import java.io.\*;

import java.util.\*;

class Stock {

String symbol;

double price;

public Stock(String symbol, double price) {

this.symbol = symbol;

this.price = price;

}

public void updatePrice(double newPrice) {

this.price = newPrice;

}

}

class User {

String name;

double balance;

HashMap<String, Integer> portfolio;

public User(String name, double balance) {

this.name = name;

this.balance = balance;

this.portfolio = new HashMap<>();

}

public void buyStock(String symbol, int quantity, double price) {

double cost = quantity \* price;

if (balance >= cost) {

balance -= cost;

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

System.out.println("Bought " + quantity + " shares of " + symbol);

} else {

System.out.println("Insufficient funds.");

}

}

public void sellStock(String symbol, int quantity, double price) {

if (portfolio.containsKey(symbol) && portfolio.get(symbol) >= quantity) {

balance += quantity \* price;

portfolio.put(symbol, portfolio.get(symbol) - quantity);

if (portfolio.get(symbol) == 0) portfolio.remove(symbol);

System.out.println("Sold " + quantity + " shares of " + symbol);

} else {

System.out.println("Not enough shares to sell.");

}

}

public void showPortfolio(Map<String, Stock> marketStocks) {

System.out.println("\n📊 Portfolio for " + name + ":");

double totalValue = 0;

for (String sym : portfolio.keySet()) {

int qty = portfolio.get(sym);

double price = marketStocks.get(sym).price;

double value = qty \* price;

System.out.println(sym + " - " + qty + " shares @ $" + price + " = $" + value);

totalValue += value;

}

System.out.printf("💵 Balance: $%.2f | 📈 Total Value: $%.2f\n", balance, balance + totalValue);

}

public String serialize() {

return name + "," + balance + "," + portfolio.toString().replace("=", ":");

}

public static User deserialize(String line) {

String[] parts = line.split(",", 3);

User user = new User(parts[0], Double.parseDouble(parts[1]));

String portfolioStr = parts[2].replace("{", "").replace("}", "").trim();

if (!portfolioStr.isEmpty()) {

String[] entries = portfolioStr.split(", ");

for (String entry : entries) {

String[] kv = entry.split(":");

user.portfolio.put(kv[0], Integer.parseInt(kv[1]));

}

}

return user;

}

}

class Market {

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

public Market() {

stocks.put("AAPL", new Stock("AAPL", 150.00));

stocks.put("GOOG", new Stock("GOOG", 2700.00));

stocks.put("TSLA", new Stock("TSLA", 700.00));

stocks.put("AMZN", new Stock("AMZN", 3300.00));

}

public void showMarket() {

System.out.println("\n📈 Current Market Prices:");

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

System.out.println(s.symbol + ": $" + s.price);

}

}

public void simulatePriceChange() {

Random rand = new Random();

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

double change = rand.nextDouble() \* 10 - 5; // -5% to +5%

s.updatePrice(Math.round((s.price + s.price \* (change / 100)) \* 100.0) / 100.0);

}

}

}

public class TradingSystem {

static final String FILE\_NAME = "portfolio.txt";

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Market market = new Market();

User user = loadUser();

if (user == null) {

System.out.print("Enter your name: ");

String name = sc.nextLine();

user = new User(name, 10000.00); // default balance

}

while (true) {

System.out.println("\n--- STOCK TRADING SYSTEM ---");

System.out.println("1. View Market");

System.out.println("2. Buy Stock");

System.out.println("3. Sell Stock");

System.out.println("4. View Portfolio");

System.out.println("5. Simulate Market Change");

System.out.println("6. Save & Exit");

System.out.print("Choose option: ");

int choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

market.showMarket();

break;

case 2:

System.out.print("Enter stock symbol: ");

String buySymbol = sc.nextLine().toUpperCase();

if (market.stocks.containsKey(buySymbol)) {

System.out.print("Enter quantity: ");

int qty = sc.nextInt();

user.buyStock(buySymbol, qty, market.stocks.get(buySymbol).price);

} else {

System.out.println("Invalid symbol.");

}

break;

case 3:

System.out.print("Enter stock symbol: ");

String sellSymbol = sc.nextLine().toUpperCase();

if (market.stocks.containsKey(sellSymbol)) {

System.out.print("Enter quantity: ");

int qty = sc.nextInt();

user.sellStock(sellSymbol, qty, market.stocks.get(sellSymbol).price);

} else {

System.out.println("Invalid symbol.");

}

break;

case 4:

user.showPortfolio(market.stocks);

break;

case 5:

market.simulatePriceChange();

System.out.println("Market prices updated!");

break;

case 6:

saveUser(user);

System.out.println("Portfolio saved. Exiting...");

return;

default:

System.out.println("Invalid choice.");

}

}

}

public static void saveUser(User user) {

try (PrintWriter out = new PrintWriter(FILE\_NAME)) {

out.println(user.serialize());

} catch (IOException e) {

System.out.println("Error saving portfolio.");

}

}

public static User loadUser() {

try (BufferedReader br = new BufferedReader(new FileReader(FILE\_NAME))) {

return User.deserialize(br.readLine());

} catch (IOException e) {

return null;

}

}

}