**Question 1**

1a) class RoomSafe

/\*\*

\* Answer to TMA01 Question 1.

\*

\* @author Wayne Sandford

\* @version 15th October 2018

\*/

public class RoomSafe

{

/\* instance variable \*/

private String password;

/\*\*

\* Constructor for objects of class RoomSafe

\*/

public RoomSafe()

{

}

}

1b)

public class RoomSafe

{

/\* instance variable \*/

private String password

/\*\*

\* Constructor for objects of class RoomSafe

\*/

public RoomSafe()

{

password = "Adminadmin1";

}

/\* instance methods \*/

/\*\*

\* Returns the password of the receiver.

\*/

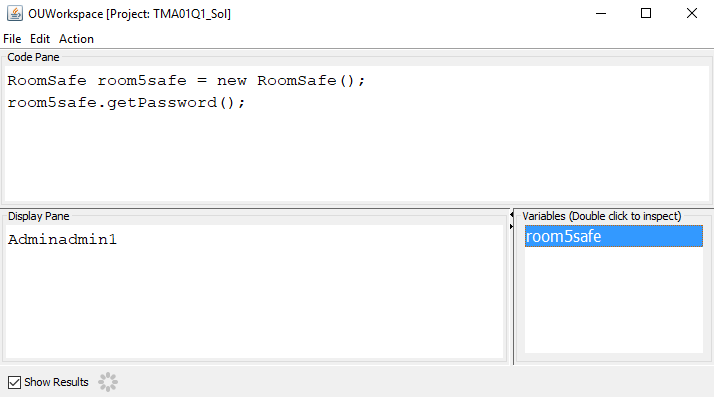
public String getPassword()

{

return this.password;

}

}

OUWorkspace output

1c) i) isValidLength(String pw) instance method

/\*\*

\* Return true if the argument pw is at least 8 characters long.

\*/

public boolean isValidLength(String pw)

{

return pw.length() >= 8;

}

1c) ii)

The charAt() method takes an integer value as its argument. It uses this value as an index. The method returns the specified character of the string determined by the index value.

1c) iii) hasUpperCase(String pw) instance method

/\*\*

\* Return true if at least one of the characters in the argument

\* pw is uppercase otherwise return false

\*/

public boolean hasUpperCase(String pw)

{ // Assume initially that the string contains no uppercase characters.

boolean result = false;

// Examine each character of pw in turn.

for (int i = 0; i < pw.length(); i++)

{

// If the character at position i is uppercase,

// set result to true

if (Character.isUpperCase(pw.charAt(i)))

{

result = true;

}

}

return result;

}

1c) iv) isValidPassword(String pw) instance method

/\*\*

\* Return true if password passes all 3 tests.

\* - At least 8 characters.

\* - Has at least one digit.

\* - Has at least one uppercase character.

\*/

public boolean isValidPassword(String pw)

{

return isValidLength(pw) && hasDigit(pw) && hasUpperCase(pw);

}

1d) setPassword(String pw) instance method

/\*\*

\* Sets the password of the receiver to the value

\* of the argument pw only if pw is valid. Displays message

\* acknowledging password is valid or not valid.

\*/

public void setPassword(String pw)

{

if(isValidPassword(pw))

{

System.out.println("The password " + pw + " is valid.");

this.password = pw;

}

else

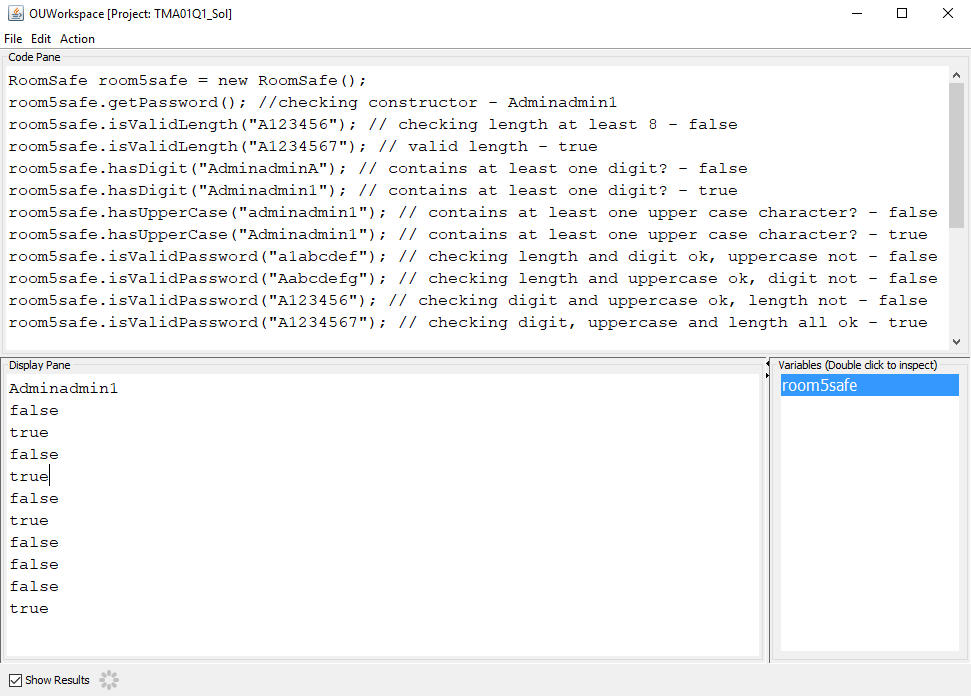
{

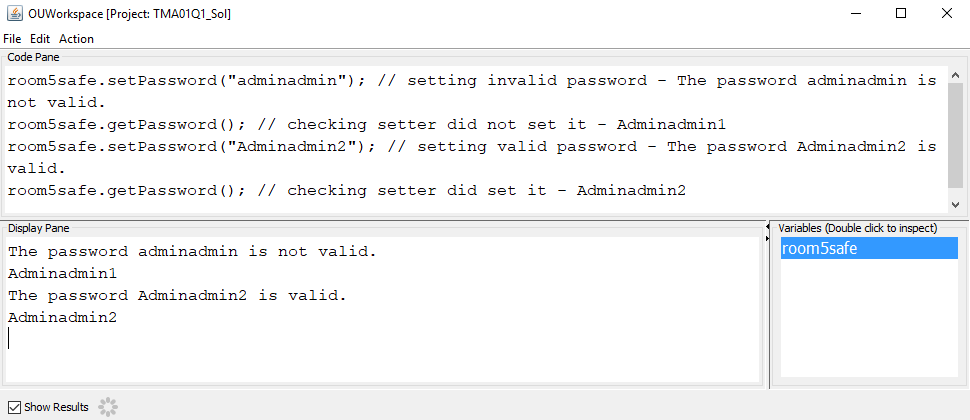
System.out.println("The password " + pw + " is not valid.");

}

}

1e) Test run





1f)

The getter method getPassword() could be made private. This would stop objects from other classes accessing what should be secure information. The hasChanged() method created in Question 1g is one way the getPassword() could be used to access the password without compromising the data. If access to the password is required, then a method that stores the user ID asking for the password would be advisable.

The methods isValidLength(), hasDigit(), hasUpperCase() and isValidPassword() could all be made private. The setPassword() method accesses these methods to determine whether the entered password is valid, therefore keeping them for internal use only.

1g) hasChanged() instance method

/\*\*

\* Return false if password is still the default, otherwise

\* return true as password has been changed.

\*/

public boolean hasChanged()

{

if(getPassword().equals("Adminadmin1"))

{

return false;

}

else

{

return true;

}

Test run

