Name of the Course : Complete Java SE8 Developer Bootcamp

Level : Easy

Tool Stack: Java 8 and Junit5

Problem Statement : [Constructors] A college management application like to initialise students based on personal details and grade .Provide appropriate constructors to initialise objects .The constructors should provide default values when not provided by user.

Description : Create class Student with following attributes and methods

| **Attributes** | **Data Type** | **default value** |
| --- | --- | --- |
| studentid | Integer |  |
| name | String |  |
| grade | String | “Z” |
|  |  |  |

| **Methods** | **Description** |
| --- | --- |
| Student() | default constructor |
| Student(int id ,String name) | parameterised constructor |
| Student (int id , String name,String grade) | parameterised constructor |
|  |  |

Learning outcome: Participant could able to learn how to overload constructors and use constructor chaining.

Name of the Course : Complete Java SE8 Developer Bootcamp

Level : Easy

Tool Stack : Java8 and Junit5

Problem Statement : [String API ] Contact Printer - A Contacts app take comma separated values for contact and prints information in a specific format .

Description : Create a simple class **Contact** with the following attributes .Provide appropriate constructors ,getters and setters.

| **Attributes** | **Data Type** | **default value** |
| --- | --- | --- |
| contactid | Integer |  |
| name | String |  |
| email | String |  |
| phone | integer |  |
| company | String |  |

Create MainClass with main method to accept contact as comma separated value and Print in the below format .

Sample Input :

1001, Abishek Sharma , [abhis@gmail.com](mailto:abhis@gmail.com),9876568456 , CTS ,Delhi -112690

Sample Output:

Contact Id : 1001

Name : Abishek Sharma

Email : [abhi@gmail.com](mailto:abhi@gmail.com)

company : CTS

City : Delhi

Pin code : 112690

Name of the Course : Complete Java SE8 Developer Bootcamp

Level : Easy

Tool Stack : Java8 and Junit5

Problem Statement : Provide a code solution to check the user credentials using Java 8 predicate.

Description : Create two classes one User class with username and password fields and with a parameterized constructor and another MainClass with two static methods 1. Public static Boolean checkUser(User user), which accepts User object and returns true or false

2. pubic static void main method, for reading the username and password from input devices and call the checkUser method to test it.

Code:

**public** **class** User {

**private** String username;

**private** String password;

**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;

}

**public** User(String username, String password) {

**super**();

**this**.username = username;

**this**.password = password;

}

}

**import** java.util.\*;

**import** java.util.function.Predicate;

**public** **class** MainClass {

**public** **static** **boolean** checkUser(User user) {

Predicate<User> predicate = usr-> usr.getUsername().equals("admin") && usr.getPassword().equals("1234");

**return** predicate.test(user);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("Enter your login");

String uname = scanner.nextLine();

System.***out***.println("Enter password");

String pwd = scanner.nextLine();

User user = **new** User(uname,pwd);

**if**(*checkUser*(user))

System.***out***.println("Valid User");

**else**

System.***out***.println("Invalid User");

}

}

Junit Testing

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** org.junit.jupiter.api.Assertions;

**import** org.junit.jupiter.api.Test;

**class** MainClassTest {

@Test

**void** testCheckUserLogin() {

//Test will pass

User user1 = **new** User("admin","1234");

User user2 = **new** User("admin","admin");

Assertions.*assertEquals*(**true**, MainClass.*checkUser*(user1));

Assertions.*assertEquals*(**false**, MainClass.*checkUser*(user2));

}

}

Test Data1

Enter your login

admin

Enter password

1234

Valid User

Test Data2

Enter your login

admim

Enter password

test

Invalid User

Learning outcome: Participant could able to learn how to use the predicate functional interface.