# משימה מספר 3-1 PasswordManager

## מטרה:

בניית מנהל סיסמאות תוך שימוש במבנה הנתונים Map

## הוראות:

כתבו מנהל סיסמאות. מנהל הסיסמאות הוא אובייקט אשר מקבל שמות משתמשים וסיסמאות ואחר כך יודע לענות אם שם משתמש וסיסמא הוכנסו למאגר.

על מנהל הסיסמאות שלכם לקיים את הדרישות הבאות:

* בבנייתו מאגר הסיסמאות ריק.
* מכיל את הפונקציות הבאות:
* public int readFile(String filename)

קורא אוסף שמות וסיסמאות מהקובץ. מבנה הקובץ מתואר להלן. המתודה מחזירה את מספר הערכים שעודכנו במבנה הנתונים הפנימי.

* public void writeFile(String filename)

כותב את מאגר השמות והסיסמאות מהזיכרון לקובץ. מבנה הקובץ מתואר להלן.

* public void add(String name, String password)

מוסיף למאגר את השם והסיסמא. זורק חריגה מתאימה (שתכתבו) ולא עושה כלום אם השם כבר קיים.

* boolean checkPassword(String name, String password)

בודק אם השם והסיסמא במאגר. מחזיר true אם"ם השם נמצא במאגר עם הסיסמא הנתונה.

מבנה קובץ הסיסמאות: בכל שורה בקובץ שם משתמש ואחריו ערבול של סיסמא (ראו הסבר להלן), מופרדים ברווחים.

התו # מסמן תחילת הערה, כלומר ממנו ואילך יש להתעלם. בקריאת הקובץ יש להתעלם משורות לא חוקיות ומשמות משתמשים שכבר קיימים במאגר.

ערבול של סיסמא מושג על ידי שימוש באלגוריתם דוגמת SHA-1. שימו לב שהאלגוריתם (לפחות לענייננו) בלתי הפיך, לכן כדי להשוות סיסמא גולמית לסיסמא מעורבלת יש לערבל את הסיסמא המקורית ולהשוות את התוצאה. תוכלו להיעזר במתודה הבאה לשם קבלת הערבול של מחרוזת. עליכם להוסיף הערה לכל שורה המסבירה את תפקידה בדיוק ובקיצור.

היעזרו במבנה נתונים מסוג Map.

**public** **static** String calcSHA1(String str) {

MessageDigest md;

String ret = "";

**try** {

md = MessageDigest.*getInstance*("SHA1");

md.update( str.getBytes() );

**byte**[] digest = md.digest();

**for** (**byte** b : digest) {

String hNum = Integer.*toHexString*(b & 0xFF );

**if** (hNum.length() == 1)

hNum = "0" + hNum;

ret = ret + hNum;

}

} **catch** (NoSuchAlgorithmException e) {

**return** **null**;

}

**return** ret;

}

## דוגמא להמחשה:

בדקו את התוכנית באמצעות הקוד הבא:

PasswordManager pm1 = **new** PasswordManager();

PasswordManager pm2 = **new** PasswordManager();

**try** {

pm1.add("user1", "secret1");

pm1.add("user2", "secret2");

pm1.add("user3", "secret3");

pm1.add("user4", "secret4");

System.*out*.println("Test1 - directly added:");

System.*out*.println("Should be " + **true** + ": " +

pm1.checkPassword("user1", "secret1"));

System.*out*.println("Should be " + **true** + ": " +

pm1.checkPassword("user3", "secret3"));

System.*out*.println("Should be " + **false** + ": " +

pm1.checkPassword("user2", "secet2"));

System.*out*.println("Should be " + **false** + ": " +

pm1.checkPassword("uer1", "secret1"));

pm1.writeFile("passwords.txt");

} **catch** (UserExistsException e) {

System.*out*.println("Error: a user already exists. ");

} **catch** (IOException e) {

System.*out*.println(e);

**return**;

}

**try** {

pm2.readFile("passwords.txt");

System.*out*.println("Test2 - loaded:");

System.*out*.println("Should be " + **true** + ": " +

pm2.checkPassword("user1", "secret1"));

System.*out*.println("Should be " + **true** + ": " +

pm2.checkPassword("user3", "secret3"));

System.*out*.println("Should be " + **false** + ": " +

pm2.checkPassword("user2", "secet2"));

System.*out*.println("Should be " + **false** + ": " +

pm2.checkPassword("uer1", "secret1"));

} **catch** (IOException e) {

System.*out*.println(e);

**return**;

}

}