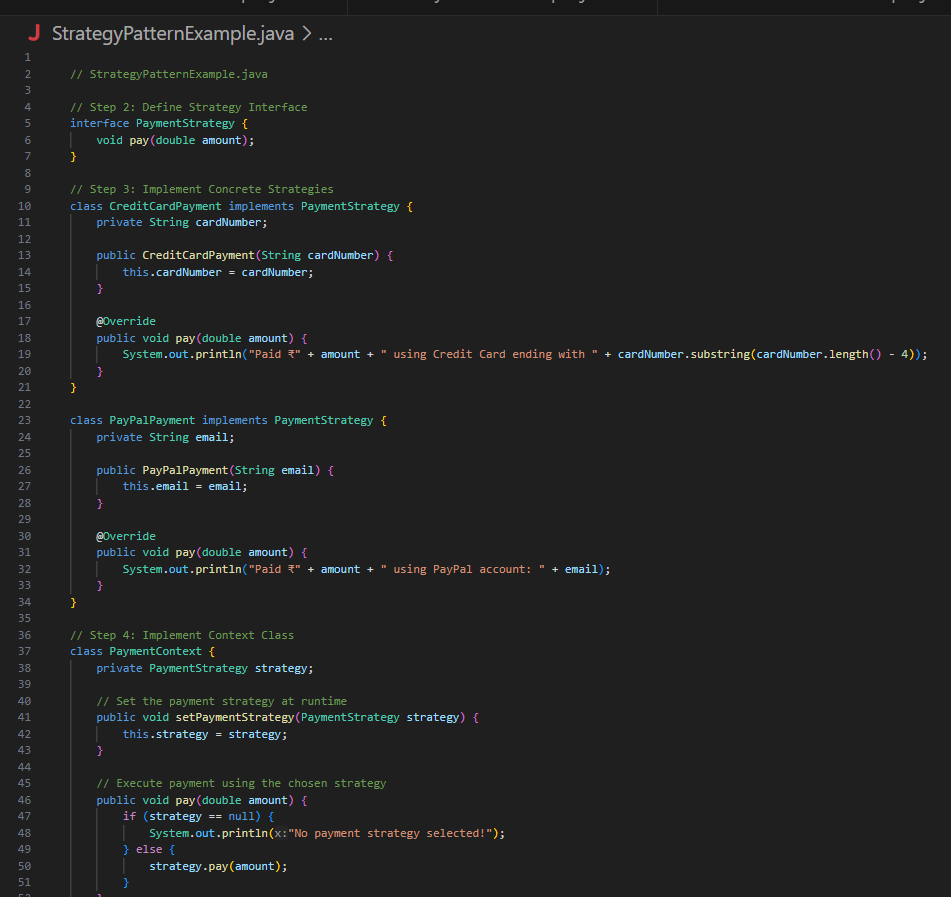
A screen shot of a computer program

AI-generated content may be incorrect.

**Exercise 8: Implementing the Strategy Pattern**

**Scenario:** You are developing a payment system where different payment methods (e.g., Credit Card, PayPal) can be selected at runtime. Use the Strategy Pattern to achieve this.

**Source code:**

****

A computer screen shot of code

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer screen

AI-generated content may be incorrect.

**Exercise 9: Implementing the Command Pattern**

**Scenario:** You are developing a home automation system where commands can be issued to turn devices on or off. Use the Command Pattern to achieve this.

**Source code:**

****

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

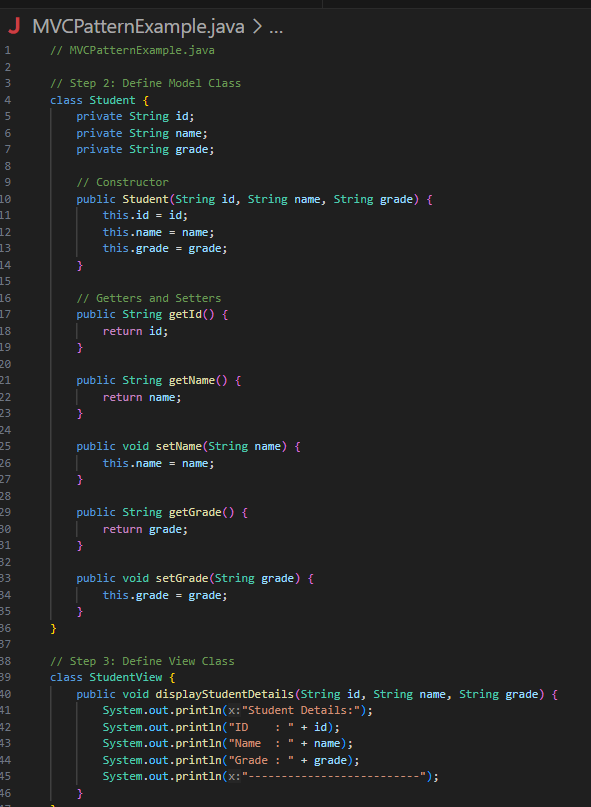
**A black screen with white text

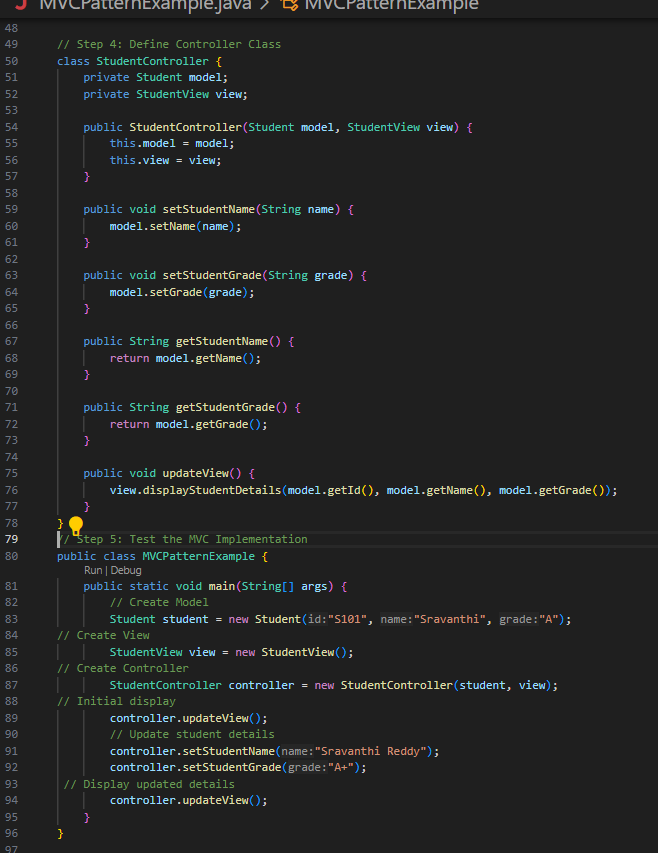
AI-generated content may be incorrect.**

**Exercise 10: Implementing the MVC Pattern**

**Scenario:** You are developing a simple web application for managing student records using the MVC pattern.

**Source code:**

****



**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Exercise 11: Implementing Dependency Injection**

**Scenario:** You are developing a customer management application where the service class depends on a repository class. Use Dependency Injection to manage these dependencies.

**Source code:**

****

**Output**:

A black screen with white text

AI-generated content may be incorrect.

**ALGORITHMS\_DATA STRUCTURES**

**Exercise 1: Inventory Management System**

**Scenario:**

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

**Source code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 2: E-commerce Platform Search Function**

**Scenario:** You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Source code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

oA screen shot of a computer program

AI-generated content may be incorrect.

**output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Exercise 3: Sorting Customer Orders**

**Scenario:** You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**Source code:**

****

A screen shot of a computer program

AI-generated content may be incorrect.

**Output**:

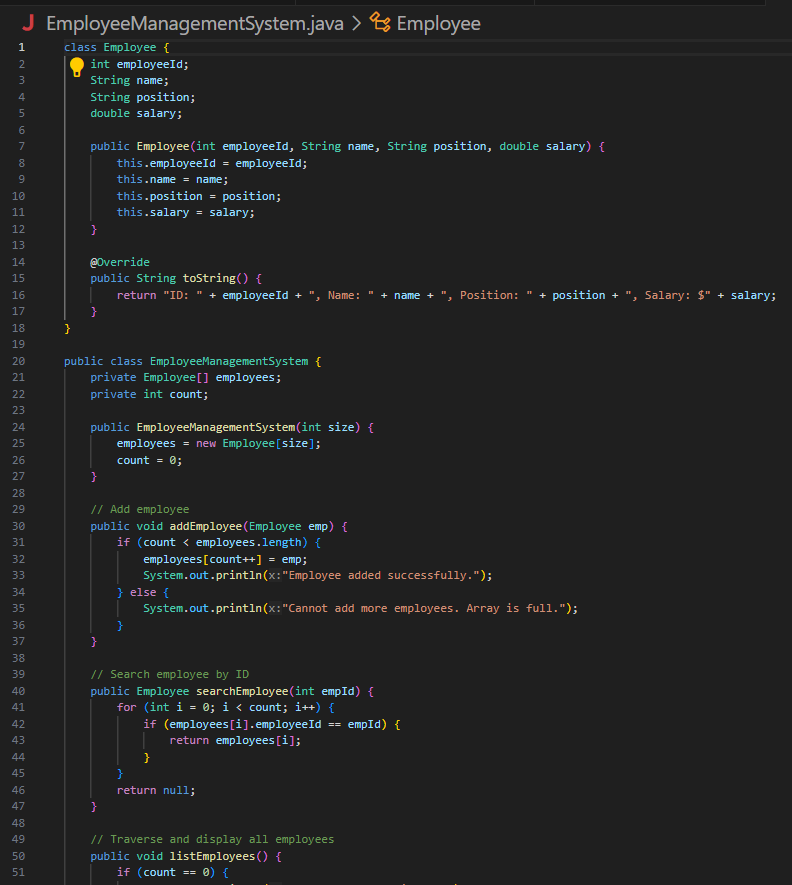
A screenshot of a computer screen

AI-generated content may be incorrect.

**Exercise 4: Employee Management System**

**Scenario:** You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**Source code:**

****

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

**A screenshot of a computer

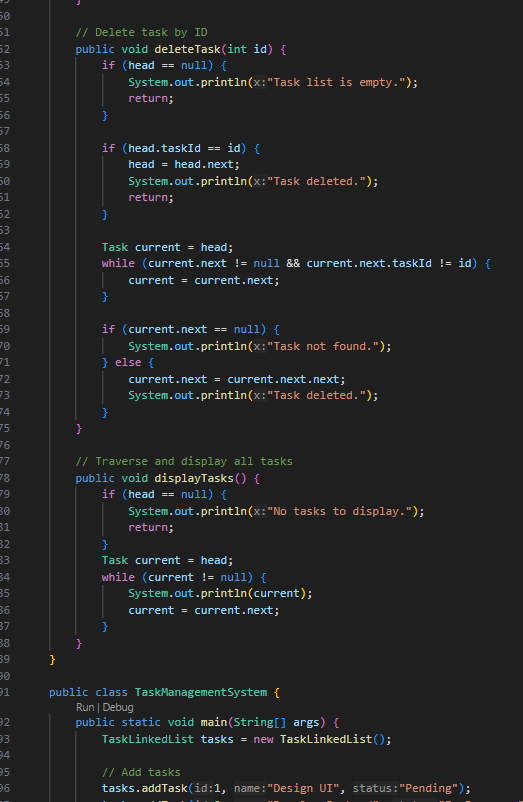
AI-generated content may be incorrect.**

**Exercise 5: Task Management System**

**Scenario:** You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

A screen shot of a computer program

AI-generated content may be incorrect.



A computer screen shot of a program

AI-generated content may be incorrect.

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Exercise 6: Library Management System**

**Scenario:** You are developing a library management system where users can search for books by title or author.

**Source code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

**Output:**

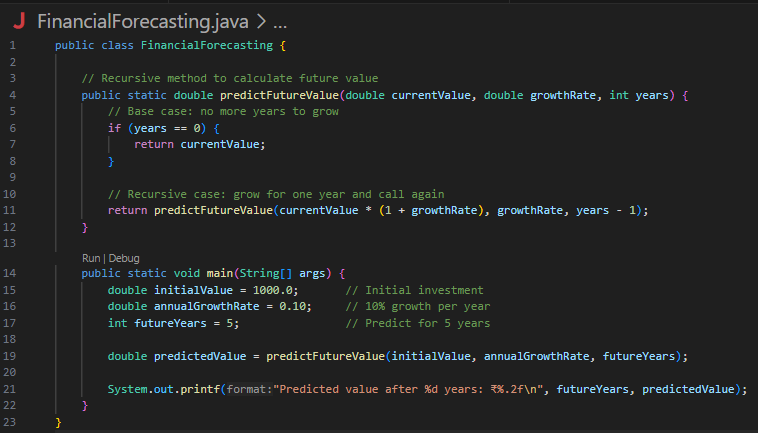
A black screen with white text

AI-generated content may be incorrect.

**Exercise 7: Financial Forecasting**

**Scenario:** You are developing a financial forecasting tool that predicts future values based on past data.

**Source code:**

****

**Output**:

A black background with white text

AI-generated content may be incorrect.