```
import java.util.Random; import java.util.Scanner;

// Employee class with parameterized constructor class Employee { private String
firstName; private String lastName; private String department;

// Parameterized constructor
public Employee(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.department = determineDepartment();
}

// Getters
public String getFirstName() {
    return firstName;
}

public String getLastName() {
    return lastName;
}

public String getDepartment() {
    return department;
}

// Method to determine department based on user input
private String determineDepartment() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("\nSelect Department:");
    System.out.println("1. Technical");
    System.out.println("2. Admin");
    System.out.println("3. Human Resource");
    System.out.println("4. Legal");
    System.out.print("Enter your choice (1-4): ");

    int choice = scanner.nextInt();

    switch(choice) {
        case 1: return "tech";
        case 2: return "admin";
        case 3: return "hr";
        case 4: return "legal";
        default:
            System.out.println("Invalid choice! Defaulting to
Technical department.");
            return "tech";
    }
}


}
```

// CredentialService class with required methods class CredentialService { private static final String COMPANY_DOMAIN = "abc.com"; private Random random;

```
public CredentialService() {
    this.random = new Random();
}

// Method to generate email address
public String generateEmailAddress(Employee employee) {
    String firstName = employee.getFirstName().toLowerCase();
    String lastName = employee.getLastName().toLowerCase();
    String department = employee.getDepartment();

    return firstName + lastName + "@" + department + "." +
COMPANY_DOMAIN;
}

// Method to generate random password
public String generatePassword() {
    String numbers = "0123456789";
    String capitalLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    String smallLetters = "abcdefghijklmnopqrstuvwxyz";
    String specialCharacters = "!@#$%^&*";

    StringBuilder password = new StringBuilder();

    // Ensure at least one character from each category
    password.append(numbers.charAt(random.nextInt(numbers.length())));

password.append(capitalLetters.charAt(random.nextInt(capitalLetters.le
ngth())));

password.append(smallLetters.charAt(random.nextInt(smallLetters.length
())));

password.append(specialCharacters.charAt(random.nextInt(specialCharact
ers.length())));

    // Generate remaining 4 characters randomly from all categories
    String allCharacters = numbers + capitalLetters + smallLetters +
specialCharacters;
    for (int i = 0; i < 4; i++) {

password.append(allCharacters.charAt(random.nextInt(allCharacters.leng
th())));
    }

    // Shuffle the password to randomize positions
    return shuffleString(password.toString());
}

// Helper method to shuffle string characters
private String shuffleString(String str) {
    char[] chars = str.toCharArray();
    for (int i = chars.length - 1; i > 0; i--) {
        int j = random.nextInt(i + 1);
        char temp = chars[i];
        chars[i] = chars[j];
        chars[j] = temp;
    }
    return new String(chars);
}

// Method to display credentials
public void showCredentials(Employee employee, String email, String
password) {
    System.out.println("\n" + "=".repeat(50));
    System.out.println("GENERATED CREDENTIALS");
    System.out.println("=".repeat(50));
    System.out.println("Dear " + employee.getFirstName() + " your 
generated credentials are as follows");
    System.out.println("Email ---> " + email);
    System.out.println("Password ---> " + password);
    System.out.println("=".repeat(50));
}


}
```

// Main class to test the application public class EmployeeCredentialSystem { public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

```
   System.out.println("=".repeat(50));
    System.out.println("EMPLOYEE CREDENTIAL GENERATION SYSTEM");
    System.out.println("=".repeat(50));

    // Get employee details
    System.out.print("Enter First Name: ");
    String firstName = scanner.nextLine().trim();

    System.out.print("Enter Last Name: ");
    String lastName = scanner.nextLine().trim();

    // Validate input
    if (firstName.isEmpty() || lastName.isEmpty()) {
        System.out.println("Error: First name and last name cannot be 
empty!");
        return;
    }

    // Create employee object using parameterized constructor
    Employee employee = new Employee(firstName, lastName);

    // Create credential service
    CredentialService credentialService = new CredentialService();
    // Generate credentials
    String email = credentialService.generateEmailAddress(employee);
    String password = credentialService.generatePassword();

    // Display credentials
    credentialService.showCredentials(employee, email, password);

    // Additional test cases
    System.out.println("\nWould you like to generate credentials for
another employee? (y/n): ");
    String choice = scanner.nextLine().trim().toLowerCase();

    if (choice.equals("y") || choice.equals("yes")) {
        main(args); // Recursive call for another employee
    } else {
        System.out.println("Thank you for using Employee Credential
Generation System!");
    }

    scanner.close();
}

// Method to run automated test cases
public static void runTestCases() {
    System.out.println("\n" + "=".repeat(60));
    System.out.println("RUNNING AUTOMATED TEST CASES");
    System.out.println("=".repeat(60));

    CredentialService service = new CredentialService();

    // Test Case 1: Technical Department
    Employee emp1 = new Employee("John", "Doe") {
        @Override
        public String getDepartment() { return "tech"; }
    };
    testEmployee(service, emp1, "Technical");

    // Test Case 2: Admin Department
    Employee emp2 = new Employee("Jane", "Smith") {
        @Override
        public String getDepartment() { return "admin"; }
    };
    testEmployee(service, emp2, "Admin");

    // Test Case 3: HR Department
    Employee emp3 = new Employee("Mike", "Johnson") {
        @Override
        public String getDepartment() { return "hr"; }
    };
    testEmployee(service, emp3, "Human Resource");

    // Test Case 4: Legal Department
    Employee emp4 = new Employee("Sarah", "Wilson") {
        @Override
        public String getDepartment() { return "legal"; }
    };
    testEmployee(service, emp4, "Legal");

    // Test password validation
    testPasswordValidation(service);
}

private static void testEmployee(CredentialService service, Employee
emp, String deptName) {
    String email = service.generateEmailAddress(emp);
    String password = service.generatePassword();

    System.out.println("\nTest Case - " + deptName + " Department:");
    System.out.println("Employee: " + emp.getFirstName() + " " +
emp.getLastName());
    System.out.println("Email: " + email);
    System.out.println("Password: " + password);
    System.out.println("Email Format Valid: " +
validateEmailFormat(email, emp));
    System.out.println("Password Format Valid: " +
validatePasswordFormat(password));
}
```

```
private static boolean validateEmailFormat(String email, Employee emp)
{
    String expectedPattern = emp.getFirstName().toLowerCase() +
emp.getLastName().toLowerCase() +
                           "@" + emp.getDepartment() + ".abc.com";
    return email.equals(expectedPattern);
}

private static boolean validatePasswordFormat(String password) {
    boolean hasNumber = password.matches(".*\\d.*");
    boolean hasCapital = password.matches(".*[A-Z].*");
    boolean hasSmall = password.matches(".*[a-z].*");
    boolean hasSpecial = password.matches(".*[!@#$%^&*].*");
    boolean isCorrectLength = password.length() == 8;

    return hasNumber && hasCapital && hasSmall && hasSpecial &&
isCorrectLength;
}

private static void testPasswordValidation(CredentialService service)
{
    System.out.println("\nPassword Validation Test:");
    for (int i = 0; i < 5; i++) {
        String password = service.generatePassword();
        System.out.println("Password " + (i+1) + ": " + password +
                         " - Valid: " +
validatePasswordFormat(password));
    }
}


}
```