Laiba Maab

CID: DEP2248

Task1: Basic Level (Manage Locations, WeatherForeCastData, Historical WeatherData**). Expert level (**Offline Mode**)**

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <fstream>

class Location
{
public:
    std::string name;
    double latitude;
    double longitude;
    Location(std::string name, double latitude, double longitude)
        : name(name), latitude(latitude), longitude(longitude) {}
};
class LocationManager
{
private:
    std::vector<Location> locations;
public:
    void addLocation(const Location& location)
    {
        locations.push_back(location);
    }
```

```
void removeLocation(const std::string& locationName)
{
   locations.erase(std::remove_if(locations.begin(),
      locations.end(),
                           [&locationName](Location& loc)
                           {
                              return loc.name == locationName;
                           }),
                 locations.end());
}

std::vector<Location> listLocations() const
{ return locations;}

void saveToFile(const std::string& filename)
{
   std::ofstream file(filename);
   if (file.is_open())
   {
      for (const auto& loc : locations)
      {
         file << loc.name << "," << loc.latitude << "," <<
      loc.longitude << "\n";
      }
      file.close();
   }
}
void loadFromFile(const std::string& filename)
{
```

```
        std::ifstream file(filename);
        if (file.is_open())
        {
            locations.clear();
            std::string line;
            while (std::getline(file, line))
            {
                size_t pos1 = line.find(',');
                size_t pos2 = line.find(',', pos1 + 1);
                std::string name = line.substr(0, pos1);
                double latitude = std::stod(line.substr(pos1 + 1, pos2 -
            pos1 - 1));
                double longitude = std::stod(line.substr(pos2 + 1));
                locations.emplace_back(name, latitude, longitude);
            }
            file.close();
        } }};
class WeatherVariable
{
private:
    std::string name;
    double value;

public:
    WeatherVariable(std::string name, double value)
        : name(name), value(value) {}

    void setValue(double newValue)
    { value = newValue; }
```

```
    double getValue() const
    { return value;}
    std::string getName() const
    { return name;}
};
class WeatherForecastingSystem
{
private:
    std::string apiUrl;
public:
    WeatherForecastingSystem(const std::string& apiUrl)
        : apiUrl(apiUrl) {}

    std::string fetchWeatherData(double latitude, double longitude)
    {
        return "Simulated weather data for coordinates (" +
            std::to_string(latitude) + ", " + std::to_string(longitude) +
            ")";
    }
    void displayWeatherData(const std::string& weatherData)
    {
        std::cout << weatherData << std::endl;
    }
};
class HistoricalWeatherSystem
{
private:
    std::string apiUrl;
public:
```

```
    HistoricalWeatherSystem(const std::string& apiUrl)
        : apiUrl(apiUrl) {}

    std::string fetchHistoricalData(double latitude, double longitude)
    {
        return "Simulated historical weather data for coordinates (" +
            std::to_string(latitude) + ", " + std::to_string(longitude) +
            ")";
    }
    void displayHistoricalData(const std::string& historicalData)
    {
        std::cout << historicalData << std::endl;
    }
};
class DataExporter
{
public:
    static void exportToCSV(const std::string& data, const
            std::string& filename)
    {
        std::ofstream file(filename);
        if (file.is_open())
        {
            file << data;
            file.close();
        }
    }
    static void exportToJSON(const std::string& data, const
            std::string& filename)
    {
```

```
        std::ofstream file(filename);
        if (file.is_open())
        {
            file << "{\n";
            file << "  \"data\": \"" << data << "\"\n";
            file << "}\n";
            file.close();
        } } };
class CloudDatabase
{
private:
    std::string cloudData;
public:
    void saveToCloud(const std::string& data)
    {
        cloudData = data;
        std::cout << "Saving data to cloud: " << data << std::endl;
    }
    std::string loadFromCloud()
    {
        std::cout << "Loading data from cloud: " << cloudData <<
            std::endl;
        return cloudData;
    }
};
int main()
{
    LocationManager locationManager;
    Location loc1("New York", 40.7128, -74.0060);
```

```
Location loc2("San Francisco", 37.7749, -122.4194);


locationManager.addLocation(loc1);

locationManager.addLocation(loc2);

locationManager.removeLocation("New York");


std::cout << "Locations:" << std::endl;

for (const auto& loc : locationManager.listLocations()) {

   std::cout << loc.name << " - Latitude: " << loc.latitude << ",
      Longitude: " << loc.longitude << std::endl;

}


locationManager.saveToFile("locations.csv");

locationManager.loadFromFile("locations.csv");


WeatherVariable temperature("Temperature", 25);

WeatherVariable windSpeed("Wind Speed", 10);


std::cout << "\nWeather Variables:" << std::endl;

std::cout << temperature.getName() << ": " <<
      temperature.getValue() << "°C" << std::endl;

std::cout << windSpeed.getName() << ": " <<
      windSpeed.getValue() << " km/h" << std::endl;


std::string apiUrl = "https://api.open-meteo.com/v1/forecast";

WeatherForecastingSystem weatherSystem(apiUrl);

double latitude = 37.7749;

double longitude = -122.4194;
```

```
    std::cout << "\nWeather Forecast Data:" << std::endl;
    std::string weatherData =
        weatherSystem.fetchWeatherData(latitude, longitude);
    weatherSystem.displayWeatherData(weatherData);

    HistoricalWeatherSystem historicalSystem(apiUrl);
    std::cout << "\nHistorical Weather Data:" << std::endl;
    std::string historicalData =
        historicalSystem.fetchHistoricalData(latitude, longitude);
    historicalSystem.displayHistoricalData(historicalData);

    DataExporter::exportToCSV(weatherData, "weather_data.csv");
    DataExporter::exportToJSON(weatherData,
        "weather_data.json");

    CloudDatabase cloudDb;
    cloudDb.saveToCloud(weatherData);
    std::string cloudData = cloudDb.loadFromCloud();
    std::cout << "\nData from Cloud: " << cloudData << std::endl;
    return 0;
}
```

https://onlinegdb.com/RNmErqq6B