Medical Equipment Inventory Management System

Scenario:

You are tasked with developing a Medical Equipment Inventory Management System in C#.

The system should allow hospital staff to manage different types of medical equipment stored in a healthcare facility, ensuring efficient tracking of inventory.

Implement the MedicalEquipmentInventorySystem class to handle equipment storage operations, focusing on CREATE, READ, and DELETE functionalities.

MedicalEquipment Class

Properties:

- EquipmentName (string): Represents the name of the medical equipment (e.g., Defibrillator, X-Ray Machine).
- Manufacturer (string): Represents the manufacturer of the equipment (e.g., Philips, GE Healthcare).
- Quantity (int): Represents the total quantity of the equipment available in the hospital.
- UnitCost (double): Represents the cost of one unit of the equipment.

Constructor:

- **MedicalEquipment(string equipmentName, string manufacturer, int quantity, double unitCost)**

Initializes a new medical equipment entry with the specified details.

Custom Exception

Create a custom exception class named EquipmentNotFoundException, which will handle errors when attempting to DELETE or GET an equipment entry that does not exist in the system.

Constructor:

- **EquipmentNotFoundException(string message)**

Initializes a new instance of the exception with the message "Equipment not found."

MedicalEquipmentInventorySystem Class

Properties:

Equipments (List<MedicalEquipment>):

A list that stores the various medical equipment available in the hospital's inventory.

Methods:

AddEquipment(string equipmentName, string manufacturer, int quantity, double unitCost)

- Adds a new medical equipment item to the system.
- If an equipment with the same name and manufacturer already exists, display:
  - "Equipment already exists."
- If the addition is successful, display:
  - "Equipment added successfully."

After successful addition, retrieve and display the newly added equipment details.

GetEquipmentDetails(string equipmentName, string manufacturer)

- Returns the details of a medical equipment item by its name and manufacturer.
- If found, display:
  - "Equipment Name: [EquipmentName], Manufacturer: [Manufacturer], Quantity: [Quantity], Unit Cost: [UnitCost]"
- If not found, throw an EquipmentNotFoundException and display:
  - "Equipment not found."

RemoveEquipment(string equipmentName, string manufacturer)

- Removes a medical equipment item from the system by its name and manufacturer.
- If not found, throw an EquipmentNotFoundException and display:
- "Equipment not found."
- If successful, display:
- "Equipment removed successfully."

Program Class (Main Menu Operations)

The Program class serves as the main entry point of the system.

It handles a menu-driven interface for interacting with the Medical Equipment Inventory System.

Operations:

AddEquipment

Inputs:

- Equipment Name (string)
- Manufacturer (string)
- Quantity (int)
- Unit Cost (double)

If added successfully: "Equipment added successfully."

If duplicate: "Equipment already exists."

Display details after addition.

GetEquipmentDetails

Inputs:

- Equipment Name (string)
- Manufacturer (string)

If found:

"Equipment Name: [EquipmentName], Manufacturer: [Manufacturer], Quantity: [Quantity], Unit Cost: [UnitCost]"

If not found: "Equipment not found."

RemoveEquipment

Inputs:

- Equipment Name (string)
- Manufacturer (string)

If successfully removed: "Equipment removed successfully."

If not found: "Equipment not found."

Exit

- Displays: "Exiting program..."

Error Handling Messages

- If a number other than 1–4 is entered:
- "Invalid choice. Please select a number from 1 to 4."
- If input is invalid (non-numeric or incorrect format):
- "Invalid input. Please enter a number."

Event Ticket Booking System

You are tasked with building an **Event Ticket Booking System** in C#, applying object-oriented programming principles. The application allows staff to book tickets for events, view all bookings, and delete bookings when a user cancels.

Classes

1. TicketBooking Class

Represents a booking with the following properties:

- **BookingID (int)**: A unique identifier for the booking.
- **CustomerName (string)**: Name of the person booking the ticket.
- **EventName (string)**: Name of the event.
- **BookingDate (string)**: Date of the booking.
- **ContactInfo (string)**: Customer's contact details.

Constructor:

public TicketBooking(int bookingId, string customerName, string eventName, string bookingDate, string contactInfo)

Initializes booking details.

Method:

public void DisplayDetails()

Displays:

"Customer: {CustomerName}, ID: {BookingID}, Event: {EventName}, Date: {BookingDate}, Contact: {ContactInfo}"

2. BookingNotFoundException Class

A custom exception thrown when trying to delete a non-existent booking.

Constructor:
public BookingNotFoundException(string message)
Initialize with:
"Booking with ID {id} not found"

3. TicketBookingManager Class

Manages all booking-related operations.

- **bookings**: A list to store TicketBooking objects.

Methods:

public void AddBooking(TicketBooking booking)

- If ID exists: **"Booking ID already exists"**
- On success: **"Booking confirmed successfully"**

public void DisplayAllBookings()

- If empty: **"No bookings available"**
- Else, display all.

public void DeleteBooking(int id)

- On success: **"Booking with ID {id} cancelled successfully"**
- If not found: throw BookingNotFoundException.

Main Program

Provides menu:

1. Book Ticket
2. Display All Bookings
3. Cancel Booking
4. Exit

Input Format

- **Menu Choice:** Integer (1–4)

Option 1: Book Ticket

- BookingID (int)
- CustomerName (string)
- EventName (string)
- BookingDate (string) ("YYYY-MM-DD")
- ContactInfo (string)

Option 2: Display All Bookings
(No input)

Option 3: Cancel Booking
BookingID (int)

Option 4: Exit
(No input)

Output Format

Option 1:
- Success: **"Booking confirmed successfully"**
- If ID exists: **"Booking ID already exists"**

Option 2:
- No records: **"No bookings available"**
- Else: formatted output

Option 3:
- Success: **"Booking with ID {id} cancelled successfully"**
- Not found: throw BookingNotFoundException with message

Option 4:
"Exiting the program..."
Invalid Choice:

"Invalid choice. Please try again"