# INVENTORY MANGEMENT SYSTEM:

# PREPARED BY:

1. ALI HASSAN (SP24-BSE-140)
2. HAMMAD ALEEM (SP24-BSE-073)
3. UMAR AMIN (SP24-BSE-055)

# CODE:

MAIN CLASS:

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

UserManager userManager = new UserManager();

Scanner scanner = new Scanner(System.in);

System.out.println("");

System.out.println("");

System.out.println("Welcome to the Inventory Management System!");

boolean loggedIn = false;

while (!loggedIn) {

System.out.println("\n1. Sign Up");

System.out.println("2. Log In");

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

int option = scanner.nextInt();

scanner.nextLine();

switch (option) {

case 1:

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

String username = scanner.nextLine();

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

String password = scanner.nextLine();

if (userManager.registerUser(username, password)) {

System.out.println("User registered successfully! You can now log in.");

}

break;

case 2:

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

username = scanner.nextLine();

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

password = scanner.nextLine();

if (userManager.loginUser(username, password)) {

System.out.println("Login successful!");

loggedIn = true;

} else {

System.out.println("Invalid credentials, please try again.");

}

break;

default:

System.out.println("Invalid option! Please choose either 1 or 2.");

}

}

Inventory inventory = new Inventory();

String filename = "products.txt";

inventory.loadFromFile(filename);

while (true) {

System.out.println("\nInventory Management System");

System.out.println("1. Add Product");

System.out.println("2. Search Product");

System.out.println("3. Remove Product");

System.out.println("4. List All Products");

System.out.println("5. Save Products to File");

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

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

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.println("Choose product type to add:");

System.out.println("1. Shirt");

System.out.println("2. Pants");

System.out.println("3. Shoes");

System.out.println("4. Jacket");

int type = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Product ID: ");

String productId = scanner.nextLine();

System.out.print("Enter Product Name: ");

String name = scanner.nextLine();

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

double price = scanner.nextDouble();

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

int quantity = scanner.nextInt();

scanner.nextLine();

switch (type) {

case 1:

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

String color = scanner.nextLine();

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

String brand = scanner.nextLine();

System.out.print("Enter Size (SMALL, MEDIUM, LARGE): ");

Size size = Size.valueOf(scanner.nextLine().toUpperCase());

inventory.addProduct(new Shirt(productId, name, price, quantity, color, brand, size));

break;

case 2:

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

String material = scanner.nextLine();

System.out.print("Enter Size (SMALL, MEDIUM, LARGE): ");

size = Size.valueOf(scanner.nextLine().toUpperCase());

inventory.addProduct(new Pants(productId, name, price, quantity, size, material));

break;

case 3:

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

brand = scanner.nextLine();

System.out.print("Enter Size (e.g., 42): ");

int shoeSize = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Type (e.g., Running, Casual): ");

String shoeType = scanner.nextLine();

inventory.addProduct(new Shoe(productId, name, price, quantity, brand, shoeSize, shoeType));

break;

case 4:

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

material = scanner.nextLine();

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

color = scanner.nextLine();

System.out.print("Enter Size (SMALL, MEDIUM, LARGE): ");

size = Size.valueOf(scanner.nextLine().toUpperCase());

inventory.addProduct(new Jacket(productId, name, price, quantity, material, color, size));

break;

default:

System.out.println("Invalid product type!");

}

break;

case 2:

System.out.print("Enter Product ID or Name to search: ");

String searchTerm = scanner.nextLine();

Product foundProduct = inventory.searchProduct(searchTerm);

if (foundProduct != null) {

System.out.println("Product Found: " + foundProduct);

} else {

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

}

break;

case 3:

System.out.print("Enter Product ID to remove: ");

String productIdToRemove = scanner.nextLine();

inventory.removeProduct(productIdToRemove);

break;

case 4:

inventory.listProducts();

break;

case 5:

inventory.saveToFile(filename);

break;

case 6:

System.out.println("Saving data to file and exiting...");

inventory.saveToFile(filename);

scanner.close();

System.exit(0);

default:

System.out.println("Invalid choice! Try again.");

}

}

}

}

INVENTORY CLASS:

import java.io.\*;

import java.util.ArrayList;

public class Inventory {

private ArrayList<Product> products;

public Inventory() {

this.products = new ArrayList<>();

}

public void addProduct(Product product) {

products.add(product);

System.out.println("Product added successfully!");

}

public void removeProduct(String productId) {

products.removeIf(product -> product.getProductId().equals(productId));

System.out.println("Product removed successfully (if it existed).");

}

public Product searchProduct(String searchTerm) {

for (Product product : products) {

if (product.getName().equalsIgnoreCase(searchTerm) || product.getProductId().equals(searchTerm)) {

return product;

}

}

return null;

}

public void listProducts() {

if (products.isEmpty()) {

System.out.println("No products in inventory.");

} else {

for (Product product : products) {

System.out.println(product);

}

}

}

public void saveToFile(String filename) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {

for (Product product : products) {

writer.write(productToString(product)); // Save product as a comma-separated string

writer.newLine();

}

System.out.println("Products saved to file successfully.");

} catch (IOException e) {

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

}

}

public void loadFromFile(String filename) {

try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {

String line;

while ((line = reader.readLine()) != null) {

Product product = parseProductFromString(line);

if (product != null) {

products.add(product);

}

}

System.out.println("Products loaded from file successfully.");

} catch (FileNotFoundException e) {

System.out.println("File not found. Starting with an empty inventory.");

} catch (IOException e) {

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

}

}

private String productToString(Product product) {

if (product instanceof Shirt) {

Shirt shirt = (Shirt) product;

return "Shirt," + shirt.getProductId() + "," + shirt.getName() + "," + shirt.getPrice() + ","

+ shirt.getQuantity() + "," + shirt.getColor() + "," + shirt.getBrand() + "," + shirt.getSize();

} else if (product instanceof Pants) {

Pants pants = (Pants) product;

return "Pants," + pants.getProductId() + "," + pants.getName() + "," + pants.getPrice() + ","

+ pants.getQuantity() + "," + pants.getMaterial() + "," + pants.getSize();

} else if (product instanceof Shoe) {

Shoe shoe = (Shoe) product;

return "Shoe," + shoe.getProductId() + "," + shoe.getName() + "," + shoe.getPrice() + ","

+ shoe.getQuantity() + "," + shoe.getBrand() + "," + shoe.getShoeSize() + "," + shoe.getType();

} else if (product instanceof Jacket) {

Jacket jacket = (Jacket) product;

return "Jacket," + jacket.getProductId() + "," + jacket.getName() + "," + jacket.getPrice() + ","

+ jacket.getQuantity() + "," + jacket.getMaterial() + "," + jacket.getColor() + "," + jacket.getSize();

}

return null;

}

private Product parseProductFromString(String line) {

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

switch (parts[0]) {

case "Shirt":

return new Shirt(parts[1], parts[2], Double.parseDouble(parts[3]),

Integer.parseInt(parts[4]), parts[5], parts[6], Size.valueOf(parts[7]));

case "Pants":

return new Pants(parts[1], parts[2], Double.parseDouble(parts[3]),

Integer.parseInt(parts[4]), Size.valueOf(parts[5]), parts[6]);

case "Shoe":

return new Shoe(parts[1], parts[2], Double.parseDouble(parts[3]),

Integer.parseInt(parts[4]), parts[5], Integer.parseInt(parts[6]), parts[7]);

case "Jacket":

return new Jacket(parts[1], parts[2], Double.parseDouble(parts[3]),

Integer.parseInt(parts[4]), parts[5], parts[6], Size.valueOf(parts[7]));

default:

return null;

}

}

}

PRODUCT CLAS:

public class Product {

private String productId;

private String name;

private double price;

private int quantity;

public Product(String productId, String name, double price, int quantity) {

this.productId = productId;

this.name = name;

this.price = price;

this.quantity = quantity;

}

public String getProductId() {

return productId;

}

public void setProductId(String productId) {

this.productId = productId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public void updateStock(int amount) {

if (this.quantity + amount < 0) {

System.out.println("Error: Stock cannot be negative!");

} else {

this.quantity += amount;

}

}

@Override

public String toString() {

return "Product [ID=" + productId + ", Name=" + name + ", Price=" + price + ", Quantity=" + quantity + "]";

}

}

JACKET CLASS:

public class Jacket extends Product {

private String material;

private String color;

private Size size;

public Jacket(String productId, String name, double price, int quantity, String material, String color, Size size) {

super(productId, name, price, quantity);

this.material = material;

this.color = color;

this.size = size;

}

public String getMaterial() { return material; }

public void setMaterial(String material) { this.material = material; }

public String getColor() { return color; }

public void setColor(String color) { this.color = color; }

public Size getSize() { return size; }

public void setSize(Size size) { this.size = size; }

@Override

public String toString() {

return super.toString() + ", Material=" + material + ", Color=" + color + ", Size=" + size;

}

}

PANT CLASS:

public class Pants extends Product {

private Size size;

private String material;

public Pants(String productId, String name, double price, int quantity, Size size, String material) {

super(productId, name, price, quantity);

this.size = size;

this.material = material;

}

public Size getSize() { return size; }

public void setSize(Size size) { this.size = size; }

public String getMaterial() { return material; }

public void setMaterial(String material) { this.material = material; }

@Override

public String toString() {

return super.toString() + ", Size=" + size + ", Material=" + material;

}

}

SHIRT CLASS:

public class Shirt extends Product {

private String color;

private String brand;

private Size size;

public Shirt(String productId, String name, double price, int quantity, String color, String brand, Size size) {

super(productId, name, price, quantity);

this.color = color;

this.brand = brand;

this.size = size;

}

public String getColor() {

return color; }

public void setColor(String color) { this.color = color; }

public String getBrand() { return brand; }

public void setBrand(String brand) { this.brand = brand; }

public Size getSize() { return size; }

public void setSize(Size size) { this.size = size; }

@Override

public String toString() {

return super.toString() + ", Color=" + color + ", Brand=" + brand + ", Size=" + size;

}

}

SHOE CLASS:

public class Shoe extends Product {

private String brand;

private double shoeSize;

private String type;

public Shoe(String productId, String name, double price, int quantity, String brand, double shoeSize, String type) {

super(productId, name, price, quantity);

this.brand = brand;

this.shoeSize = shoeSize;

this.type = type;

}

public String getBrand() { return brand; }

public void setBrand(String brand) { this.brand = brand; }

public double getShoeSize() { return shoeSize; }

public void setShoeSize(double shoeSize) { this.shoeSize = shoeSize; }

public String getType() { return type; }

public void setType(String type) { this.type = type; }

@Override

public String toString() {

return super.toString() + ", Brand=" + brand + ", ShoeSize=" + shoeSize + ", Type=" + type;

}

}

ENUM:

public enum Size {

SMALL, MEDIUM, LARGE

}

WATCH CLASS:

public class Watch extends Product {

private String brand; // Brand of the watch

private String material; // Material of the watch (e.g., leather, steel)

private String type; // Type of watch (e.g., analog, digital)

public Watch(String productId, String name, double price, int quantity, String brand, String material, String type) {

super(productId, name, price, quantity);

this.brand = brand;

this.material = material;

this.type = type;

}

// Getters and Setters

public String getBrand() { return brand; }

public void setBrand(String brand) { this.brand = brand; }

public String getMaterial() { return material; }

public void setMaterial(String material) { this.material = material; }

public String getType() { return type; }

public void setType(String type) { this.type = type; }

@Override

public String toString() {

return super.toString() + ", Brand=" + brand + ", Material=" + material + ", Type=" + type;

}

}

USER CLASS:

public class User {

private String username;

private String password;

public User(String username, String password) {

this.username = username;

this.password = password;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

}

USERMANAGER:

import java.io.\*;

import java.util.ArrayList;

public class UserManager {

private ArrayList<User> users;

private static final String FILE\_NAME = "users.txt";

public UserManager() {

users = new ArrayList<>();

loadUsersFromFile();

}

private void loadUsersFromFile() {

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

String line;

while ((line = reader.readLine()) != null) {

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

if (parts.length == 2) {

users.add(new User(parts[0], parts[1]));

}

}

} catch (IOException e) {

System.out.println("No previous users found, starting with an empty user list.");

}

}

public void saveUsersToFile() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME))) {

for (User user : users) {

writer.write(user.getUsername() + "," + user.getPassword());

writer.newLine();

}

} catch (IOException e) {

System.out.println("Error saving users to file.");

}

}

public boolean registerUser(String username, String password) {

for (User user : users) {

if (user.getUsername().equals(username)) {

System.out.println("Username already exists!");

return false;

}

}

users.add(new User(username, password));

saveUsersToFile();

return true;

}

public boolean loginUser(String username, String password) {

for (User user : users) {

if (user.getUsername().equals(username) && user.getPassword().equals(password)) {

return true;

}

}

return false;

}

}