Company Management System

Description:

Create a Java program to simulate a simple company structure using **OOP principles**. The system will manage employees (managers and workers) and clients. It will include features for tracking employee working hours, vacation days, and sick days, and managing client spending over time. **Static variables** and methods will be used where appropriate.

Final static constants will be utilized for fixed values.

The project emphasizes encapsulation with **getters and setters only where needed**, and logical methods to modify the tracked data.

Class Structure and Improvements

Abstract Class: Person

The base class for all people in the company.

- **Attributes:**
  - static int idCounter: To generate unique IDs for all Person objects.
  - String name
  - String id (assigned using the static counter)
- **Methods:**
  - Constructor: Initializes name, age, and assigns a unique ID using the static counter.
  - Abstract: void displayInfo()
  - static void resetAllDays(ArrayList<Worker> workers): Resets all vacation and sick days for all workers (recommended using private helper methods.

Class: Client (Inherits Person)

Represents a client of the company.

- **Attributes:**
  - String companyName
  - double[] dailySpending (size 30): Tracks the client's spending for the past 30 days.
- **Methods:**
  - Override: void displayInfo()
  - void updateDailySpending(int day, double amount): Updates spending for a specific day.

Abstract Class: Worker (Inherits Person)

A base class for all workers.

- **Attributes:**
  - final static int START_VACATION_DAYS = 10 (initial vacation days for workers)
  - double[] dailyHours (size 30): Tracks working hours for each day in the past 30 days.
  - int vacationDays: Tracks remaining vacation days.
- **Methods:**
  - Abstract: double calculatePaycheck()
  - Override: void displayInfo()
  - void logHours(int day, double hours): Logs working hours for a specific day.
  - boolean takeVacationDays(int days): Deducts vacation days if sufficient.
  - Constructor: Initializes attributes and assigns starting vacation days.

Class: Manager (Inherits Worker)

Represents a manager in the company.

- **Attributes:**
  - final static int EXTRA_VACATION_DAYS = 10: Additional vacation days for managers.
  - ArrayList<Worker> team: Tracks the workers managed by the manager.
- **Methods:**
  - Override: double calculatePaycheck(): Includes base salary and bonus.
  - void addTeamMember(Worker worker): Adds a worker to the manager's team.
  - void displayTeam(): Displays all workers managed by the manager.

Class: RegularWorker (Inherits Worker)

Represents a regular worker.

- **Attributes:**
  - final static int START_SICK_DAYS = 15: Starting sick days for regular workers.
  - int sickDays: Tracks remaining sick days.
- **Methods:**
  - Override: double calculatePaycheck(): Calculates based on hours worked and hourly rate.
  - boolean takeSickDays(int days): Deducts sick days if sufficient.

Interface: `Payable`

Defines methods for salary-related operations.

- **Methods:**
  - `double calculatePaycheck() - monthly pay`

- `void displayPayDetails()- the details about the worker pey (hourly rate and bonuses)`

Key Features and Logic

1. **Automatic Unique IDs for Persons:**
   - Managed using a static counter in the Person class.
   - IDs are assigned automatically when a new Person is created.
2. **Static Methods:**
   - **Reset All Days:** The resetAllDays method in Person resets vacation and sick days for all workers, accounting for managers' extra vacation days.
3. **Dynamic Day Tracking:**
   - **Clients:** Track daily spending for the past 30 days with an array (dailySpending).
   - **Workers:** Track daily hours worked with an array (dailyHours).
4. **Vacation and Sick Days Management:**
   - Workers can take vacation or sick days, which decrement the respective counters.
   - Use boolean return types to indicate success or failure of requests (e.g., insufficient days).
5. **Encapsulation:**
   - Use getters and setters sparingly, only for attributes requiring external access or modification.
   - Emphasize attribute encapsulation to prevent direct access.

Suggested Methods Across Classes

- **Person Class:**
  - static void resetAllDays(ArrayList<Worker> workers): Resets vacation and sick days for all workers, accounting for managers' extra days.
- **Client Class:**
  - void updateDailySpending(int day, double amount): Adds spending for a specific day.
- **Worker Class:**
  - boolean takeVacationDays(int days): Deducts vacation days.
  - void logHours(int day, double hours): Logs hours worked for a specific day.
- **Manager Class:**
  - void addTeamMember(Worker worker): Adds a worker to the manager's team.
- **RegularWorker Class:**
  - boolean takeSickDays(int days): Deducts sick days.

- **Feel free to play with the methods however you want, keeping the same principles.**

Main class

1. Create objects for Client, Manager and RegularWorker.
2. Populate data (spending, working hours, vacation days…) using private helper methods with random values.
3. Use Scanner for user interaction, allowing the user to:
    - View data.
    - Perform operations (log hours, update spending, manage teams…).
4. Print results using appropriate class methods.

For those who wants it harder:

- Replace 30-day arrays with two-dimensional arrays for months and days, enabling year-long data tracking for clients' spending and workers' hours.
- Generate detailed reports for yearly totals, monthly breakdowns, and comparisons (e.g., top-performing workers, clients with the highest spending).
- Add a system where managers assign performance ratings to workers, which impact their yearly bonuses and total paycheck calculations.
- Create a system where different roles (e.g., Admin, Manager, Regular Worker) have specific permissions to access or modify certain data, ensuring proper security and functionality.
- Add to the current methods to check and give more money for extra hours. (8-10 hours gain another 125% on those hours, and 10-12 gives 150% on those hours, and above 12 its 200% for each hour).