# Implementing the Singleton Pattern

public class Singleton {  
 private Singleton() {  
 System.*out*.println("Singleton instance created");  
 }  
 private static class SingletonHelper {  
 private static final Singleton *INSTANCE* = new Singleton();  
 }  
 public static Singleton getInstance() {  
 return SingletonHelper.*INSTANCE*;  
 }  
}  
  
  
class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Calling getInstance first time:");  
 Singleton s1 = Singleton.*getInstance*();  
  
 System.*out*.println("Calling getInstance second time:");  
 Singleton s2 = Singleton.*getInstance*();  
  
 System.*out*.println("Are both instances same? " + (s1 == s2));  
 }  
}



# Implementing the Factory Method Pattern

package Implementing\_the\_Factory\_Method\_Pattern;  
  
*// Product Interface*interface Food {  
 String prepare();  
}  
  
*// Concrete Products*class Pizza implements Food {  
 public String prepare() {  
 return "Preparing Pizza 🍕";  
 }  
}  
  
class Burger implements Food {  
 public String prepare() {  
 return "Preparing Burger 🍔";  
 }  
}  
  
*// Factory class*abstract class FoodFactory {  
 abstract Food createFood();  
}  
  
*// Concrete Factories*class PizzaFactory extends FoodFactory {  
 public Food createFood() {  
 return new Pizza();  
 }  
}  
  
class BurgerFactory extends FoodFactory {  
 public Food createFood() {  
 return new Burger();  
 }  
}  
  
*// Client Code*public class FoodApp {  
 public static void main(String[] args) {  
 FoodFactory factory = new PizzaFactory(); *// can swap to BurgerFactory, etc.* Food food = factory.createFood();  
 System.*out*.println(food.prepare());  
  
 factory = new BurgerFactory();  
 Food food2 = factory.createFood();  
 System.*out*.println(food2.prepare());  
 }  
}

A screenshot of a computer

AI-generated content may be incorrect.

# E-commerce Platform Search Function

class Product {

int id;

String name;

Product(int id, String name) {

this.id = id;

this.name = name;

}

}

public class EcommerceSearch {

public static int binarySearch(Product[] products, int targetId) {

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (products[mid].id == targetId)

return mid;

else if (products[mid].id < targetId)

left = mid + 1;

else

right = mid - 1;

}

return -1;

}

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop"),

new Product(105, "Mouse"),

new Product(110, "Keyboard"),

new Product(120, "Monitor"),

new Product(130, "Printer")

};

int index = binarySearch(products, 110);

if (index != -1)

System.out.println("Product found: " + products[index].name);

else

System.out.println("Product not found.");

}

}

A screenshot of a computer

AI-generated content may be incorrect.

# Financial Forecasting

public class FinancialForecast {

public static int[] forecastRevenue(int[] dailyRevenue, int days) {

int n = dailyRevenue.length;

int[] prefixSum = new int[n];

prefixSum[0] = dailyRevenue[0];

for (int i = 1; i < n; i++) {

prefixSum[i] = prefixSum[i - 1] + dailyRevenue[i];

}

int[] forecast = new int[n - days + 1];

for (int i = 0; i <= n - days; i++) {

int sum = prefixSum[i + days - 1];

if (i > 0) sum -= prefixSum[i - 1];

forecast[i] = sum;

}

return forecast;

}

public static void main(String[] args) {

int[] dailyRevenue = {100, 120, 130, 90, 150, 170, 160};

int days = 3;

int[] result = forecastRevenue(dailyRevenue, days);

for (int r : result) {

System.out.print(r + " ");

}

}

}

