جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

Attendance Management System project report

Submitted by

Name	ID	Class
Raghad Alqahtani	*2240001622*	Employee
Zainab Alhabib	2240004289	User
Lama Almutairi	*2240001380*	Manager
Layan Alabisi	*2240007790*	*AttendanceRecord*
Alya Shehab	*2240002062*	*Admin*
		AttendanceSystem

Introduction

Our project introduces an Attendance Management System designed to streamline employee attendance processes and enhance organizational efficiency. The system allows employees to clock in and out, provides managers with tools to track their team's attendance and approve leave requests, and enables administrators to manage users and generate detailed reports.

How it Works:

The code is structured into six main classes, each responsible for a specific functionality, ensuring clarity and organization:

1. **User Class:** Manages user details like name, ID, and role, along with login and logout functionalities.
2. **Employee Class:** Handles clock-in and clock-out processes and provides access to attendance records.
3. **Manager Class:** Oversees team management, leave approvals, and report generation.
4. **Admin Class:** Facilitates user management, including adding/removing users, generating monthly reports, and tracking attendance.
5. **Attendance System Class:** Manages the entire system, including attendance tracking and notification handling.
6. **Attendance Record Class:** Maintains timestamps for employee check-ins and check-outs and calculates working hours.

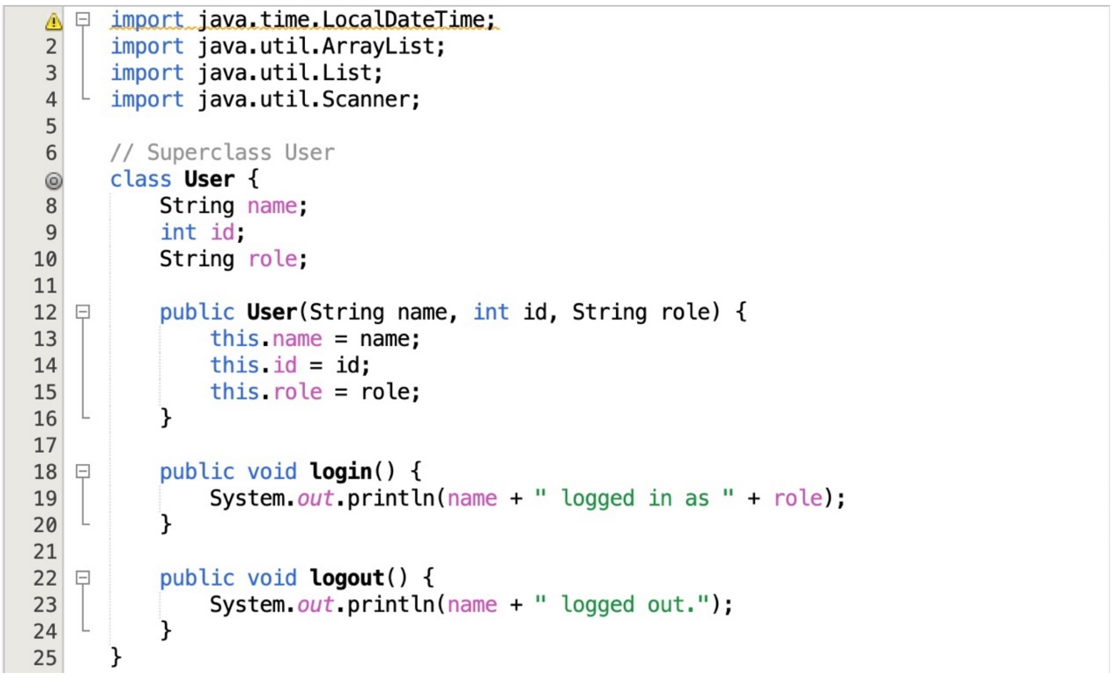
System Benefits:

- Simplifies employee attendance logging.
- Empowers managers with tools for effective team management.
- Provides administrators with a comprehensive overview for improved decision-making.

Objective

The primary goal of this project is to create a sophisticated electronic system for tracking employee attendance and departures using Java. This system is designed to streamline time management and boost overall work efficiency within organizations. By automating and precisely monitoring when employees clock in and out, administrators can effortlessly oversee schedules, generate detailed monthly reports, and ultimately enhance productivity across the board.

Code explanation



```java
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// Superclass User
class User {
    String name;
    int id;
    String role;

    public User(String name, int id, String role) {
        this.name = name;
        this.id = id;
        this.role = role;
    }

    public void login() {
        System.out.println(name + " logged in as " + role);
    }

    public void logout() {
        System.out.println(name + " logged out.");
    }
}
```

User Class

Attributes:

1. **String name:**
    - A variable to store the user's name.
2. **int id:**
    - A variable to store the user's unique identifier (ID)
3. **String role:**
    - A variable to store the user's role.

Constructor:
This is a constructor that initializes a new instance of the User class.

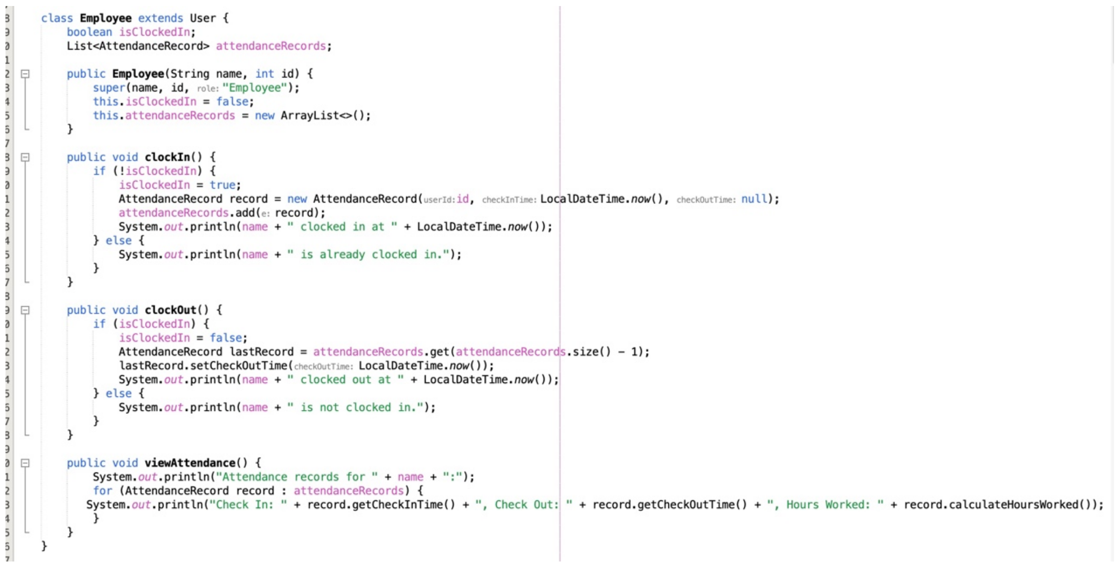
Parameters:

1. **String name:**
    - the name of the user.
2. **int id:**
    - the user's ID.
3. **String role:**
    - the user's role.

this keyword: Refers to the current object instance, differentiating between the class attributes and the parameters.

Methods:

1. **Login Method:**
    - This method simulates the user logging into the system.
    - It prints a message indicating the user's name and their role upon login.
2. **Logout Method:**
    - This method simulates the user logging out of the system.
    - It prints a message indicating that the user has logged out.



```java
class Employee extends User {
    boolean isClockedIn;
    List<AttendanceRecord> attendanceRecords;

    public Employee(String name, int id) {
        super(name, id, role: "Employee");
        this.isClockedIn = false;
        this.attendanceRecords = new ArrayList<>();
    }

    public void clockIn() {
        if (!isClockedIn) {
            isClockedIn = true;
            AttendanceRecord record = new AttendanceRecord(userId:id, checkInTime: LocalDateTime.now(), checkOutTime: null);
            attendanceRecords.add(e: record);
            System.out.println(name + " clocked in at " + LocalDateTime.now());
        } else {
            System.out.println(name + " is already clocked in.");
        }
    }

    public void clockOut() {
        if (isClockedIn) {
            isClockedIn = false;
            AttendanceRecord lastRecord = attendanceRecords.get(attendanceRecords.size() - 1);
            lastRecord.setCheckOutTime(checkOutTime: LocalDateTime.now());
            System.out.println(name + " clocked out at " + LocalDateTime.now());
        } else {
            System.out.println(name + " is not clocked in.");
        }
    }

    public void viewAttendance() {
        System.out.println("Attendance records for " + name + ":");
        for (AttendanceRecord record : attendanceRecords) {
       System.out.println("Check In: " + record.getCheckInTime() + ", Check Out: " + record.getCheckOutTime() + ", Hours Worked: " + record.calculateHoursWorked());
        }
    }
}
```

Employee Class

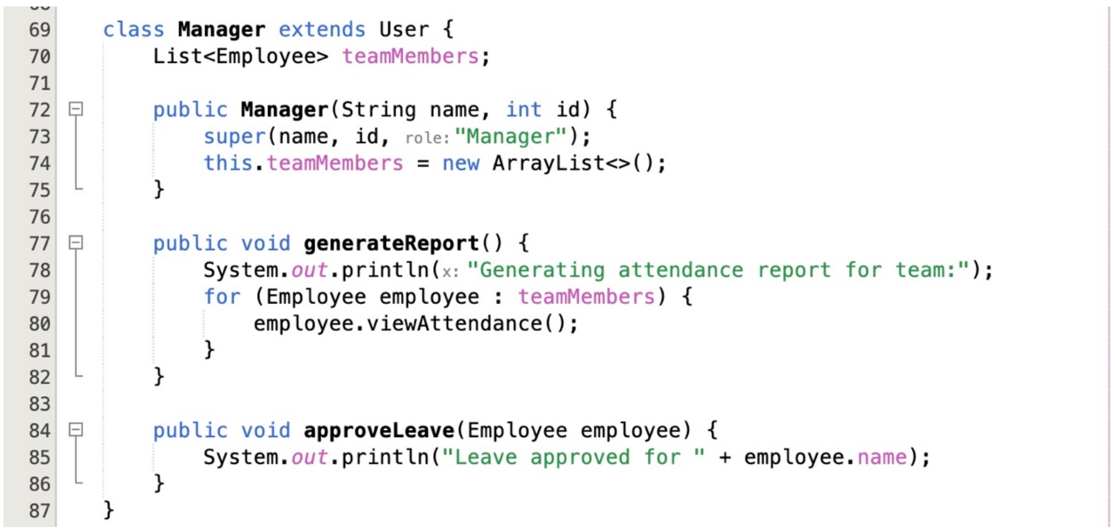
Attributes:

1. **isClockedIn:**
   - Type: Boolean
   - Description: Indicates whether the employee is currently clocked in.
2. **attendanceRecords:**
   - Type: List<AttendanceRecord>
   - Description: A list containing the attendance records of the employee.

Methods:

1. **Employee(String name, int id) (Constructor):**
   - Description: Initializes an Employee object with a name, ID, and default values (isClockedIn set to false and an empty list for attendanceRecords).
2. **clockIn():**
   - Clocks in the employee if they are not already clocked in.
   - Adds a new attendance record to the attendanceRecords list with the current time as the clock-in time.
3. **clockOut():**
   - Clocks out the employee if they are currently clocked in.
   - Updates the last attendance record in the list with the current time as the clock-out time.

4. **viewAttendance():**
    - Displays all attendance records for the employee, including the clock-in time, clock-out time, and the total hours worked.



```java
class Manager extends User {
    List<Employee> teamMembers;

    public Manager(String name, int id) {
        super(name, id, role: "Manager");
        this.teamMembers = new ArrayList<>();
    }

    public void generateReport() {
        System.out.println(x: "Generating attendance report for team:");
        for (Employee employee : teamMembers) {
            employee.viewAttendance();
        }
    }

    public void approveLeave(Employee employee) {
        System.out.println("Leave approved for " + employee.name);
    }
}
```

Manager class

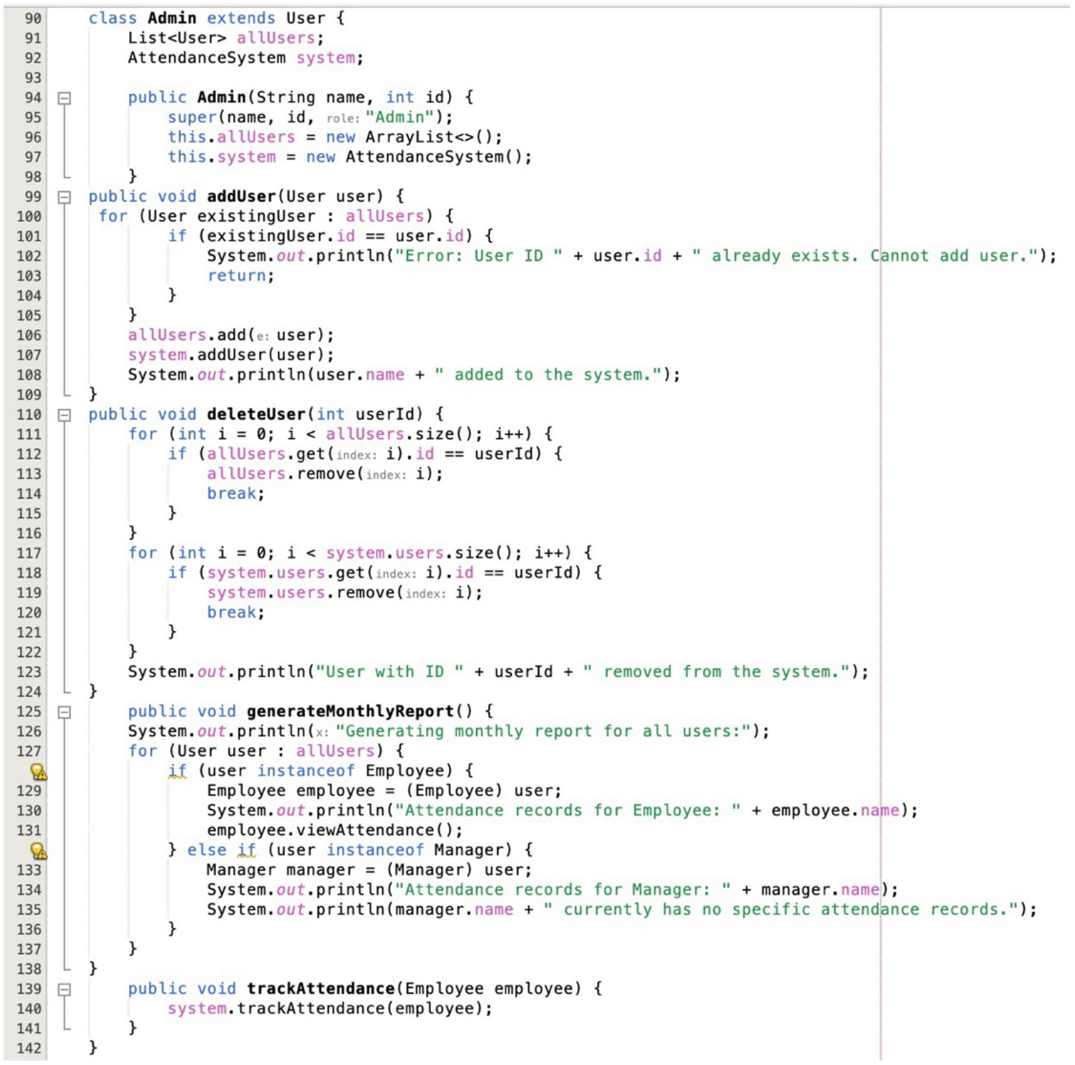
Attributes:

1. **teamMembers:**
    - A list of Employee objects representing the manager's team.

Methods:

1. **Constructor: Manager(String name, int id):**
    - Initializes a Manager object with a name, ID, and the role "Manager".
    - Starts with an empty list of team members.
2. **generateReport():**
    - Loops through the teamMembers list and calls each member's viewAttendance() method to display attendance details.
3. **approveLeave(Employee employee):**

- Prints a message approving leave for a specific employee.



```java
class Admin extends User {
    List<User> allUsers;
    AttendanceSystem system;

    public Admin(String name, int id) {
        super(name, id, role: "Admin");
        this.allUsers = new ArrayList<>();
        this.system = new AttendanceSystem();
    }
public void addUser(User user) {
 for (User existingUser : allUsers) {
        if (existingUser.id == user.id) {
            System.out.println("Error: User ID " + user.id + " already exists. Cannot add user.");
            return;
        }
    }
    allUsers.add(e: user);
    system.addUser(user);
    System.out.println(user.name + " added to the system.");
}
public void deleteUser(int userId) {
    for (int i = 0; i < allUsers.size(); i++) {
        if (allUsers.get(index: i).id == userId) {
            allUsers.remove(index: i);
            break;
        }
    }
    for (int i = 0; i < system.users.size(); i++) {
        if (system.users.get(index: i).id == userId) {
            system.users.remove(index: i);
            break;
        }
    }
    System.out.println("User with ID " + userId + " removed from the system.");
}
    public void generateMonthlyReport() {
    System.out.println(x: "Generating monthly report for all users:");
    for (User user : allUsers) {
        if (user instanceof Employee) {
            Employee employee = (Employee) user;
            System.out.println("Attendance records for Employee: " + employee.name);
            employee.viewAttendance();
        } else if (user instanceof Manager) {
            Manager manager = (Manager) user;
            System.out.println("Attendance records for Manager: " + manager.name);
            System.out.println(manager.name + " currently has no specific attendance records.");
        }
    }
}
    public void trackAttendance(Employee employee) {
        system.trackAttendance(employee);
    }
}
```

Admin Class

Attributes:

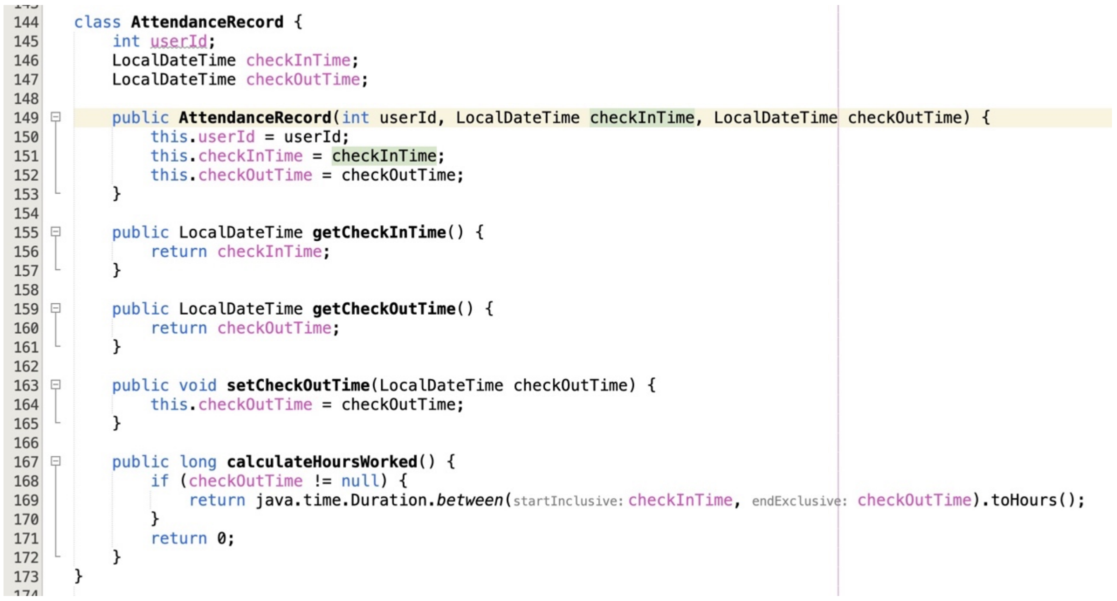
1. **List<User> allUsers:**
   - Stores all users in the system.
2. **AttendanceSystem system:**
   - An instance of the AttendanceSystem class to manage attendance.

Methods:

1. **addUser(User user):**

- Adds a new user to the allUsers list if the user ID is unique. Adds the user to the attendance system and prints a confirmation or error message.

2. **deleteUser(int userId):**
   - Removes a user from allUsers and system.users based on the user ID. Prints a message confirming the removal.
3. **generateMonthlyReport():**
   - Generates a monthly attendance report for all users, differentiating between employees and managers.
4. **trackAttendance(Employee employee):**
   - Calls the trackAttendance method from the AttendanceSystem for the specified employee.



```java
class AttendanceRecord {
    int userId;
    LocalDateTime checkInTime;
    LocalDateTime checkOutTime;

    public AttendanceRecord(int userId, LocalDateTime checkInTime, LocalDateTime checkOutTime) {
        this.userId = userId;
        this.checkInTime = checkInTime;
        this.checkOutTime = checkOutTime;
    }

    public LocalDateTime getCheckInTime() {
        return checkInTime;
    }

    public LocalDateTime getCheckOutTime() {
        return checkOutTime;
    }

    public void setCheckOutTime(LocalDateTime checkOutTime) {
        this.checkOutTime = checkOutTime;
    }

    public long calculateHoursWorked() {
        if (checkOutTime != null) {
            return java.time.Duration.between( startInclusive: checkInTime,  endExclusive: checkOutTime).toHours();
        }
        return 0;
    }
}
```

AttendanceRecord Class

Purpose: Represents a single attendance record for an employee.

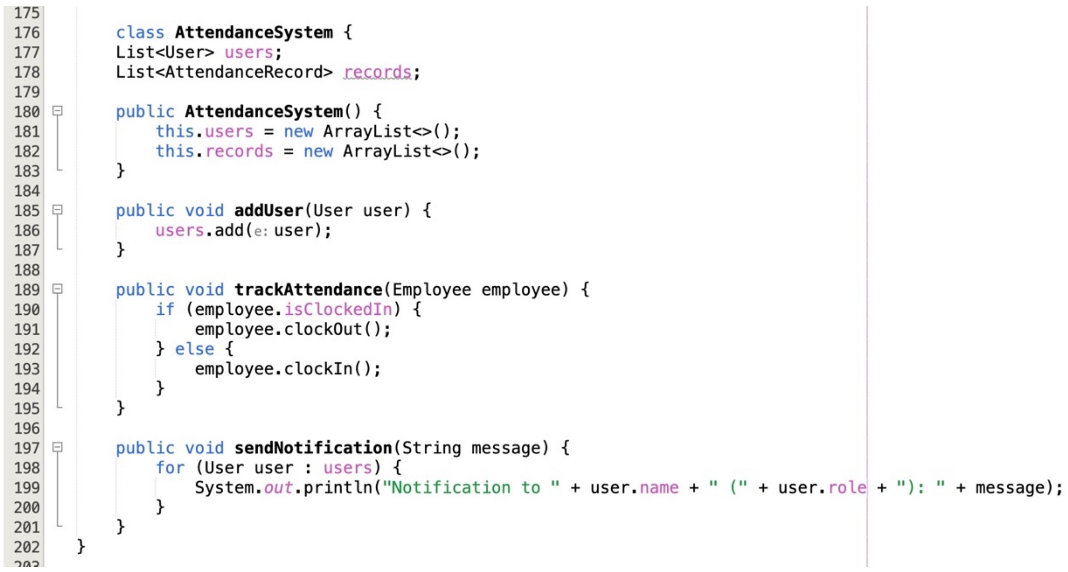
Attributes:

1. **int userId:**
   - The unique ID of the user associated with the attendance record.
2. **LocalDateTime checkInTime:**
   - The check-in timestamp for the user.

3. **LocalDateTime checkOutTime:**
    - The check-out timestamp for the user.

Methods:

1. **getCheckInTime():**
    - Returns the check-in timestamp for the user.
2. **getCheckOutTime():**
    - Returns the check-out timestamp for the user.
3. **setCheckOutTime(LocalDateTime checkOutTime):**
    - Updates the check-out timestamp for the user.
4. **calculateHoursWorked():**
    - Calculates the total hours worked between checkInTime and checkOutTime. If checkOutTime is not set, it returns 0.



```java
    class AttendanceSystem {
    List<User> users;
    List<AttendanceRecord> records;

    public AttendanceSystem() {
        this.users = new ArrayList<>();
        this.records = new ArrayList<>();
    }

    public void addUser(User user) {
        users.add(e: user);
    }

    public void trackAttendance(Employee employee) {
        if (employee.isClockedIn) {
            employee.clockOut();
        } else {
            employee.clockIn();
        }
    }

    public void sendNotification(String message) {
        for (User user : users) {
            System.out.println("Notification to " + user.name + " (" + user.role + "): " + message);
        }
    }
}
```

Attendance System Class

Purpose: The AttendanceSystem class tracks employee attendance by managing clock-in and clock-out actions while storing attendance records.

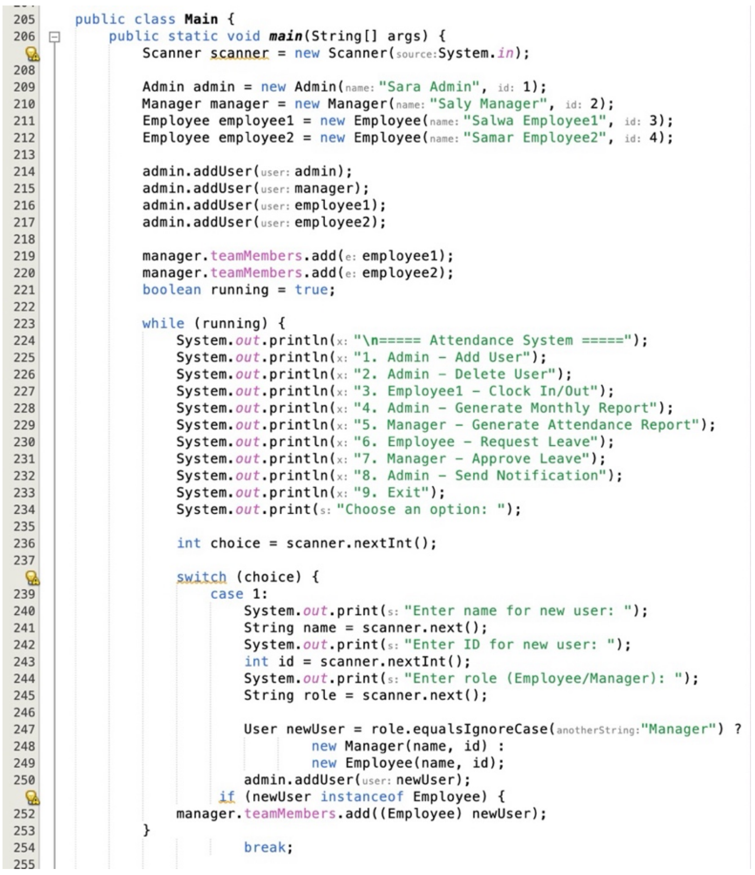
Attributes:

1. **isClockedIn:**
    - Boolean to track if the employee is clocked in.
2. **attendanceRecords:**

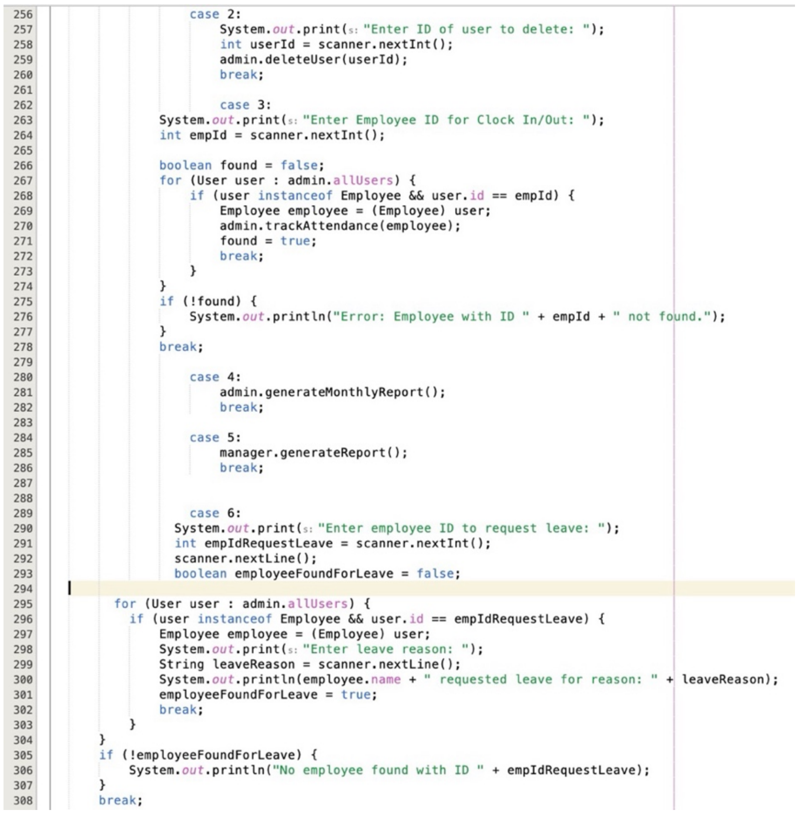
- List to store clock-in and clock-out timestamps.

Methods:

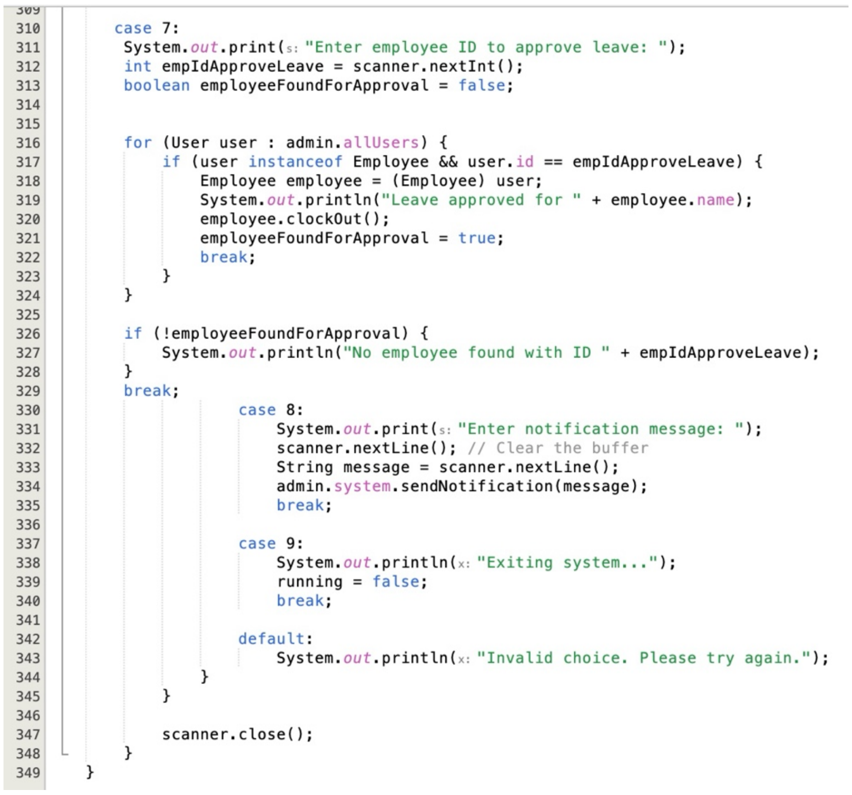
1. **AttendanceSystem():**
    - Constructor to initialize isClockedIn as false and attendanceRecords as empty.
2. **clockIn():**
    - Sets isClockedIn to true and logs the clock-in time.
    - Prints an error if already clocked in.
3. **clockOut():**
    - Sets isClockedIn to false and logs the clock-out time.
    - Prints an error if already clocked out.
4. **viewAttendance():**
    - Returns the attendance log.
5. **main():**
    - Demonstrates the system's functionality by clocking in, clocking out, handling errors, and printing records.



```java
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source:System.in);

        Admin admin = new Admin(name: "Sara Admin", id: 1);
        Manager manager = new Manager(name: "Saly Manager", id: 2);
        Employee employee1 = new Employee(name: "Salwa Employee1", id: 3);
        Employee employee2 = new Employee(name: "Samar Employee2", id: 4);

        admin.addUser(user: admin);
        admin.addUser(user: manager);
        admin.addUser(user: employee1);
        admin.addUser(user: employee2);

        manager.teamMembers.add(e: employee1);
        manager.teamMembers.add(e: employee2);
        boolean running = true;

        while (running) {
            System.out.println(x: "\n===== Attendance System =====");
            System.out.println(x: "1. Admin - Add User");
            System.out.println(x: "2. Admin - Delete User");
            System.out.println(x: "3. Employee1 - Clock In/Out");
            System.out.println(x: "4. Admin - Generate Monthly Report");
            System.out.println(x: "5. Manager - Generate Attendance Report");
            System.out.println(x: "6. Employee - Request Leave");
            System.out.println(x: "7. Manager - Approve Leave");
            System.out.println(x: "8. Admin - Send Notification");
            System.out.println(x: "9. Exit");
            System.out.print(s: "Choose an option: ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print(s: "Enter name for new user: ");
                    String name = scanner.next();
                    System.out.print(s: "Enter ID for new user: ");
                    int id = scanner.nextInt();
                    System.out.print(s: "Enter role (Employee/Manager): ");
                    String role = scanner.next();

                    User newUser = role.equalsIgnoreCase(anotherString:"Manager") ?
                            new Manager(name, id) :
                            new Employee(name, id);
                    admin.addUser(user: newUser);
                 if (newUser instanceof Employee) {
            manager.teamMembers.add((Employee) newUser);
        }
                    break;
```



```java
                case 2:
                    System.out.print("Enter ID of user to delete: ");
                    int userId = scanner.nextInt();
                    admin.deleteUser(userId);
                    break;

                    case 3:
        System.out.print("Enter Employee ID for Clock In/Out: ");
        int empId = scanner.nextInt();

        boolean found = false;
        for (User user : admin.allUsers) {
            if (user instanceof Employee && user.id == empId) {
                Employee employee = (Employee) user;
                admin.trackAttendance(employee);
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Error: Employee with ID " + empId + " not found.");
        }
        break;

            case 4:
                admin.generateMonthlyReport();
                break;

            case 5:
                manager.generateReport();
                break;


            case 6:
          System.out.print("Enter employee ID to request leave: ");
          int empIdRequestLeave = scanner.nextInt();
          scanner.nextLine();
          boolean employeeFoundForLeave = false;

  for (User user : admin.allUsers) {
    if (user instanceof Employee && user.id == empIdRequestLeave) {
        Employee employee = (Employee) user;
        System.out.print("Enter leave reason: ");
        String leaveReason = scanner.nextLine();
        System.out.println(employee.name + " requested leave for reason: " + leaveReason);
        employeeFoundForLeave = true;
        break;
    }
}
if (!employeeFoundForLeave) {
    System.out.println("No employee found with ID " + empIdRequestLeave);
}
break;
```



```java
   case 7:
    System.out.print("Enter employee ID to approve leave: ");
    int empIdApproveLeave = scanner.nextInt();
    boolean employeeFoundForApproval = false;


    for (User user : admin.allUsers) {
        if (user instanceof Employee && user.id == empIdApproveLeave) {
            Employee employee = (Employee) user;
            System.out.println("Leave approved for " + employee.name);
            employee.clockOut();
            employeeFoundForApproval = true;
            break;
        }
    }

    if (!employeeFoundForApproval) {
        System.out.println("No employee found with ID " + empIdApproveLeave);
    }
    break;
                case 8:
                    System.out.print("Enter notification message: ");
                    scanner.nextLine(); // Clear the buffer
                    String message = scanner.nextLine();
                    admin.system.sendNotification(message);
                    break;

                case 9:
                    System.out.println("Exiting system...");
                    running = false;
                    break;

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }

        scanner.close();
    }
}
```

Main Class:

Purpose: The Main class serves as the starting point of the program, where all operations are initiated. It demonstrates creating users, tracking attendance, and generating reports.

Attributes and Responsibilities:

1. **Attributes:**
    - List<User> allUsers: A list that stores all the users (employees and managers) for processing.

2. **Responsibilities:**
    - User Creation: Employees and managers are created and added to the system.

3. **Simulating Attendance:**
    - Employees log their check-in and check-out times to generate attendance records.

4. **Generating Monthly Reports:**
    - Attendance details for employees are printed, while a placeholder message is displayed for managers.

Key Functions:

1. **checkIn() and checkOut():**
    - Allow employees to log their work hours by storing timestamps.
2. **viewAttendance():**
    - Displays the employee's attendance records, including check-in and check-out times.
3. **calculateHoursWorked():**
    - Calculates total hours worked between check-in and check-out times.
4. **generateMonthlyReport():**
    - Iterates through all users and prints attendance records for employees or placeholders for managers.

Features:

Provides a menu with options to:

1. Add a user (Admin only)
2. Delete a user (Admin only)
3. Clock in/out (Employee)

4. Generate monthly report (Admin)
5. Generate attendance report for a team (Manager)
6. Request leave (Employee)
7. Approve leave (Manager)
8. Send notifications (Admin)
9. Exit the program.

Conclusion

The Attendance Management System presented is a significant step toward improving administrative efficiency and enhancing employee accountability. By distributing roles among administrators, managers, and employees, the system ensures seamless integration of attendance tracking and report generation, simplifying the monitoring of daily employee performance. With features such as precise clock-in and clock-out recording and leave management, the system effectively meets the needs of organizations, making it an essential tool for any workplace striving for innovation and structured management.

Reference

This project was built using knowledge from various programming resources, including Java programming textbooks, classroom material, and online documentation like javaTpoint.

https://www.javatpoint.com/java-tutorial

https://docs.oracle.com/javase/