Using Arrays and Lists with OOP in C#

Review – Why Collections?

In the real world, we rarely deal with one object at a time. We need to **manage groups of objects** (e.g., list of rooms, list of guests, list of employees).

Arrays vs Lists

Feature	Array	List
Size	Fixed	Dynamic
Syntax	Room[] rooms = new Room[5];	List<Room> rooms = new List<Room>();
Flexibility	Less flexible	More powerful (Add, Remove, etc.)

Arrays of Objects

Example: Array of 3 Guests

```
class Guest
{
  public string Name { get; set; }
  public string NationalID { get; set; }

  public Guest(string name, string id)
  {
    Name = name;
    NationalID = id;
  }

  public void Display()
  {
    Console.WriteLine($"Name: {Name}, ID: {NationalID}");
  }
}
```

```
class Program
{
    static void Main()
    {
        Guest[] guests = new Guest[3];
        guests[0] = new Guest("Karim", "EG123");
        guests[1] = new Guest("Sara", "EG456");
        guests[2] = new Guest("Ali", "EG789");

        foreach (Guest g in guests)
        {     g.Display(); }
    }
}
```

List of Objects

Example: List of Rooms

```
class Room
{
    public int RoomNumber { get; set; }
    public bool IsBooked { get; private set; }
    public Room(int number)
    {   RoomNumber = number;
        IsBooked = false;   }
    public void Book()
    {
        if (!IsBooked)
        {   IsBooked = true;
            Console.WriteLine($"Room {RoomNumber} booked.");
        }
    }
}
```

```
class Program
{
    static void Main()
    {
        List<Room> rooms = new List<Room>();

        // Adding rooms
        rooms.Add(new Room(101));
        rooms.Add(new Room(102));
        rooms.Add(new Room(103));

        // Booking all rooms
        foreach (Room r in rooms)
        {
            r.Book();
        }
    }
}
```

Operations on Lists

Add, Search, Update, Delete

```
class Guest
{
    public string Name { get; set; }
    public string NationalID { get; set; }

    public Guest(string name, string id)
    {
        Name = name;
        NationalID = id;
    }
}
```

```
class Program
{
    static void Main()
    {
        List<Guest> guests = new List<Guest>();

        // Add guests
        guests.Add(new Guest("Laila", "EG001"));
        guests.Add(new Guest("Youssef", "EG002"));

        // Search for a guest
        string searchID = "EG002";
        Guest found = guests.Find(g => g.NationalID == searchID);
        if (found != null)
        {
            Console.WriteLine($"Guest found: {found.Name}");
        }

        // Remove guest by ID
        guests.RemoveAll(g => g.NationalID == "EG001");

        // Display remaining guests
        foreach (var g in guests)
        {
            Console.WriteLine($"Name: {g.Name}, ID: {g.NationalID}");
        }
    }
}
```

Mini System – Hotel Management with List

Task Overview:

Build a system to manage:

- List of rooms (add, book, display available)
- List of guests (add, search by ID)

Design Overview:

```
class Hotel
{
    public List<Room> Rooms { get; set; } = new List<Room>();
    public List<Guest> Guests { get; set; } = new List<Guest>();

    public void AddRoom(int roomNumber)
    {
        Rooms.Add(new Room(roomNumber));
    }

    public void AddGuest(string name, string id)
    {
        Guests.Add(new Guest(name, id));
    }

    public void DisplayAvailableRooms()
    {
        foreach (var room in Rooms)
        {
            if (!room.IsBooked)
                Console.WriteLine($"Room {room.RoomNumber} is available.");
        }
    }
}
```

Discussion Points

- Why would we use List<Room> instead of Room[]?
- What happens if we don't encapsulate fields like IsBooked?
- What's the difference between Find() and foreach?
- Can you reuse the same Guest for multiple bookings?

Practice Tasks

1. Create a Hotel class that allows:
   - Adding rooms and guests
   - Searching a guest by national ID
   - Displaying all booked rooms
2. Add a method to allow cancelling a room booking.
3. Create an array of 5 employees. Ask the user to input data for each employee, then display them.
4. Advanced: Create a mini system where each guest can book multiple rooms.