OOSE Lab Week 11

Contents

[1 Create a Unit Test 1](#_Toc89175208)

Week 11

# Create a Unit Test

Choose a single class that you would like to unit test.

Using NetBeans or otherwise create a unit test for this class.

Class: AuthenticationService.java

package org.example;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class AuthenticationService {  
  
 private Map<String, String> userCredentials;  
  
 public AuthenticationService() {  
 this.userCredentials = new HashMap<>();  
 }  
  
 public boolean login(String username, String password) {  
 if (userCredentials.containsKey(username)) {  
 String storedPassword = userCredentials.get(username);  
 return storedPassword.equals(password);  
 }  
 displayWrongDetails();  
 return false;  
 }  
  
 public void createAccount(String username, String password) {  
 if (!userCredentials.containsKey(username)) {  
 userCredentials.put(username, password);  
 }  
 }  
  
 public boolean verifyPassword(String username, String password) {  
 if (userCredentials.containsKey(username)) {  
 String storedPassword = userCredentials.get(username);  
 return storedPassword.equals(password);  
 }  
 return false;  
 }  
  
 public void displayWrongDetails() {  
 System.*out*.println("Invalid username or password. Please try again.");  
 }  
}

Tests: AuthenticationServiceTest

import org.example.AuthenticationService;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class AuthenticationServiceTest {  
  
 @Test  
 public void testLoginWithValidCredentials() {  
 AuthenticationService authService = new AuthenticationService();  
 authService.createAccount("testUser", "password123");  
  
 boolean result = authService.login("testUser", "password123");  
  
 *assertTrue*(result);  
 }  
  
 @Test  
 public void testLoginWithInvalidCredentials() {  
 AuthenticationService authService = new AuthenticationService();  
 authService.createAccount("testUser", "password123");  
  
 boolean result = authService.login("testUser", "wrongPassword");  
  
 *assertFalse*(result);  
 }  
 @Test  
 public void testCreateAccount() {  
 AuthenticationService authService = new AuthenticationService();  
  
 authService.createAccount("newUser", "newPassword");  
  
 boolean result = authService.login("newUser", "newPassword");  
  
 *assertTrue*(result);  
 }  
  
 @Test  
 public void testCreateAccountWithExistingUsername() {  
 AuthenticationService authService = new AuthenticationService();  
 authService.createAccount("existingUser", "password123");  
  
 *assertThrows*(IllegalArgumentException.class, () -> authService.createAccount("existingUser", "newPassword"));  
 }  
  
  
 @Test  
 public void testVerifyPasswordWithCorrectPassword() {  
 AuthenticationService authService = new AuthenticationService();  
 authService.createAccount("user1", "password123");  
  
 boolean result = authService.verifyPassword("user1", "password123");  
  
 *assertTrue*(result);  
 }  
  
 @Test  
 public void testVerifyPasswordWithIncorrectPassword() {  
 AuthenticationService authService = new AuthenticationService();  
 authService.createAccount("user2", "password123");  
  
 boolean result = authService.verifyPassword("user2", "wrongPassword");  
  
 *assertFalse*(result);  
 }  
}