**Data Modeling for Classes**

**Classes are divided into two categories: Entity Classes, which represent data in transition and manipulation, and Administrative Classes, which handle application management and operations. Administrative classes should preferably be static.**

**1. Entity Classes**

**1.1. IpAddress Class**

* **Description: Represents the primary object of the application, related to IP addresses. Automatically generated by Entity Framework.**
* **Attributes:**
  + **IpNumber (String): Unique identifier for the IP.**
  + **Id (int): Unique identifier.**
  + **CountryId (int): Foreign key for the associated country.**
  + **Country (Country): Reference to the country related to this IP.**
  + **CreatedAt (DateTime): Timestamp of creation.**
  + **LastUpdate (Date): Date of the last update for this IP.**
* **Functions:**
  + **Constructor: Initializes the class with defined attributes (via partials).**
  + **Getters and Setters: Methods for accessing and modifying attributes.**

**1.2. Country Class**

* **Description: Represents a country and its unique attributes.**
* **Attributes:**
  + **Id (int): Unique identifier.**
  + **CountryName (String): Full name of the country.**
  + **TwoLCode (String): Two-letter ISO code.**
  + **ThreeLCode (String): Three-letter ISO code.**
  + **CreatedAt (DateTime): Timestamp of creation.**
  + **IpAddresses (List<IpAddress>): Collection of associated IP addresses.**
* **Functions:**
  + **Constructor: Initializes the class with defined attributes.**
  + **Getters and Setters: Methods for accessing and modifying attributes.**

**1.3. IpLocatorContext Class**

* **Description: Automatically created by Entity Framework to interact with the database. No modifications are required.**

**2. Administrative Classes**

**All administrative classes will be static. To facilitate access, these classes should have concise, meaningful names. Proper commenting is crucial to explain the purpose of each class and method.**

**2.1. CM (Cache Memory) Class**

* **Description: Manages all operations related to cache memory.**
* **Attributes: None (relies on IMemoryCache framework).**
* **Functions:**
  + **HasKey: Checks if a key exists in the cache.**
  + **GetData: Retrieves the value associated with a key, if it exists.**
  + **AddData: Adds a new key-value pair to the cache.**
  + **UpdateData: Updates the value for an existing key in the cache.**
  + **ResetTTL: Resets the TTL (Time to Live) of an entry in the cache to extend its validity.**

**2.2. DB (Database) Class**

* **Description: Manages all operations related to the database, including CRUD operations, validation, and integration with Entity Framework.**
* **Attributes: None (all logic encapsulated in methods).**
* **Functions:**
  + **GetCountry: Retrieves a country by its two-letter ISO code.**
  + **GetIP: Retrieves an IP stored in the database.**
  + **AddCountry: Adds a new country to the database.**
  + **AddIP: Adds a new IP to the database.**
  + **UpdateDatabase: Updates database records by comparing them with the external API.**
  + **Generic CRUD methods: For reading, writing, updating, and deleting database entries.**
  + **GetId(IP and Country): Returns the ID of a country or IP.**
  + **UtilReports: Functions for generating reports without Entity Framework:**
    - **GetAllReports: Returns a DataTable of all countries and their data in the database.**
    - **GetSelectedReports: Accepts a list of codes, validates them, processes a query, and returns a tuple containing a DataTable and an error message (empty if no errors occurred).**
    - **ValidateCode: Inspects a list of strings, identifies valid database keys, and returns a list of invalid keys.**
    - **IsItemInDatabase: Verifies if a string exists as a valid item in the database.**

**2.3. EA (External API) Class**

* **Description: Manages all interactions with the external public API (IP2C).**
* **Attributes: None (all logic encapsulated in methods).**
* **Functions:**
  + **GetIp: Sends a request to the external API for an IP.**

**2.4. DS (Data Serializer) Class**

* **Description: Converts IP objects into anonymous objects for serialization and API responses.**
* **Attributes: None (all logic encapsulated in methods).**
* **Functions:**
  + **GetCountry: Accepts an IpAddress object and returns an anonymous object containing the country name, two-letter code, and three-letter code.**
  + **GetUpdateReports: Used internally for manual testing to verify data consistency in the database. Generates a report convertible to an anonymous object.**
  + **GetReports: Accepts a DataTable and a string (error message). The DataTable is converted into a list of dictionaries (string: object), later serialized into JSON by the main application.**

**2.5. UD (Update Data) Class**

* **Description: Encapsulates the logic for batch data updates in the database and cache.**
* **Attributes: None (all logic encapsulated in methods).**
* **Functions:**
  + **UpdateDatabase: Scans the database in batches of 100 records, compares with the external API, and updates inconsistent data.**

**Auxiliary and Testing Classes**

* **HT, HS, EH: Contain automation logic for testing classes.**
* **ET, RT: Classes for task-specific testing logic.**