Customer Profile Entity Example

Classes and Their Roles:

1. **`CustomerProfile` (Class):**
    - Represents the business entity `Customer Profile` with attributes: name, postcode, phone number, email, and payment status.
    - Handles validation logic for its attributes using regex and ensures data integrity before it's persisted in the database.
    - Acts as a blueprint for objects that represent customer data.

2. **`CustomerProfileCommandLine` (User Interface Layer):**
    - Implements `CommandLinesExecution`, making it a console-based menu interface for interacting with the `CustomerProfileCRUD` class.
    - Role:
        - Displays a menu of options (e.g., Create, Read, Update, Delete).
        - Collects user input for customer attributes.
        - Calls appropriate methods from `CustomerProfileCRUD` to perform the requested operations.
        - Example:
            - If a user chooses "1. Create Customer Account," it collects the necessary data, validates it via `CustomerProfile`, and persists it using `CustomerProfileCRUD`.

3. **`CustomerProfileCRUD` (Data Access Object):**
    - Provides methods to perform CRUD (Create, Read, Update, Delete) operations on the `customer_profile` table in the database.
    - Uses `EntitiesMySQLAccess` to establish the connection.
    - Translates Java objects (`CustomerProfile`) into SQL queries and vice versa.
    - Example:
        - **Create**: Saves a `CustomerProfile` object into the database.
        - **Read**: Retrieves all or specific customer profiles as `ResultSet`.
        - **Update**: Updates the details of a customer record by ID.
        - **Delete**: Deletes records based on the ID or deletes all records when instructed.

4. **`CustomerProfile` (Class):**
    - Represents the business entity `Customer Profile` with attributes: name, postcode, phone number, email, and payment status.
    - Handles validation logic for its attributes using regex and ensures data integrity before it's persisted in the database.
    - Acts as a blueprint for objects that represent customer data.

5. **`EntitiesExceptionHandler` (Custom Exception Handler):**
    - Captures and manages exceptions specific to the system.
    - Provides a way to customise error messages and ensure they are meaningful for users or developers debugging the application.
    - Used extensively in `CustomerProfile` validations to handle input errors.

6. **`EntitiesMySQLAccess` (Database Access Layer):**
    - Manages the database connection.
    - Provides a reusable connection mechanism for database-related operations across the system.
    - Abstracts the connection details such as the host, user, and password.

7. **`CommandLinesExecution` (Interface):**
    - Serves as a blueprint for command-line execution classes.
    - Ensures any implementing class (like `CustomerProfileCommandLine`) provides an `execute()` method, which serves as the entry point for the command-line program.

Data Flow / Integration:

1. **User Interaction**:
    - A user interacts with the **`CustomerProfileCommandLine`** menu via the console.
    - Their choice determines the operation (e.g., create, read, update, delete).
2. **Input Handling and Validation**:
    - User inputs are captured, and a **`CustomerProfile`** object is created.
    - The object's attributes are validated within the `CustomerProfile` class (e.g., name, postcode) to ensure correctness.
3. **Database Interaction**:
    - If validation passes, the operation proceeds.
    - The **`CustomerProfileCRUD`** class translates the operation into SQL queries and uses **`EntitiesMySQLAccess`** to interact with the database.
4. **Error Handling**:
    - Any validation failure or database error triggers an **`EntitiesExceptionHandler`**, ensuring a smooth user experience with clear error messages.
5. **Output Handling**:
    - Results of operations (e.g., "Customer Details Saved" or table data) are displayed to the user via the **`CustomerProfileCommandLine`** interface.

System Workflow Example:

Create Customer Record:

1. User selects "1. Create Customer Account" from the console menu.
2. `CustomerProfileCommandLine` collects inputs for name, postcode, phone number, email, and payment status.
3. A `CustomerProfile` object is created, and its attributes are validated.
4. If valid, `CustomerProfileCRUD.createCustomerDetailsAccount()` is called.
5. SQL is executed via `EntitiesMySQLAccess`, and the record is inserted into the database.
6. A success message is displayed to the user.