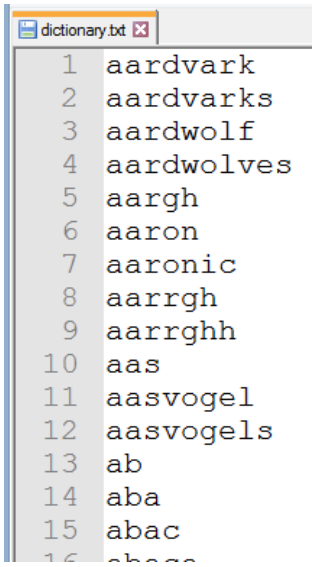
A database called “ISD3Exam” (which is available for download from Moodle) contains a single table called users. The structure of the users table is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	**Username**	varchar(25)	latin1_swedish_ci		No	*None*		Change Drop Primary Unique Index
2	**Password**	varchar(60)	latin1_swedish_ci		No	*None*		Change Drop Primary Unique Index

The table stores the usernames and passwords of 17 members of the IT Department. The usernames are stored in plaintext but the passwords are encrypted with MD5. The contents of this table are as follows:

Username	Password
alan.ryan	c395246f710b0e2c86b7ed82f7f56ce3
brendan.watson	f3639baeb4530db03ef930eb16073f61
carol.rainsford	599dcce2998a6b40b1e38e8c6006cb0a
des.ocarroll	9b7d722b58370498cd39104b2d971978
elizabeth.bourke	c6234e283c3ccfc2dad5469d5d98c739
gerry.guinane	030bb043d582f8f5569a8be85ea51911
ita.kavanagh	7c8c54bbcb0c6a411c5a87b50c3ac8b4
john.jennings	3518e159542afc83642c21dd643112a4
ken.oakley	09ef45a3371030c09e8497a2b562d5e0
lindy.farmer	0897747926d64dd6b27d892887021d71
lorraine.callanan	feb789b85f47c1182e6c547af98749a7
mike.oconnell	7c6a180b36896a0a8c02787eeafb0e4c
neil.higgins	d32787f6ce73d0611e541e2fedcad42d
oliver.hyde	87d08454821ecd4205704ae61aeb5c84
seamus.doyle	67057548df58d3cc208f89f6ff5e4e7e
suzanne.ogorman	98ded198ba3a0a30e7cc64e4a6a93f45
willie.ward	fd820a2b4461bddd116c1518bc4b0f77

A file called *dictionary.txt* (which is also available for download) contains (in plaintext) words from the dictionary along with commonly used passwords. There are *354926* lines in the file and each line contains a different word. The following screengrab shows the first 15 lines of this file:



Write a program that carries out the following task: “encrypt each word in the file with MD5 and then search the users table in the database for any passwords that match each encrypted word. You must then print the passwords of any users you find in plaintext”. This task must run as a separate thread in your program.

Your program should identify the following passwords for the following usernames.

```
password for alan.ryan is thursday
password for brendan.watson is cloudy
password for carol.rainsford is type
password for des.ocarroll is darkness
password for gerry.guinane is frequent
password for ita.kavanagh is appropriate
password for ken.oakley is limerick
password for lorraine.callanan is surplus
password for mike.oconnell is password1
password for neil.higgins is hardenability
password for suzanne.ogorman is planning
password for willie.ward is william
```

Your program won’t (shouldn’t) print the passwords for all the users. This is because some users have passwords which were not found in the file. For example, the password for *john.jennings* is *asq257GtrFase3D*. Remember, the file only contains words from the dictionary and commonly used passwords. The moral of the story is never use a password that is a word from the dictionary or easily guessed.

This program is carrying out what is called a dictionary attack to crack passwords. To mitigate against such an attack, developers should salt all passwords.

You should make use of the following code\method in your solution. This method will hash\encrypt a `String` using a specified algorithm. This method accepts two `String`s.

1. The first `String` represents the algorithm that will be used.
2. The second `String` represents the `String` (in plaintext) that will be encrypted.

```
private static String hashPassword(String algorithm, String password) {
    StringBuilder sb = new StringBuilder();
    try {
        MessageDigest messaged = MessageDigest.getInstance(algorithm);
        messaged.update(password.getBytes());
        byte[] mdArray = messaged.digest();
        sb = new StringBuilder(mdArray.length * 2);

        for (byte b : mdArray) {
            int v = b & 0xff;
            if (v < 16) {
                sb.append('0');
            }//end if
            sb.append(Integer.toHexString(v));
        }//end for

    }//end for
    catch (NoSuchAlgorithmException nsae) {
        System.out.println(nsae);
    }//end catch
    return sb.toString();
  }//end hashPassword
```

To use this method simply call it like this:

```
String encryptedText = hashPassword("MD5", "aString");
```

- This is where *MD5* is the encryption algorithm used and *aString* is the plaintext you wish to encrypt. *encryptedText* obviously contains "aString" hashed with MD5.