```java
package User;
import java.io.*;
/**
 * Przykład działającego Singletona max statycznego.
 *
 * Przykład Singletona z metodami statycznymi odnoszącymi się do pol
 * zadeklarowanego jedynego singletona metodami statycznymi - tzn. mamy
 * statyczne settery i gettery.
 *
 * Metody loadUser nie dało się zrobić statycznej pomimo że inteliJ to przyjmuje
 * potem uruchomienie wyrzuca błędy.
 *
 * */
public class User {

    private String login;
    private String password;
    private String color;
    private boolean userLoginStatus=false;
    private static User oneUser=new User();

    private User(){
        System.out.println("tu konstruktor oneUsera");
        loadUserData();
        printUser();
    }

    public static User getOneUser(){ return oneUser; }

    public void loadUserData(){
        String nazwaPliku ="src/Resources/iniUserData.txt";
        //String nazwaPliku="D:\\PROGRAMMING\\JAVA\\Warehouse1\\src\\Resources\\
iniUserData.txt"
        File plik = new File(nazwaPliku);
        FileReader fr;
        BufferedReader br;

        try {
            fr=new FileReader(plik);
            br = new BufferedReader(fr);
            login=br.readLine();
            password=br.readLine();
            color=br.readLine();
            userLoginStatus=Boolean.parseBoolean(br.readLine());
                //br.close();
        }
        //catch (FileNotFoundException e) {e.printStackTrace(); }
        catch (IOException e) {e.printStackTrace(); }
    }

    public static void saveUserData(){
        String nazwaPliku ="src/Resources/iniUserData.txt";
        File plik = new File(nazwaPliku);

        try {
```

```
55              BufferedWriter bw = new BufferedWriter(new FileWriter(plik));
56              bw.write(oneUser.login+"\n");
57              bw.write(oneUser.password+"\n");
58              bw.write(oneUser.color+"es\n");
59              bw.write("false2");
60              bw.close();
61          }
62          //catch (FileNotFoundException e) {e.printStackTrace(); }
63          catch (IOException e) {e.printStackTrace(); }
64      }
65
66      public static void printUser(){
67          System.out.println("teras static z USER");
68      }
69
70      public static String getLogin() {
71          return oneUser.login; }
72
73      public static String getPassword() {
74          return oneUser.password; }
75
76      public static String getColor() {
77          return oneUser.color; }
78
79      public static boolean isUserLoginStatus() {
80          return oneUser.userLoginStatus; }
81
82      public static  void setLogin(String login) {
83          oneUser.login = login; }
84
85      public static void setPassword(String password) {
86          oneUser.password = password; }
87
88      public static void setColor(String color) {
89          oneUser.color = color; }
90
91      public static void setUserLoginStatus(boolean userLoginStatus) {
92          oneUser.userLoginStatus = userLoginStatus; }
93
94      @Override
95      public String toString() {
96          return "User{" +
97                  "login='" + oneUser.login + '\'' +
98                  ", password='" + oneUser.password + '\'' +
99                  ", color='" + oneUser.color + '\'' +
100                 ", userLoginStatus=" + oneUser.userLoginStatus +
101                 '}';
102     }
103 }
104
```