import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.util.List;

// Item class to represent an auction item

class Item {

private String itemName;

private double startingPrice;

private Bid highestBid;

public Item(String itemName, double startingPrice) {

this.itemName = itemName;

this.startingPrice = startingPrice;

}

public String getItemName() {

return itemName;

}

public double getStartingPrice() {

return startingPrice;

}

public Bid getHighestBid() {

return highestBid;

}

public void setHighestBid(Bid bid) {

this.highestBid = bid;

}

@Override

public String toString() {

return "Item: " + itemName + ", Starting Price: $" + startingPrice;

}

}

// Bid class to represent a bid

class Bid {

private String bidderName;

private double bidAmount;

public Bid(String bidderName, double bidAmount) {

this.bidderName = bidderName;

this.bidAmount = bidAmount;

}

public String getBidderName() {

return bidderName;

}

public double getBidAmount() {

return bidAmount;

}

@Override

public String toString() {

return "Bidder: " + bidderName + ", Bid Amount: $" + bidAmount;

}

}

// AuctionSystem class to manage items and bids

class AuctionSystem {

private List<Item> items = new ArrayList<>();

public void addItem(String itemName, double startingPrice) {

items.add(new Item(itemName, startingPrice));

}

public String placeBid(String itemName, String bidderName, double bidAmount) {

Item item = findItemByName(itemName);

if (item == null) {

return "Item not found.";

}

if (bidAmount < item.getStartingPrice()) {

return "Bid must be higher than the starting price.";

}

if (item.getHighestBid() != null && bidAmount <= item.getHighestBid().getBidAmount()) {

return "Bid must be higher than the current highest bid.";

}

item.setHighestBid(new Bid(bidderName, bidAmount));

return "Bid placed successfully!";

}

public List<Item> getItems() {

return items;

}

private Item findItemByName(String itemName) {

for (Item item : items) {

if (item.getItemName().equalsIgnoreCase(itemName)) {

return item;

}

}

return null;

}

}

// Main GUI class

public class AuctionSystemGUI {

private AuctionSystem auctionSystem = new AuctionSystem();

public static void main(String[] args) {

new AuctionSystemGUI().createAndShowGUI();

}

public void createAndShowGUI() {

JFrame frame = new JFrame("Auction Management System");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(600, 400);

JPanel panel = new JPanel();

panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));

frame.add(panel);

JButton addItemButton = new JButton("Add Item");

JButton placeBidButton = new JButton("Place Bid");

JButton showItemsButton = new JButton("Show Items");

panel.add(addItemButton);

panel.add(placeBidButton);

panel.add(showItemsButton);

addItemButton.addActionListener(e -> addItemDialog(frame));

placeBidButton.addActionListener(e -> placeBidDialog(frame));

showItemsButton.addActionListener(e -> showItemsDialog(frame));

frame.setVisible(true);

}

private void addItemDialog(JFrame frame) {

JTextField itemNameField = new JTextField();

JTextField startingPriceField = new JTextField();

Object[] message = {

"Item Name:", itemNameField,

"Starting Price:", startingPriceField,

};

int option = JOptionPane.showConfirmDialog(frame, message, "Add Item", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

String itemName = itemNameField.getText();

double startingPrice;

try {

startingPrice = Double.parseDouble(startingPriceField.getText());

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(frame, "Invalid starting price. Please enter a valid number.");

return;

}

auctionSystem.addItem(itemName, startingPrice);

JOptionPane.showMessageDialog(frame, "Item added successfully!");

}

}

private void placeBidDialog(JFrame frame) {

JTextField itemNameField = new JTextField();

JTextField bidderNameField = new JTextField();

JTextField bidAmountField = new JTextField();

Object[] message = {

"Item Name:", itemNameField,

"Bidder Name:", bidderNameField,

"Bid Amount:", bidAmountField,

};

int option = JOptionPane.showConfirmDialog(frame, message, "Place Bid", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

String itemName = itemNameField.getText();

String bidderName = bidderNameField.getText();

double bidAmount;

try {

bidAmount = Double.parseDouble(bidAmountField.getText());

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(frame, "Invalid bid amount. Please enter a valid number.");

return;

}

String result = auctionSystem.placeBid(itemName, bidderName, bidAmount);

JOptionPane.showMessageDialog(frame, result);

}

}

private void showItemsDialog(JFrame frame) {

List<Item> items = auctionSystem.getItems();

StringBuilder message = new StringBuilder("Auction Items:\n");

for (Item item : items) {

message.append(item.toString()).append("\n");

if (item.getHighestBid() != null) {

message.append("Current Highest Bid: ").append(item.getHighestBid()).append("\n");

} else {

message.append("No bids yet.\n");

}

message.append("\n");

}

if (items.isEmpty()) {

message.append("No items available.");

}

JOptionPane.showMessageDialog(frame, message.toString(), "Show Items", JOptionPane.INFORMATION\_MESSAGE);

}

}