```java
package use_case_controller;

import java.text.ParseException;

/**
 * Class Driver to launch the case management system.
 * @author Daria Vekic (Student ID: 586661)
 *
 */
public class Driver {

    /**
     * Method main to launch system
     * @param args
     * @throws ParseException thrown if date cannot be parsed
     */
    public static void main(String[] args) throws ParseException {
        launch();
    } //end method main

    /**
     * Method to start the case management system.
     * Calls methods to add pre-populated data to system.
     */
    static void launch() throws ParseException {
        //FileIO instance for reading/writing to file
        FileIO fileHandler = new FileIO();
        fileHandler.createPaths();
        //Declare and initialize data structures
        ArrayList<Case> cases = new ArrayList<Case>();
        ArrayList<Client> clients = new ArrayList<Client>();
        ArrayList<Employee> employees = new ArrayList<Employee>();
        fillEmpList(employees);
        fillCaseList(cases, employees);
        fillClientList(clients, cases);
        HashMap<String, String> loginsMap = new HashMap<String, String>();
        setAllPasswords(loginsMap); //unames and pwords to be written to file
        fileHandler.writeToFile(loginsMap);
        initializeView(employees, clients, cases); //launch the GUI
    } //end method initialiseData

    /**
     * Method to construct the interface for user to interact with.
     * @param employees the List of Employee objects
     * @param clients the List of Client objects
     * @param cases the List of Case objects
     */
    static void initializeView(ArrayList<Employee> employees, ArrayList<Client> clients, ArrayList<Case> cases) {
        LogInInterface login = new LogInInterface(employees, clients, cases);
    } //end method initializeView

    /**
     * Method to fill Map with login credentials required to access system.
     * Password can be updated by user.
     * @param loginsMap the Map to add login credentials to
     */
    static void setAllPasswords(HashMap<String, String> loginsMap){
        setMap(loginsMap, "wullyum.mcrae", "wullie123");
        setMap(loginsMap, "java.duke", "python123");
        setMap(loginsMap, "suzanne.gardener", "wullie321");
        setMap(loginsMap, "jason.dom", "XML!");
        setMap(loginsMap, "gerry.butler", "wullie");
    } //end method setAllPasswords

    /**
     * Method to control routine of adding a single login credentials to the Map.
     * Generates a salt value to add to hash value of password.
     * @param loginsMap the Map to add the login credential to.
     * @param username the Key in the map.
     * @param password the message to hash and store as Value in the map.
     */
    static void setMap(HashMap<String, String> loginsMap, String username, String password) {
        String salt = BCrypt.gensalt(10);
        String hash = BCrypt.hashpw(password, salt);
        loginsMap.put(username, hash);
    } //end method setInitialPasswords

    /**
     * Method to statically fill a List with Employee objects
     * @param employees the List to add elements to
     * @throws ParseException thrown if date of birth cannot be parsed
     */
    static void fillEmpList(ArrayList<Employee> employees) throws ParseException {
        employees.add(new Employee("Wullyum", "McRae", "10/11/1972", "The Big Hoose", "Falkirk", "FK2 9AD", "01324 403000"));
        employees.add(new Employee("Java", "Duke", "01/04/1972", "The Bigger Hoose", "Falkirk", "FK2 9AD", "01324 403001"));
        employees.add(new Employee("Suzanne", "Gardener", "06/03/1984", "Loggie House, 5A Upper Glen Road", "Bridge of Allan", "FK9 4PX", 
"07700015911"));
        employees.add(new Employee("Gerry", "Butler", "13/11/1969", "Insch, Burnside Road", "Whitecraigs", "G46 6TT", "07753463113"));
        employees.add(new Employee("Jason", "Dom", "26/07/1967", "Hillpark House", "Bridge of Allan", "FK9 4EE", "07700050034"));
    } //end method fillEmpList

    /**
     * Method to statically fill a List with Client objects
     * @param clients the List to add elements to
     * @param cases the List required to link a Client to a Case
     * @throws ParseException thrown if date of birth cannot be parsed
     */
    static void fillClientList(ArrayList<Client> clients, ArrayList<Case> cases) throws ParseException {
        clients.add(new Client("John", "Doe", "12/05/1979", "Somewhere", "Only We Know", "FK10 4DA", "01234567891", cases.get(0)));
        clients.add(new Client("The", "Batman", "11/04/1978", "Doon the Toon", "Clacks, Scotland", "FK10 2AR", "01259213989", cases.get(1)));
        clients.add(new Client("Dan", "Bruce", "27/02/2001", "A Very Nice House", "Alloa, Scotland", "ZE1 0BA", "07753463113", cases.get(2)));
        clients.add(new Client("Twitchy", "Twitch", "30/03/1939", "Cell 1a", "HMP Polmont", "FK2 0AB", "01324 711558", cases.get(3)));
        clients.add(new Client("Rishi", "Sunak", "13/06/1994", "Flat 30G, High Street", "Govan, Glasgow", "PA3 4BF", "08009777766", cases.get
(4)));
    } //end method fillClientList

    /**
     * Method to statically fill a List with Case objects
     * @param cases the List to add elements to
     * @param employees the List required to link a Case to an Employee
     */
    static void fillCaseList(ArrayList<Case> cases, ArrayList<Employee> employees) {
        cases.add(new Case("Doe v Smith", CaseType.CRI, "Accused of theft", employees.get(0)));
        cases.add(new Case("Batman v Superman", CaseType.CRI, "Accused of theft", employees.get(1)));
        cases.add(new Case("Bruce v Wayne", CaseType.PIN, "Industrial accident", employees.get(2)));
        cases.add(new Case("Twitch v McCool", CaseType.CRI, "Accused of assault", employees.get(3)));
        cases.add(new Case("Sunak v Johnson", CaseType.IMM, "Home Office appeal", employees.get(4)));
    } //end method fillCaseList
} //end class Driver
```