Here's the Java implementation based on your requirements:

AuthenticationInfo.java:

Java

```
public class AuthenticationInfo {
    private String url;
    private String username;
    private String password;

    public AuthenticationInfo(String url, String username, String password) {
        this.url = url;
        this.username = username;
        this.password = password;

    }

    // Getters and setters
    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public
 String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;

    }
}
```

Use code with caution.

EncryptionUtil.java:

Java

```
public class EncryptionUtil {
    public static String encrypt(String password, int shift) {
        StringBuilder encryptedPassword = new StringBuilder();
        for (char c : password.toCharArray()) {
            encryptedPassword.append((char) ((c - 'a' + shift) % 26 + 'a'));
        }
        return encryptedPassword.toString();
    }

    public static String decrypt(String encryptedPassword, int shift) {
        StringBuilder decryptedPassword = new StringBuilder();
        for (char c : encryptedPassword.toCharArray()) {
            decryptedPassword.append((char)
 ((c - 'a' - shift + 26) % 26 + 'a'));
        }
        return decryptedPassword.toString();
    }
}
```

Use code with caution.

Application.java:

Java

```
import java.util.ArrayList;
import java.util.Scanner;

public class Application {
    private ArrayList<AuthenticationInfo> loginInfos = new ArrayList<>();

    public void start() {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("1. New Login Info");
            System.out.println("2. View Login Info");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    createLoginRecord(scanner);
                    break;
```

```
                case 2:
                    viewLoginRecord(scanner);
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        } while (choice != 3);
    }

    private void createLoginRecord(Scanner scanner) {
        System.out.print("Enter URL: ");
        String url = scanner.next();
        System.out.print("Enter username: ");
        String username = scanner.next();
        System.out.print("Enter password: ");
        String password = scanner.next();


        int shift = password.length() / 2; // Adjust shift as needed
        String encryptedPassword = EncryptionUtil.encrypt(password, shift);

        AuthenticationInfo info = new AuthenticationInfo(url, username, encryptedPassword);
        loginInfos.add(info);
        System.out.println("Login information added successfully.");
    }

    private void viewLoginRecord(Scanner scanner) {
        System.out.print("Enter URL to view: ");
        String urlToView = scanner.next();

        for (AuthenticationInfo info : loginInfos) {
            if (info.getUrl().equals(urlToView)) {
                int shift = info.getPassword().length() / 2; // Adjust shift as needed
                String decryptedPassword = EncryptionUtil.decrypt(info.getPassword(), shift)
                System.out.println("Username: " + info.getUsername());
                System.out.println("Password: " + decryptedPassword);
                return;
            }
        }

        System.out.println("Login information not found.");
    }

    public static void main(String[] args) {
        Application app = new Application();
        app.start();
    }
}
```

Use code with caution.

Note:

- The encryption and decryption methods use a simple Caesar cipher with a shift based on the password length. You can replace this with a more robust encryption algorithm like AES or RSA.
- The shift value can be adjusted to improve security.
- Consider adding error handling and input validation to make the application more robust.
- For real-world applications, using a secure password manager library is recommended.