Ben Brandhorst

SDEV 325 6380

Detecting Software Vulnerabilities

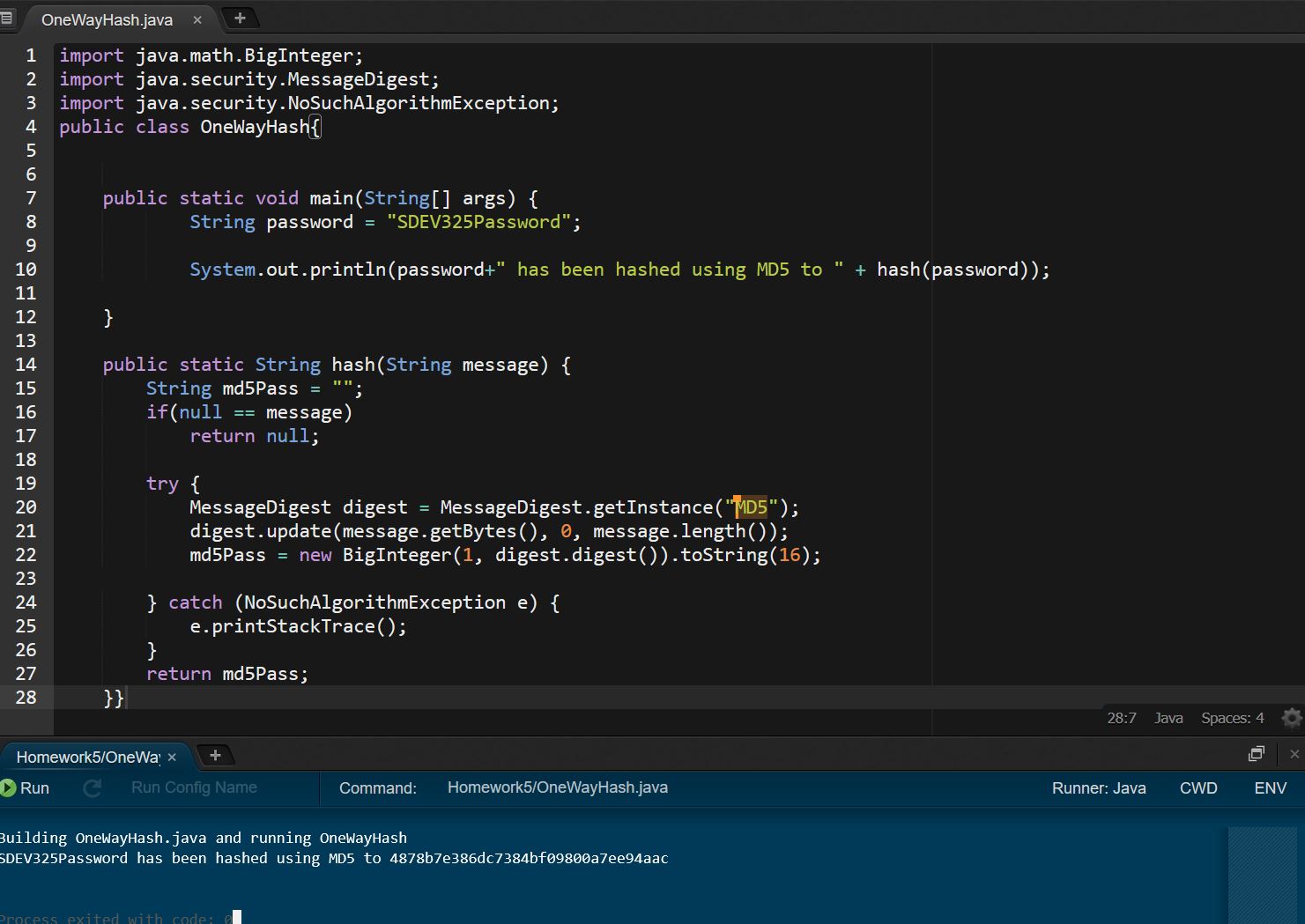
Professor Luke Donoho

10/12/2019

**Demonstrating Porous Defenses**

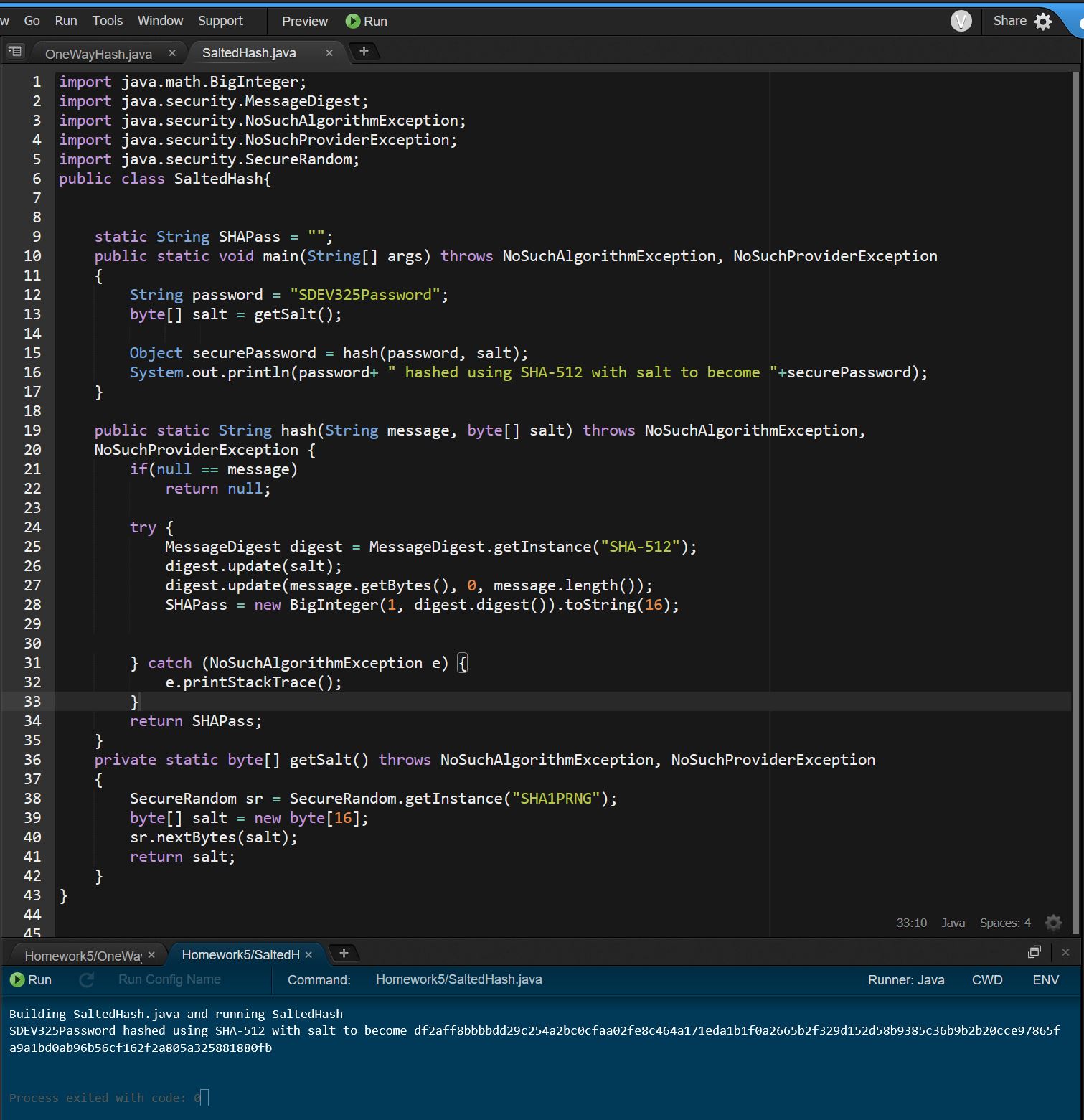
**Example #1**

**Use of a One-Way Hash without a Salt**

****

In the above example, we can an example of a password being hashed using the MD5 message-digest algorithm. This produces a 128-bit hash value. However, there are vulnerabilities with both the hashing algorithm and the fact that there is not a salt included with the hash. This increases the potential for an attacker to be able to reverse the hash and discover the password. Additionally, without the use of a salt, if multiple users pick the same password, the hash values will return the same.

**Updated Example Including Salted Hash**

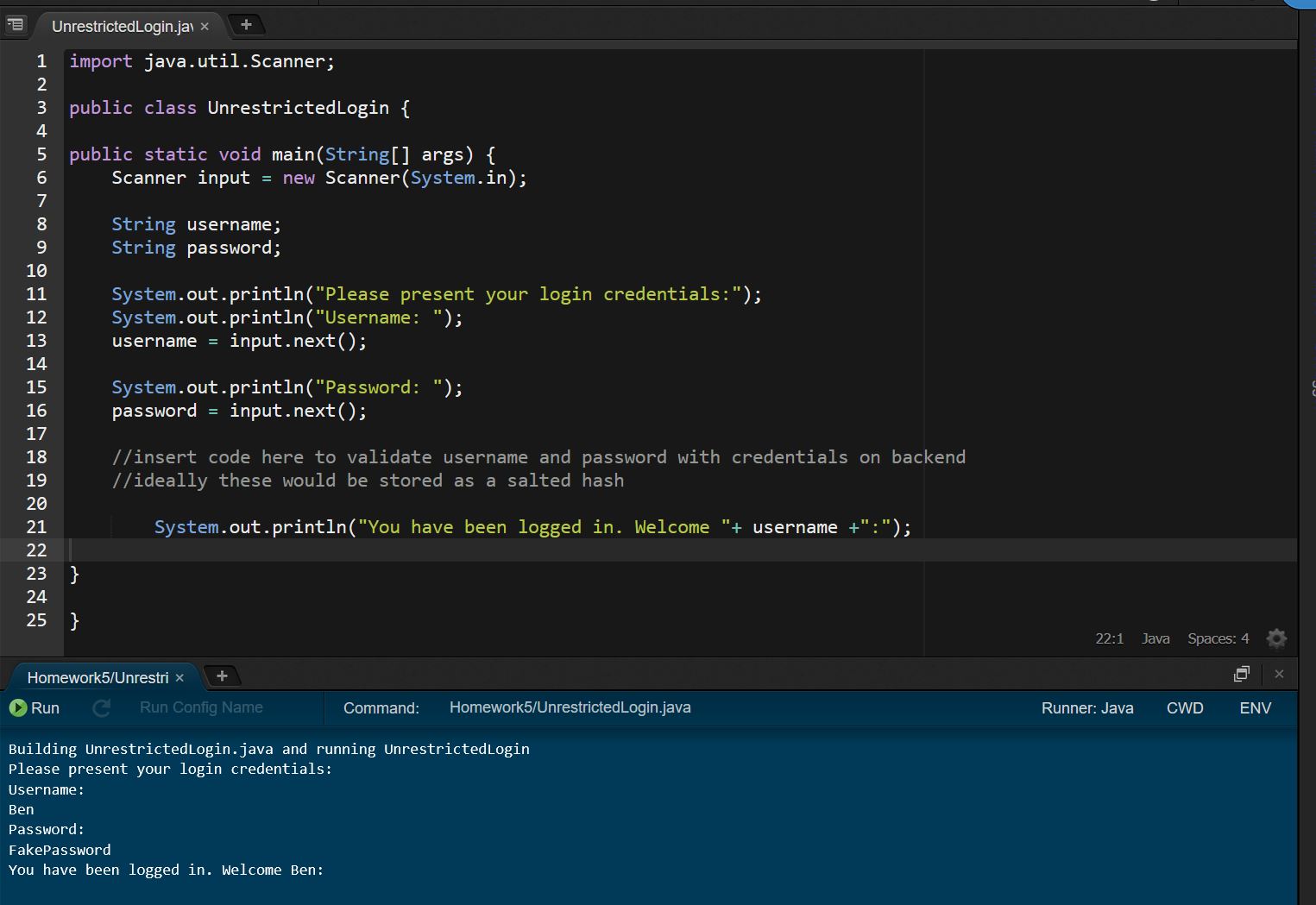
****

In the above example, we’ve updated the hashing algorithm to SHA-512 which is a 512-bit algorithm that is much more difficult to crack compared to the MD5’s 128 bit. Additionally, we have created a salt that is randomly generated and included with the password hash. This allows the password hash to remain unique even if another user uses the exact same password.

**Example #2**

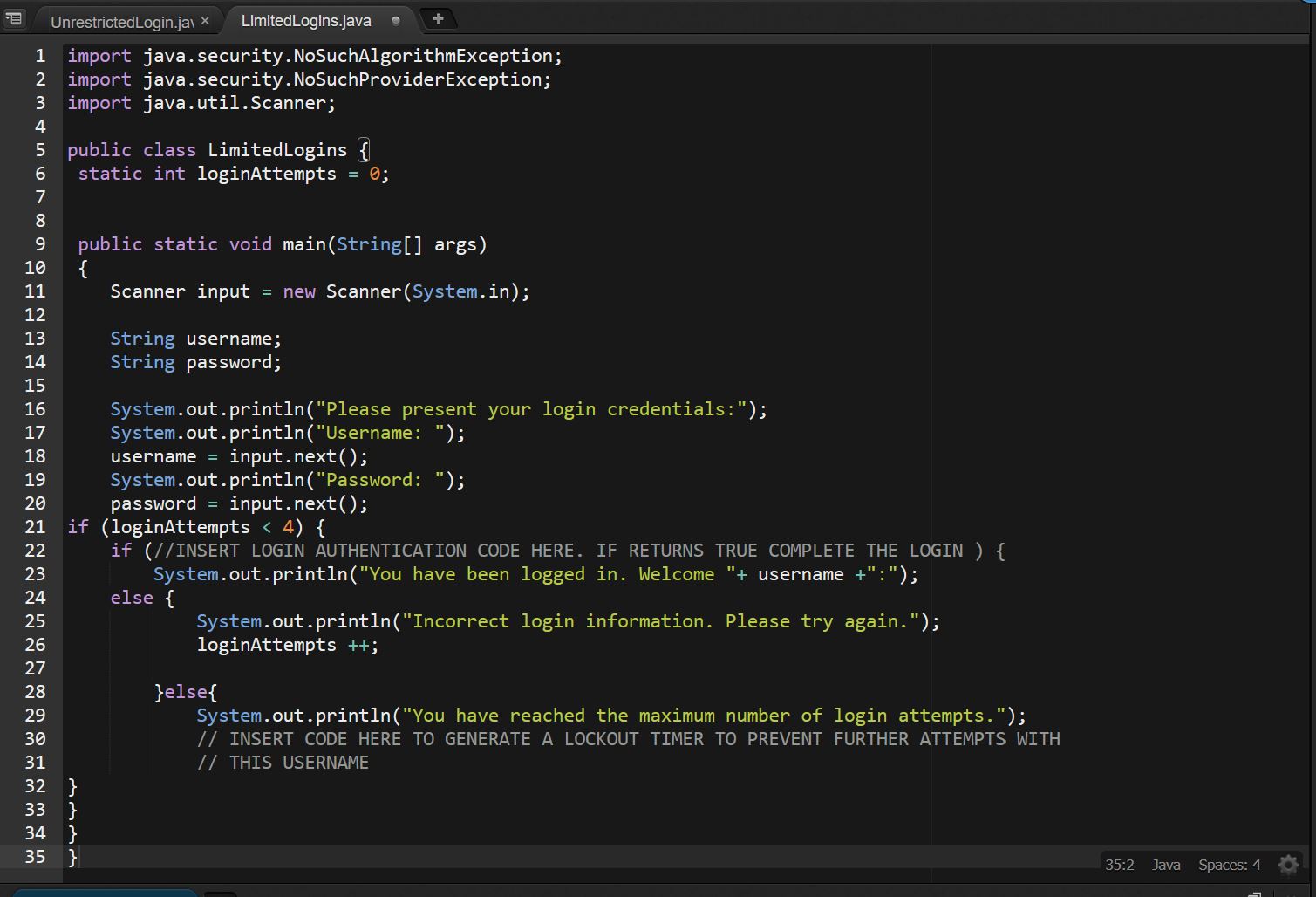
**Improper Restriction of Excessive Authentication**

**Attempts**



This is an example of a simple login using Java. The example assumes there exists working code to connect to a database to retrieve the stored login credentials. Additionally, for simplicity’s sake this example is not using hashes or salt as that was covered in the previous example.   
  
We can see from this example that there are no restrictions on the number of login attempts which leaves it vulnerable to dictionary, brute force, and other attacks.

**Restricted Login Attempts**

****

Here we’ve used a simple loop to calculate the number of times a user has attempted to login. If the incorrect credentials are used too many times, the loop breaks and the user is informed that they’ve reached the maximum number of login attempts. If the connection with the database were more developed, I’d also include a lockout timer that restricts the login with that username for a specified period of time.