package org.example;

import javax.swing.\*;

import java.awt.BorderLayout;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.io.\*;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Random;

class Stock {

String symbol;

double price;

public Stock(String symbol, double price) {

this.symbol = symbol;

this.price = price;

}

public void updatePrice() {

double change = (Math.random() - 0.5) \* 5; // random change ±2.5

price = Math.max(1, price + change);

}

}

class User {

String name;

double balance;

Map<String, Integer> portfolio = new LinkedHashMap<>();

public User(String name, double balance) {

this.name = name;

this.balance = balance;

}

public void buyStock(Stock stock, int quantity) {

double totalCost = stock.price \* quantity;

if (balance >= totalCost) {

balance -= totalCost;

portfolio.put(stock.symbol, portfolio.getOrDefault(stock.symbol, 0) + quantity);

}

}

public void sellStock(Stock stock, int quantity) {

int owned = portfolio.getOrDefault(stock.symbol, 0);

if (owned >= quantity) {

balance += stock.price \* quantity;

portfolio.put(stock.symbol, owned - quantity);

}

}

public double getPortfolioValue(Map<String, Stock> market) {

double value = 0;

for (Map.Entry<String, Integer> entry : portfolio.entrySet()) {

Stock stock = market.get(entry.getKey());

if (stock != null) {

value += stock.price \* entry.getValue();

}

}

return value;

}

}

class Market {

Map<String, Stock> stocks = new LinkedHashMap<>();

Random random = new Random();

public void addStock(String symbol, double price) {

stocks.put(symbol, new Stock(symbol, price));

}

public void step() {

for (Stock stock : stocks.values()) {

stock.updatePrice();

}

}

}

public class StockTradingApp extends JFrame {

private final User user;

private final Market market;

private final JTextArea displayArea;

private final JComboBox<String> stockSelector;

private final JTextField quantityField;

private final javax.swing.Timer marketTimer;

private final java.util.List<Double> portfolioHistory;

private final DecimalFormat df = new DecimalFormat("#.##");

public StockTradingApp() {

user = new User("Zainab", 10000);

market = new Market();

portfolioHistory = new ArrayList<>();

// Initial market data

market.addStock("AAPL", 150);

market.addStock("GOOG", 2800);

market.addStock("TSLA", 700);

market.addStock("MSFT", 300);

setTitle("Stock Trading Platform");

setSize(700, 500);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

JPanel topPanel = new JPanel(new FlowLayout());

stockSelector = new JComboBox<>(market.stocks.keySet().toArray(new String[0]));

quantityField = new JTextField(5);

JButton buyButton = new JButton("Buy");

JButton sellButton = new JButton("Sell");

JButton saveButton = new JButton("Save Portfolio");

JButton loadButton = new JButton("Load Portfolio");

topPanel.add(new JLabel("Stock:"));

topPanel.add(stockSelector);

topPanel.add(new JLabel("Quantity:"));

topPanel.add(quantityField);

topPanel.add(buyButton);

topPanel.add(sellButton);

topPanel.add(saveButton);

topPanel.add(loadButton);

add(topPanel, BorderLayout.NORTH);

displayArea = new JTextArea();

displayArea.setEditable(false);

add(new JScrollPane(displayArea), BorderLayout.CENTER);

// Button Actions

buyButton.addActionListener(this::handleBuy);

sellButton.addActionListener(this::handleSell);

saveButton.addActionListener(e -> savePortfolio());

loadButton.addActionListener(e -> loadPortfolio());

// Market auto-update

marketTimer = new javax.swing.Timer(2000, e -> {

market.step();

recordPortfolioValue();

updateDisplay();

});

marketTimer.start();

recordPortfolioValue();

updateDisplay();

setVisible(true);

}

private void handleBuy(ActionEvent e) {

try {

String symbol = (String) stockSelector.getSelectedItem();

int qty = Integer.parseInt(quantityField.getText());

user.buyStock(market.stocks.get(symbol), qty);

updateDisplay();

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Enter a valid quantity");

}

}

private void handleSell(ActionEvent e) {

try {

String symbol = (String) stockSelector.getSelectedItem();

int qty = Integer.parseInt(quantityField.getText());

user.sellStock(market.stocks.get(symbol), qty);

updateDisplay();

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Enter a valid quantity");

}

}

private void updateDisplay() {

StringBuilder sb = new StringBuilder();

sb.append("User: ").append(user.name).append("\n");

sb.append("Balance: $").append(df.format(user.balance)).append("\n");

sb.append("Portfolio Value: $").append(df.format(user.getPortfolioValue(market.stocks))).append("\n");

sb.append("Total Value: $").append(df.format(user.balance + user.getPortfolioValue(market.stocks))).append("\n\n");

sb.append("Portfolio:\n");

for (Map.Entry<String, Integer> entry : user.portfolio.entrySet()) {

if (entry.getValue() > 0) {

sb.append(entry.getKey()).append(": ").append(entry.getValue()).append(" shares\n");

}

}

sb.append("\nMarket Prices:\n");

for (Stock stock : market.stocks.values()) {

sb.append(stock.symbol).append(" - $").append(df.format(stock.price)).append("\n");

}

sb.append("\nPortfolio History:\n");

for (int i = 0; i < portfolioHistory.size(); i++) {

sb.append("Step ").append(i + 1).append(": $").append(df.format(portfolioHistory.get(i))).append("\n");

}

displayArea.setText(sb.toString());

}

private void recordPortfolioValue() {

portfolioHistory.add(user.getPortfolioValue(market.stocks) + user.balance);

}

private void savePortfolio() {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("portfolio.dat"))) {

oos.writeObject(user.portfolio);

oos.writeDouble(user.balance);

JOptionPane.showMessageDialog(this, "Portfolio saved!");

} catch (IOException ex) {

JOptionPane.showMessageDialog(this, "Error saving portfolio: " + ex.getMessage());

}

}

@SuppressWarnings("unchecked")

private void loadPortfolio() {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("portfolio.dat"))) {

user.portfolio = (Map<String, Integer>) ois.readObject();

user.balance = ois.readDouble();

JOptionPane.showMessageDialog(this, "Portfolio loaded!");

updateDisplay();

} catch (IOException | ClassNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Error loading portfolio: " + ex.getMessage());

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(StockTradingApp::new);

}

}