Functionality 1: Credit Card Fraud Detection System

Description:

This system is responsible for detecting potentially fraudulent transactions in real-time. It analyzes a set of criteria including transaction amount, frequency of transactions, location changes, and the use of blacklists to determine whether a transaction should be flagged as fraudulent, require verification, or block the card. Additionally, it calculates a **risk score** for each transaction, providing a more nuanced measure of suspicion.

Input Domain:

- **Transaction Amount:** double (e.g., 15000.00)
- **Transaction Time:** LocalDateTime (e.g., 2024-10-01T12:00:00)
- **Transaction Location:** String (e.g., "USA", "France", "HighRiskCountry")
- **Previous Transactions:** List of transactions (amount, time, location)
- **Blacklisted Locations:** List of locations (e.g., "HighRiskCountry")

Output:

- **Fraud Flag:** boolean (indicates whether the transaction is considered fraudulent)
- **Blocked Status:** boolean (indicates whether the card is blocked)
- **Verification Required:** boolean (indicates whether the transaction requires additional verification)
- **Risk Score:** int (0-100, measures the risk level of the transaction)

Rules:

1. **High Transaction Amount:**
    - If the transaction amount exceeds a configurable threshold (e.g., $10,000), the transaction is flagged as potentially fraudulent, requiring further verification, risk score +50.
2. **Excessive Transactions in Short Time Frame:**
    - If more than 10 transactions occur within a 1-hour period, the card is temporarily **blocked** for security reasons. This prevents further transactions until the cardholder is contacted or the block is lifted manually, risk score +30.
3. **Location Change within Short Time Frame:**
    - If a transaction occurs in a different geographical location than the previous transaction and within a short time frame (e.g., 30 minutes), the transaction is flagged as potentially fraudulent and requires verification, risk score +20.
4. **Blacklist Check:**
    - Transactions originating from blacklisted locations (e.g., countries known for high levels of fraud) automatically result in the card being blocked, regardless of transaction amount or other factors, risk score 100.
5. **Risk Score Calculation:**
    - Each transaction is assigned a **risk score** ranging from 0 to 100, based on various factors like transaction amount, time, frequency, and location changes. Higher scores indicate higher risk, helping to prioritize investigations.

Functionality 2: Flight Booking System

Description:

This booking system handles scenarios for flight reservations. It implements dynamic pricing based on seat availability, market demand, group discounts, and last-minute booking fees. Additionally, the system supports cancellations with varying refund policies and the use of reward points for frequent flyers.

Input Domain:

- **Number of Passengers:** int (e.g., 2, 5)
- **Booking Time:** LocalDateTime (e.g., 2024-10-01T12:00:00)
- **Departure Time:** LocalDateTime (e.g., 2024-10-02T12:00:00)
- **Seat Availability:** int (e.g., 100)
- **Current Price:** double (e.g., 500.00)
- **Previous Sales:** int (e.g., 50 seats sold)
- **Is Cancellation:** boolean (indicates if the booking is a cancellation)
- **Reward Points:** int (e.g., 5000 points)

Output:

- **Booking Confirmation:** boolean (indicates if the booking was successful)
- **Total Price:** double (final price after dynamic pricing, discounts, fees, and reward points)
- **Refund Amount:** double (refund amount in case of cancellation)
- **Points Used:** boolean (indicates if reward points were used in the booking)

Rules:

1. **Seat Availability:**
    - Bookings are only confirmed if there are enough seats available on the flight. If not enough seats are available, the booking is rejected.
2. **Dynamic Pricing:**
    - The way to calculate the final base price is based on the current price per number of passengers and a price factor, which is calculated as follows (previousSales / 100.0) * 0.8
3. **Last-Minute Fee:**
    - If a customer books a flight less than 24 hours before departure, a special last-minute fee of $100 is applied to the total fare.
4. **Group Discount:**
    - For group bookings of more than 4 passengers, a 5% discount is applied to the total fare.
5. **Reward Points Redemption:**
    - Frequent flyers can redeem reward points to reduce the price of their tickets. Each point reduces the fare by $0.01 (1 cent).
6. **Cancellation Policy:**
    - Cancellations made more than 48 hours before departure receive a full refund.
    - Cancellations made within 48 hours of departure receive only a 50% refund.

Functionality 3: Smart Energy Management System

Description:

The **Smart Energy Management System** controls energy usage in a smart home by managing devices based on current conditions such as energy prices, time of day, temperature, and usage limits. The system operates under several well-defined rules, which should be followed by students when creating tests. Below is a detailed breakdown of the rules governing the system.

Input Domain:

- **Real-Time Energy Price:** double (e.g., 0.15 per kWh)
- **Energy Price Threshold:** double (e.g., 0.20 per kWh)
- **Device Priorities:** Map<String, Integer> (e.g., "Heating" = 1, "Lights" = 2, "Appliances" = 3)
- **Current Time:** LocalDateTime (e.g., 2024-10-01T12:00:00)
- **Current Temperature:** double (e.g., 21.5 degrees Celsius)
- **Desired Temperature Range:** double[] (e.g., [20.0, 24.0] degrees Celsius)
- **Energy Usage Limits:** double (e.g., 30 kWh per day)
- **Total Energy Used Today:** double (e.g., 25 kWh)
- **Scheduled Devices:** List<DeviceSchedule> (e.g., [Device: "Oven", Time: 18:00])

Output:

- **Device Status:** Map<String, Boolean> (e.g., {"Heating"=true, "Lights"=false, "Appliances"=false})
- **Energy-Saving Mode:** boolean (indicates whether energy-saving mode is active)
- **Temperature Regulation:** boolean (indicates whether heating/cooling is active)
- **Total Energy Used:** double (updated total energy used)

Rules:

1. **Energy-Saving Mode Based on Price Threshold:** When the **current energy price** exceeds the **price threshold** set by the user, the system enters **energy-saving mode**. In this mode, the system turns off all low-priority devices and keeps high-priority devices active to conserve energy.
   Behavior:
   - Devices are classified by priority (with lower numbers representing higher priority).
   - All devices with a priority greater than 1 are considered low-priority and will be turned **off** during energy-saving mode.
   - Devices with a priority of 1 (high-priority) will remain **on**.

   Example:
   - **Input:** currentPrice = 0.25, priceThreshold = 0.20, devicePriorities = {"Heating"=1, "Lights"=2, "Appliances"=3}
   - **Output:** The system enters energy-saving mode, turning off low-priority devices ("Lights" and "Appliances"). "Heating" remains **on**.

2. **Night Mode (Between 11:00 PM and 6:00 AM):** From **11:00 PM** to **6:00 AM**, the system activates **night mode**. In this mode, only essential devices such as "Security" and "Refrigerator" remain **on**. All non-essential devices are turned **off** regardless of their priority.
   Behavior:
   - During night mode, devices other than "Security" and "Refrigerator" are turned **off**.

- Non-essential devices (e.g., "Lights", "Appliances", etc.) are turned **off** regardless of their priority.

Example:

- **Input:** currentTime = 23:30, devicePriorities = {"Security"=1, "Lights"=2, "Appliances"=3}
- **Output:** The system turns off "Lights" and "Appliances" while keeping "Security" **on**.

3. **Temperature Regulation:** The system regulates temperature by turning on the **heating** or **cooling** system based on the current indoor temperature relative to a user-defined range. Temperature regulation is **active** if the current temperature is outside the desired range.

   Behavior:

   - If the **current temperature** is **below** the minimum of the desired range, the system turns the **heating** system **on**.
   - If the **current temperature** is **above** the maximum of the desired range, the system turns the **cooling** system **on**.
   - If the temperature is within the desired range, both the heating and cooling systems are **off**.

   Example:

   - **Input:** currentTemperature = 18.0, desiredRange = [20.0, 24.0]
   - **Output:** The system turns on **heating** to raise the temperature.

4. **Energy Usage Limit:** If the **total energy usage** for the day is approaching or exceeding the **user-defined daily limit**, the system progressively turns off low-priority devices to stay within the limit. High-priority devices remain active as long as possible.

   Behavior:

   - The system monitors daily energy usage, and when **total energy used today** is **greater than or equal to the limit**, it starts turning off **low-priority devices**.
   - Devices with a priority greater than 1 are progressively turned off until energy consumption is reduced.
   - High-priority devices (priority 1) are turned off last, only when necessary.

   Example:

   - **Input:** totalEnergyUsedToday = 29, energyLimit = 30, devicePriorities = {"Heating"=1, "Lights"=2, "Appliances"=3}
   - **Output:** The system turns off "Lights" and "Appliances" to stay within the energy limit. "Heating" remains **on**.

5. **Scheduled Devices:** The system allows users to schedule devices to be turned **on** at specific times. These schedules are checked continuously, and the system turns on the specified devices when the scheduled time arrives.

   Behavior:

   - If the **current time** matches the **scheduled time** for a device, the system turns the device **on** regardless of its priority or energy-saving mode status.
   - The scheduled devices can override energy-saving and night modes when they are activated.

   Example:

   - **Input:** currentTime = 18:00, scheduledDevices = [{"Oven" at 18:00}]
   - **Output:** The system turns on "Oven" at the scheduled time of 18:00.