**Exercise 7: Financial Forecasting**

**Code:**

import java.util.\*;

public class FinancialForecastingApp {

static class FinancialData {

private final List<Double> historicalData;

public FinancialData(List<Double> historicalData) {

this.historicalData = new ArrayList<>(historicalData);

}

public List<Double> getHistoricalData() {

return Collections.unmodifiableList(historicalData);

}

}

static class Forecaster {

public double calculateSimpleMovingAverage(FinancialData data, int period) {

List<Double> historical = data.getHistoricalData();

if (historical.size() < period || period <= 0) {

throw new IllegalArgumentException("Insufficient data or invalid period for SMA calculation.");

}

double sum = 0;

for (int i = historical.size() - period; i < historical.size(); i++) {

sum += historical.get(i);

}

return sum / period;

}

public double forecastNextPeriodSMA(FinancialData data, int period) {

List<Double> historical = data.getHistoricalData();

if (historical.size() < period || period <= 0) {

throw new IllegalArgumentException("Insufficient historical data for forecasting.");

}

return calculateSimpleMovingAverage(data, period);

}

}

public static void main(String[] args) {

List<Double> stockPrices = Arrays.asList(100.0, 102.5, 101.0, 103.5, 105.0, 104.0, 106.5, 107.0, 109.0, 110.5);

FinancialData marketData = new FinancialData(stockPrices);

Forecaster forecaster = new Forecaster();

System.out.println("--- Financial Forecasting using Simple Moving Average ---");

System.out.println("Historical Data: " + marketData.getHistoricalData());

int smaPeriod = 3;

try {

double forecast3DaySMA = forecaster.forecastNextPeriodSMA(marketData, smaPeriod);

System.out.printf("3-Day SMA for next period: %.2f%n", forecast3DaySMA);

} catch (IllegalArgumentException e) {

System.out.println("Error: " + e.getMessage());

}

smaPeriod = 5;

try {

double forecast5DaySMA = forecaster.forecastNextPeriodSMA(marketData, smaPeriod);

System.out.printf("5-Day SMA for next period: %.2f%n", forecast5DaySMA);

} catch (IllegalArgumentException e) {

System.out.println("Error: " + e.getMessage());

}

List<Double> revenues = Arrays.asList(50000.0, 52000.0, 51500.0, 53000.0, 54000.0, 56000.0, 55500.0);

FinancialData companyRevenue = new FinancialData(revenues);

System.out.println("\n--- Revenue Forecasting ---");

System.out.println("Historical Revenue: " + companyRevenue.getHistoricalData());

smaPeriod = 4;

try {

double forecastRevenue4Period = forecaster.forecastNextPeriodSMA(companyRevenue, smaPeriod);

System.out.printf("4-Period SMA Revenue forecast: %.2f%n", forecastRevenue4Period);

} catch (IllegalArgumentException e) {

System.out.println("Error: " + e.getMessage());

}

List<Double> shortData = Arrays.asList(10.0, 20.0);

FinancialData shortMarketData = new FinancialData(shortData);

System.out.println("\n--- Forecasting with insufficient data ---");

System.out.println("Historical Data: " + shortMarketData.getHistoricalData());

try {

forecaster.forecastNextPeriodSMA(shortMarketData, 3);

} catch (IllegalArgumentException e) {

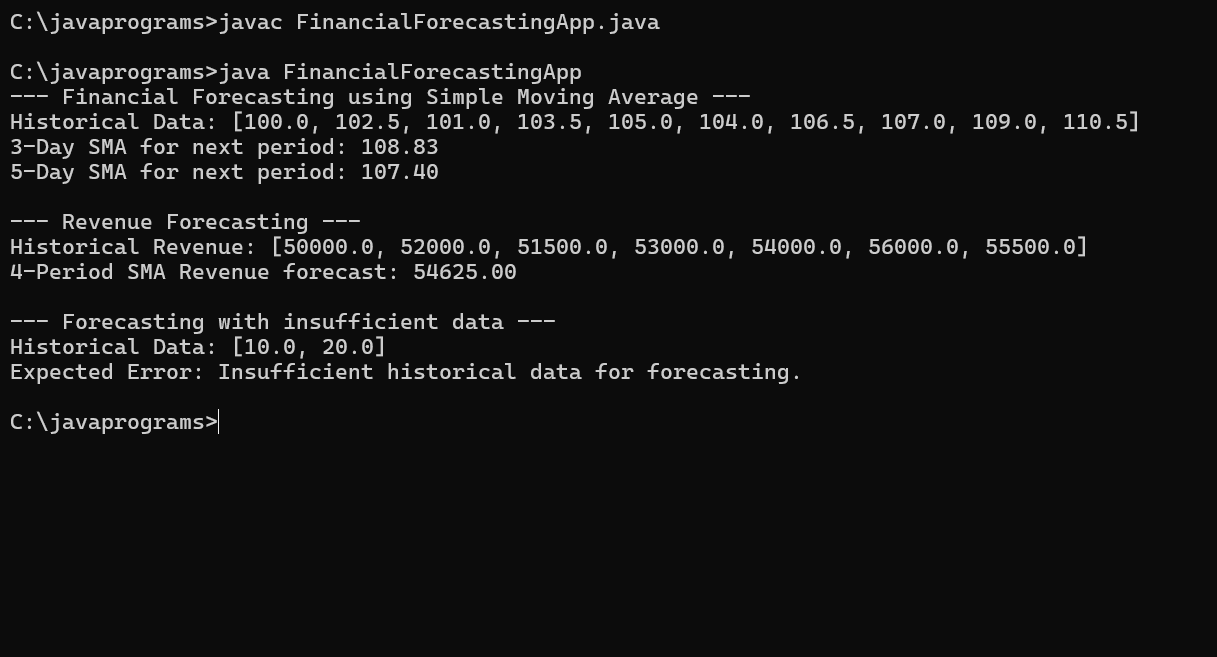
System.out.println("Expected Error: " + e.getMessage());

}

}

}

**Output:**

****