

Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 02 REPORT

MUSTAFA BİNGÜL 141044077

Course Assistant:

Nur Banu ALBAYRAK

Q1-/

DATA STRUCTURES
HWO2#

Q1

```
① for (int i=0; i<n-1; i++) {  
    for (int j=i+1; j<n; j++) {  
        3 simple statement  
    }  
}
```

3 simple statement'a göre değişir durum eğer 3 simple statement arasında return ifadesi var ise döngülerin ilk değerlerinde saptandığını düşünürsek Best case ve worst case durumları olarak ele alınız ve Best case ($T_B(n) = \Theta(1)$) olur ve worst case ($T_W(n) = \Theta(n^2)$) olur. Ama genel ortamda bakalım Bu algoritma $T(n) = \Theta(n^2)$ Sürede çalışmaktadır.

```
② public static int length(string str) {  
    if (str == null || str.equals("")) } → constant time (1)  
    return 0;  
    else  
    return 1 + length(str.substring(1)); }  
                                     T(n-1)
```

substring fonksiyonu verilen kumanda itibaren string döndürmektedir string uzunluğuna n derseniz;

$$T(n) = 1 + T(n-1), \quad T(1) = 1$$

$$= 2 + T(n-2)$$

$$= 3 + T(n-3)$$

$$= \dots$$

$$= k + T(n-k)$$

$$= n-1 + T(1)$$

$$\underline{T(n) = \Theta(n)}$$

$$(k=n-1)$$

Toplama işlemi içinde constant time derseniz (1)

Q2-/

Q2-

① Bu fonksiyon selection sort yapar.

② $T_B(n) = \Theta(n^2) \rightarrow \text{Best Case}$

$T_W(n) = \Theta(n^2) \rightarrow \text{Worst Case}$

Dizideki n elemanı için $n-1$ tane karşılaştırma olur,
Aynı şekilde diğer elemanlar içinde bu karşılaştırma
sayısı 1'er 1'er azalarak devam eder.

Yani listedeki tüm elemanların karşılaştırmaları

$$\frac{n \cdot (n-1)}{2} \text{ kere gerçekleştirilir.}$$

Her eleman 1 kere yer değiştirir kesin. Tüm diziyi
için n kere yer değiştirme yapılır.

$$n + \frac{n \cdot (n-1)}{2} =$$

Bu algoritma $\text{pör} (\text{selection sort})$ listedeki tüm elemanlar
aynı işlemlerden geçmektedir.
Bundan ötürü $\Theta(n^2)$ süresinde çalışmaktadır.

Q3-/

Q3

$f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ ve $f(n) = \Omega(g(n))$
 $g(n) = 2^n \rightarrow$ fibonacci serisi örnek olabilir

$$\underbrace{c_2 f(n)}_{\Omega \text{ notasyonu}} \leq T(n) \leq \underbrace{c_1 f(n)}_{O \text{ notasyonu}}$$

Θ notasyonu

$$\boxed{c_2 \cdot 2^n \leq 2^n \leq c_1 \cdot 2^n} \rightarrow \text{Bu durumu sağlayacak } c_1 \text{ ve } c_2 \text{ değerleri vardır}$$

$c_2 = 1, n = 5, c_1 = 2$
için seçilir.

Q4-/

Q4-

①

```
public int[] recursiveSort(int[] arr, int n) {
    int i;
    if (n > 1)
        recursiveSort(arr, n-1);
    else {
        int j = arr[n];
        i = n-1;
        while (i >= 0 && arr[i] > j) {
            arr[i+1] = arr[i];
            i = i-1;
        }
        arr[i+1] = j;
    }
    return arr;
}
```

②

$T(n) = O(1)$, $n=1$ (Best case olur zaten sıralıdır)
 $T(n) = T(n-1) + O(n)$, $n > 1$ (Worst case sıralı olmadığı
durumda bahsedebiliriz)

③

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= n + (n-1) + T(n-2) \\ &= n + (n-1) + (n-2) + T(n-3) \\ &= \dots \\ &= \frac{n(n-1)}{2} + O(1) \end{aligned}$$

$$\boxed{T(n) = O(n^2)}$$

Arrayın sırası'ını bir
özellik tekrar göndeririz.

Q5-/

Q5

① $f(n) = n^{0.1}$, $g(n) = (\log n)^{10}$

$f(n) = O(g(n)) \longrightarrow \underline{n \geq n_0}$ ise:

$n^{0.1} \leq c \cdot (\log n)^{10}$

$\log n$ (fark) çok çok yavaş büyür $n^{0.1}$ 'e göre
Bundan ötürü $n^{0.1} \neq O((\log n)^{10})$ $f(n) = O(g(n))$ yaşanmaz.

$f(n) = \Omega(g(n))$

$c(\log n)^{10} \leq n^{0.1} \longrightarrow n \geq n_0$

Bu durumu sağlayacak n değerleri vardır. Zaten $\log n$ fonksiyonu yukarıda belirttiğim gibi çok yavaş büyür.

sonuç olarak;

$f(n) = \Omega(g(n))$
 $n^{0.1} = \Omega((\log n)^{10})$ ✓

② $f(n) = n!$, $g(n) = 2^n$

$f(n) = O(g(n)) \longrightarrow n \geq n_0$;

$n! \leq c \cdot (2^n)$

Bu iki fonksiyonda çok hızlı büyürler. Bu sebeple çok yavaş fonksiyondur. Yaptığım araştırmalar göre girdi sayılarının eşit olduğu durumlarda $(n!)$ daha hızlı büyümektedir. ve daha yavaştır. Bundan dolayı $f(n) \neq O(g(n))$ deriz. ve

$f(n) = \Omega(g(n))$

$c(2^n) \leq n!$ ✓

$\boxed{f(n) = \Omega(g(n))}$
 $n! = \Omega(2^n)$

Q.5

③

$$f(n) = (\log n)^{\log n} \quad g(n) = 2^{(\log_2 n)^2}$$

$$f(n) = O(g(n))$$

$$f(n) = n^{\log \log n}$$

$$g(n) = (2^{\log_2 n})^{\log_2 n} = n^{\log_2 n}$$

$$\frac{n^{\log \log n}}{n^{\log_2 n}} \leq c \cdot (n^{\log_2 n}) \quad \checkmark \quad \left(\begin{array}{l} n \geq n_0, \\ c, \text{ positif} \end{array} \right)$$

$$c=1, n=10$$

defai için saptarmaktadırlar zaten.

$$\underline{f(n) = O(g(n))} \Rightarrow (\log n)^{\log n} = O(2^{(\log_2 n)^2}) \quad \checkmark$$

Q6-/

Performance Analysis

Performans analizini yaparken *System.currentTimeMillis()*; methodunu kullandım. Öncelikle (book.csv) dosyasından 100 adet girdi ile kitap okuma işlemini gerçekleştirdim ve veri yapıma veri ekleme analizini gerçekleştirdim; Array ve Array List yapılarında 2 milisaniye ile okuma işlemini tamamlarken 100 girdi ile Linked List yapısında 1 milisaniyede tamamladım. Aynı şekilde 100 kitap arasından bir kitap ı sildirdim ve sonuç olarak Array, Array Listte 1 milisaniye hesaplarken Linked Listte 0 milisaniye sonucunu aldım bunun sebebi az veri olduğundan olabilir veri sayısı çoğaldıkça aralarındaki fark daha da fazla olabilir. Tüm kitapları sergilettiğim fonksiyonda ise yani verilere ulaşım testinde 3 veri yapısı için de 2 milisaniye sonucunu elde ettim bu durum aralarındaki farkı görecektir kadar yeterli veri olmadığından olabilir.

Sonuç olarak elde ettiğim verilere ve araştırmalarıma dayanarak Linked List daha hızlı olduğunu söyleyebilirim.

1.Requirements

1.1-Overall Description

Part-1

Ödevde basit bir kütüphane otomasyonu istenmiştir.

Tek yönetici vardır. (User ID:0 Password:123)

Yönetici kitap veya kullanıcı eklediğinde id leri otomatik sistem tarafından atanmaktadır.

Kullanıcı veya kitap ekledikten sonra yönetici id leri görmek isterse menülere ulaşım görebilir.

Yönetici kullanıcı ekleyip çıkarabilir aynı şekilde kitapta ekleyip çıkarabilir.

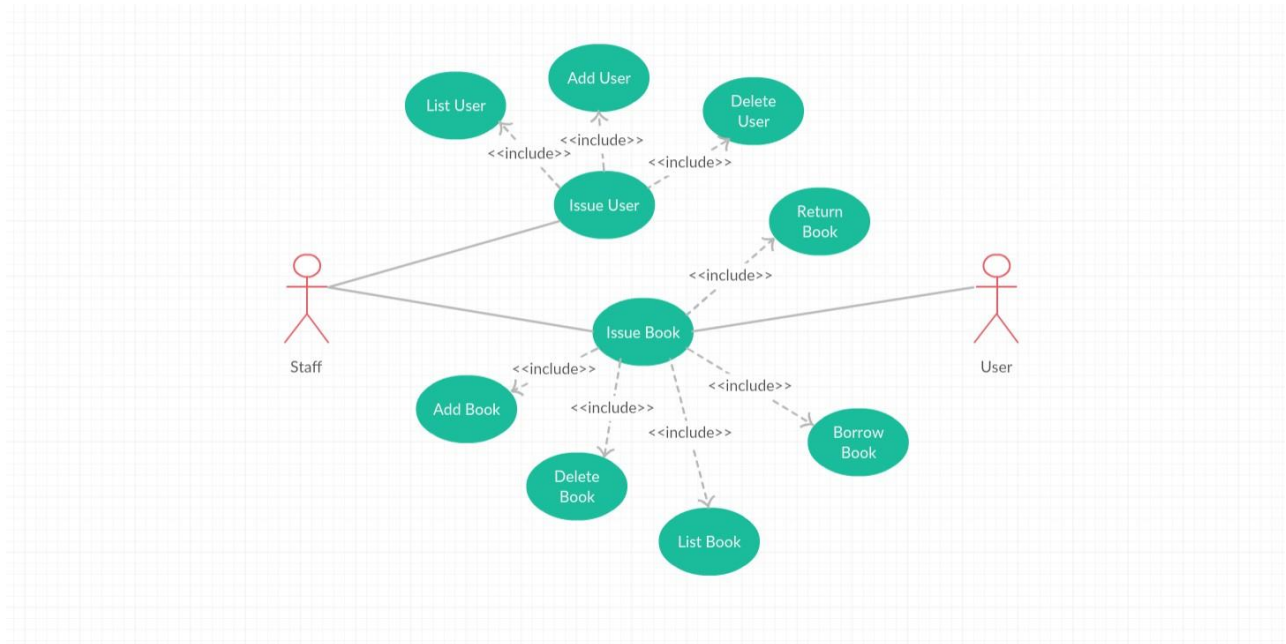
Birden fazla kullanıcı vardır ve istedikleri kitabı ödünç alabilir ve geri bırakabilirler. Son olarak kitap ve kullanıcı verileri (.csv) uzantılı dosyada tutulmaktadır.

Ödevde istenilen Array, ArrayList ve LinkedList yapılarına ayrı ayrı implementi yapılmıştır.

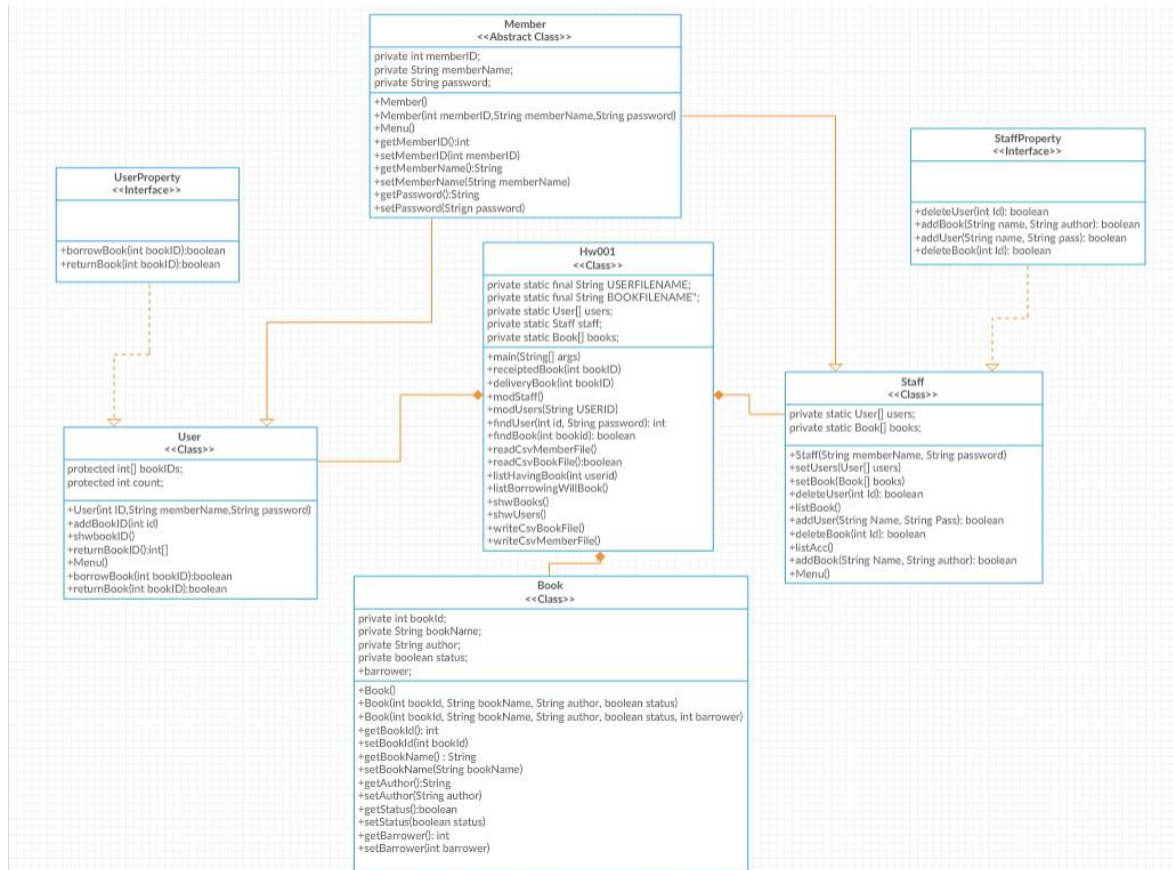
1.2-System Requirements

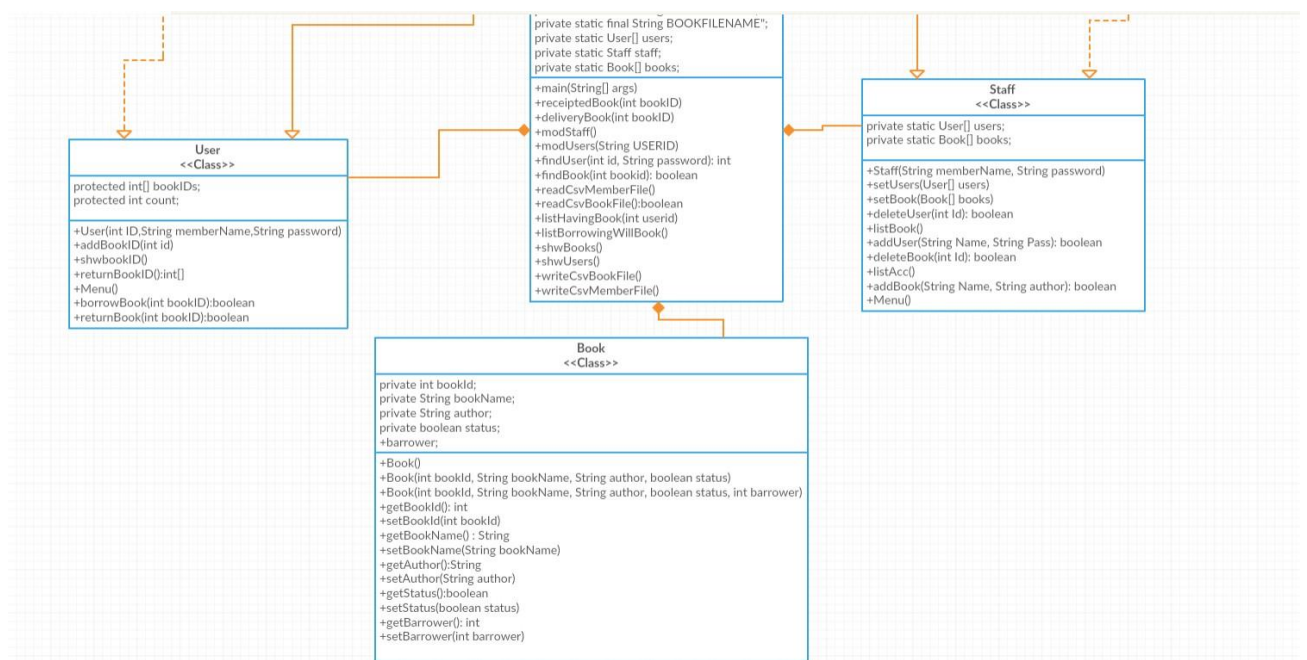
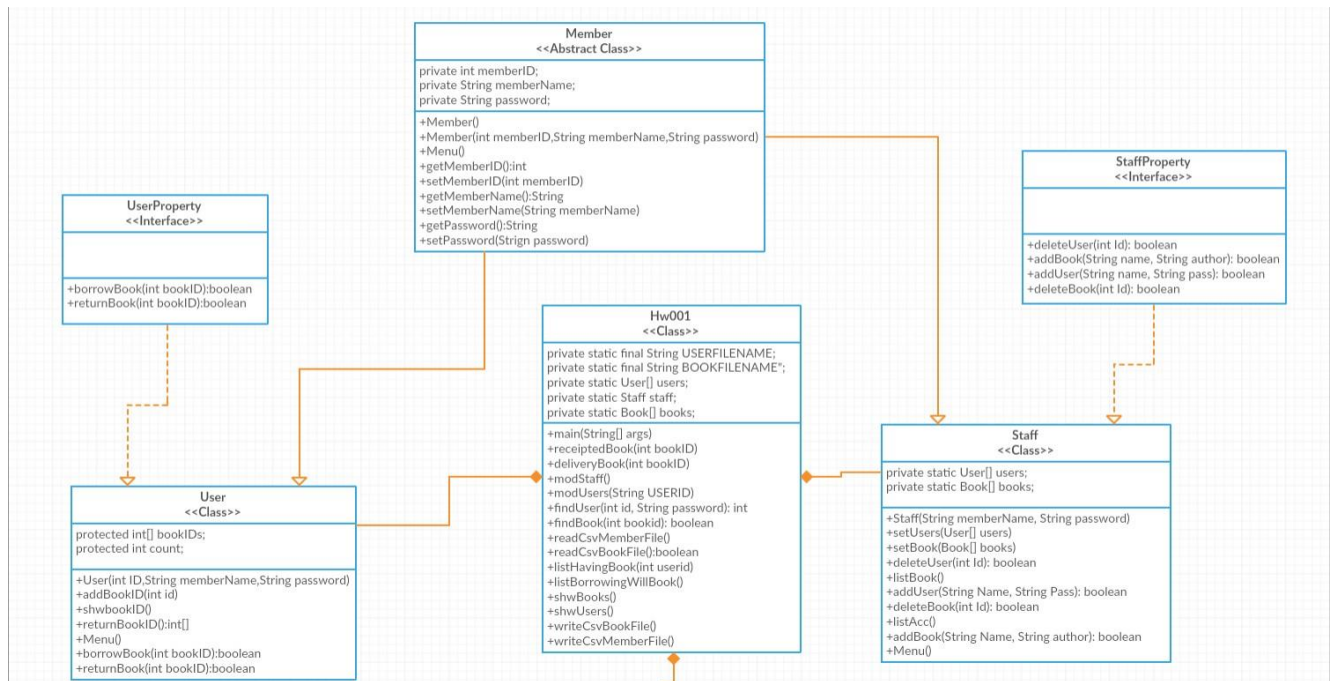
1.2.1 book.csv ve mem.csv dosyaları.

2. Use Case Diagrams



3. Class Diagrams





5. Problem Solutions Approach

Basit bir kütüphane otomasyonu yapmamız istendi. Bu durumu düşündüm ve kütüphane yöneticisi, kütüphane kullanıcılarının ve kitap nesnelerinin olması gerektiğine karar verdim. Book,Staff,User,Member sınıflarımı ve StaffProperty ve UsersProperty interface lerini oluşturdum. Staff sınıfımı abstract olan Member sınıfımdan extend edip StaffProperty interface imden implement ettim. Aynı şekilde User sınıfımda Member sınıfımdan extend edip UserProperty interface inden implement ettim. Ve main imi yazdığım Hw001 sınıfımın içinde User ve Book arrayları oluşturup ve Staff objesi oluşturdum. Ve ödevde istenilen yönetici için ve kullanıcılar için gerekli methodları yazarak projeyi böyle gerçekleştirdim. Ve ödevi Array,ArrayList ve LinkedList yapılarına ayrı ayrı implement ettim.

6. Test Cases

```
~/Desktop/Hwi01 - [Hwi01] - ~/Desktop/Hwi01/src/Hw001.java -
File Edit View Navigate Code Analyze Refactor Build Run Io
Hwi01 > src > Hw001 >
Run Hw001

~ WELCOME TO LIBRARY AUTOMATION ~
User ID:
0
Password:
123
Admin:Mustafa

***** MENU *****
1- Add a new member
2- Delete a member
3- List all accounts
4- List all books
5- Add a Book
6- Delete a book
7- Exit
Enter choice:
3
User ID - User Name
1 - Ahmet
2 - Mehmet

Admin:Mustafa

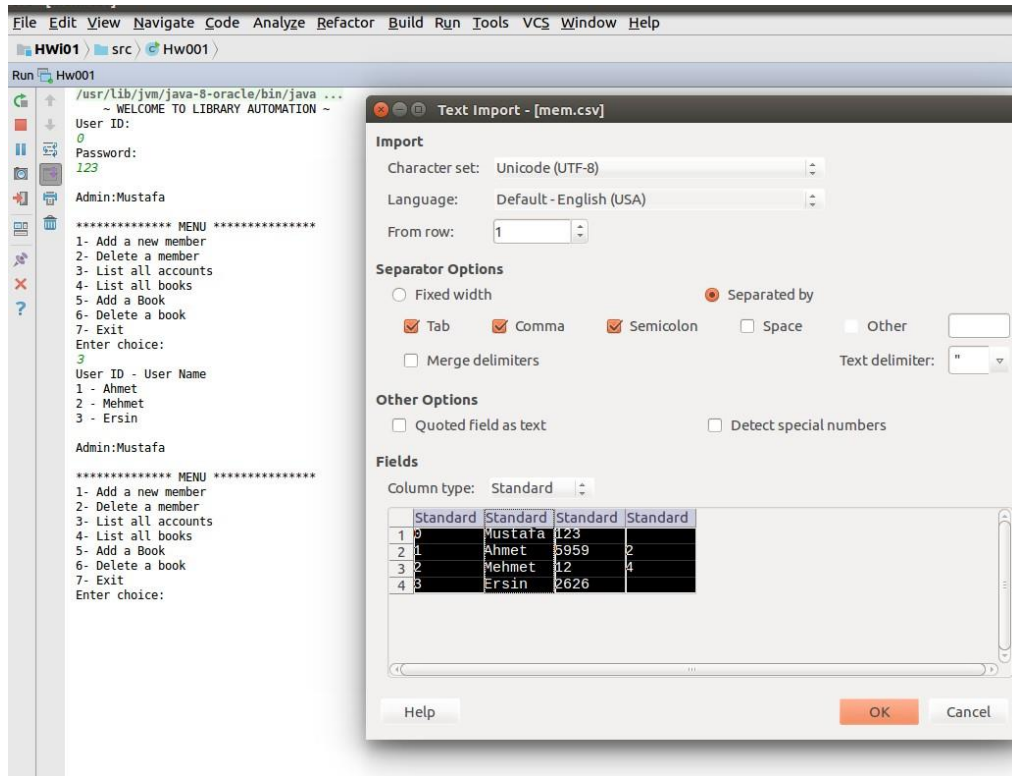
***** MENU *****
1- Add a new member
2- Delete a member
3- List all accounts
4- List all books
5- Add a Book
6- Delete a book
7- Exit
Enter choice:
1
Username - Password
Ersin
2626
Added.

Admin:Mustafa

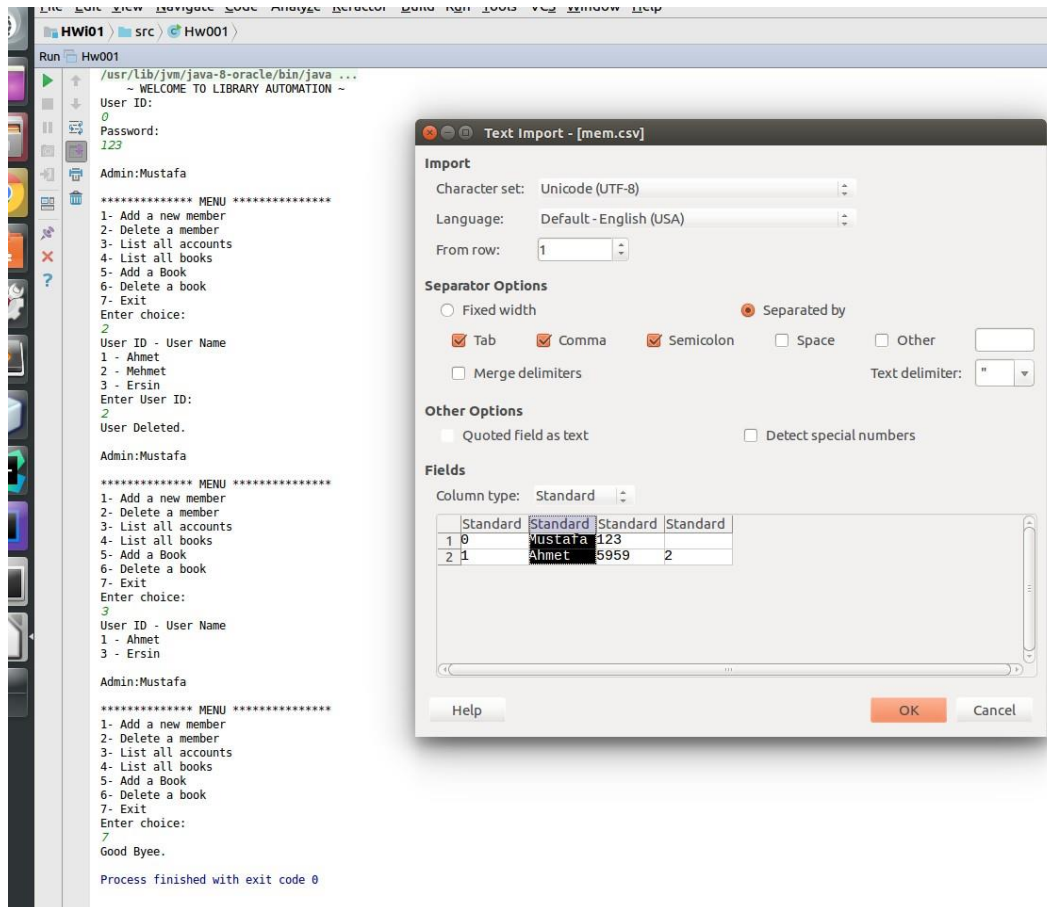
***** MENU *****
1- Add a new member
2- Delete a member
3- List all accounts
4- List all books
5- Add a Book
6- Delete a book
7- Exit
Enter choice:
3
User ID - User Name
1 - Ahmet
2 - Mehmet
3 - Ersin

Admin:Mustafa

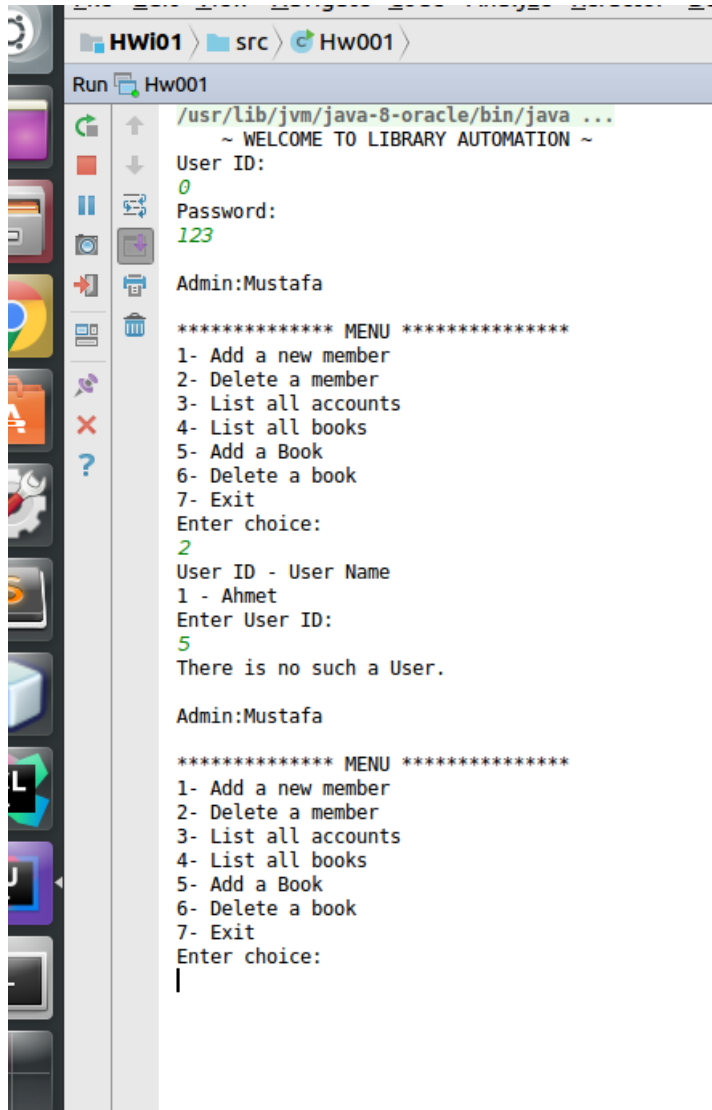
***** MENU *****
1- Add a new member
2- Delete a member
```



6.1-Kullanıcı ekleme başarılı.



6.2-Kullanıcı silme başarılı.



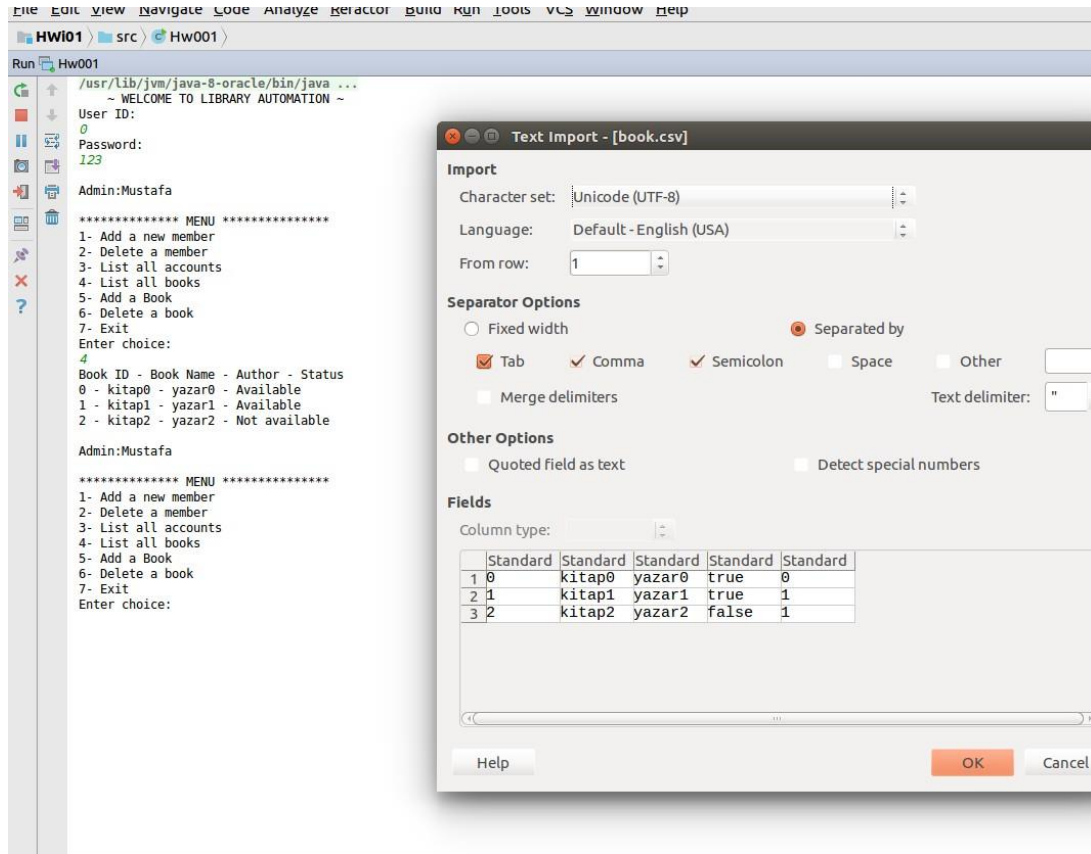
```
HWI01 > src > Hw001 >
Run Hw001
/usr/lib/jvm/java-8-oracle/bin/java ...
~ WELCOME TO LIBRARY AUTOMATION ~
User ID:
0
Password:
123
Admin:Mustafa

***** MENU *****
1- Add a new member
2- Delete a member
3- List all accounts
4- List all books
5- Add a Book
6- Delete a book
7- Exit
Enter choice:
2
User ID - User Name
1 - Ahmet
Enter User ID:
5
There is no such a User.

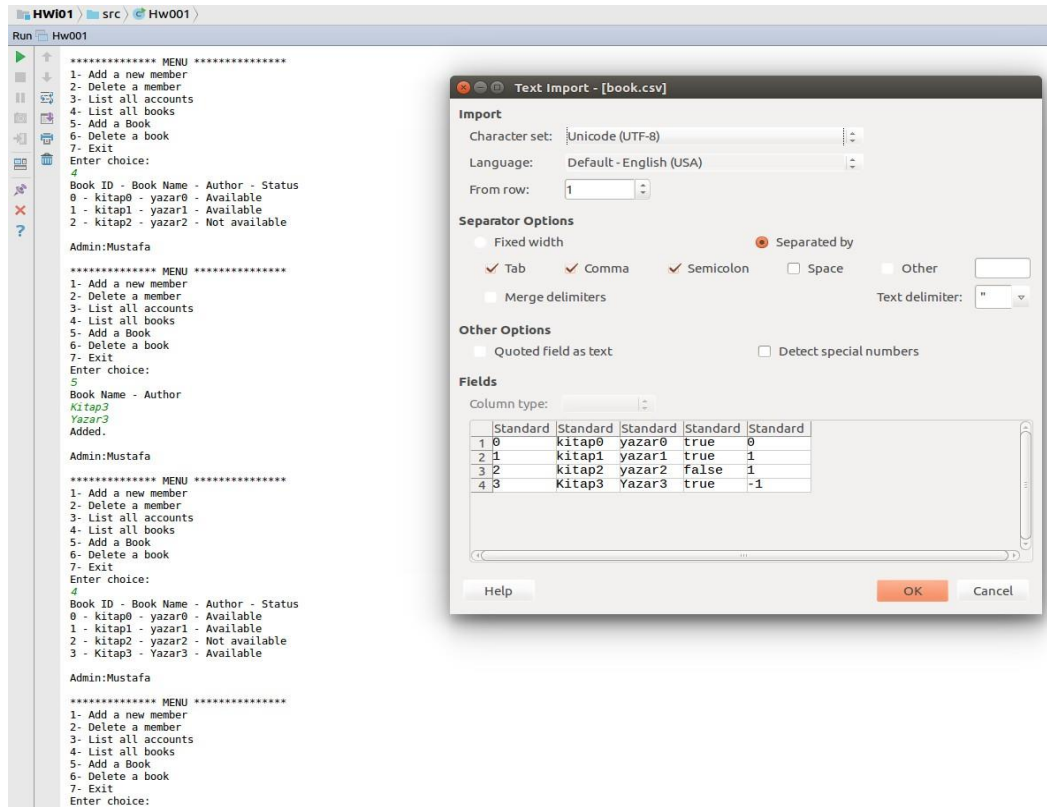
Admin:Mustafa

***** MENU *****
1- Add a new member
2- Delete a member
3- List all accounts
4- List all books
5- Add a Book
6- Delete a book
7- Exit
Enter choice:
|
```

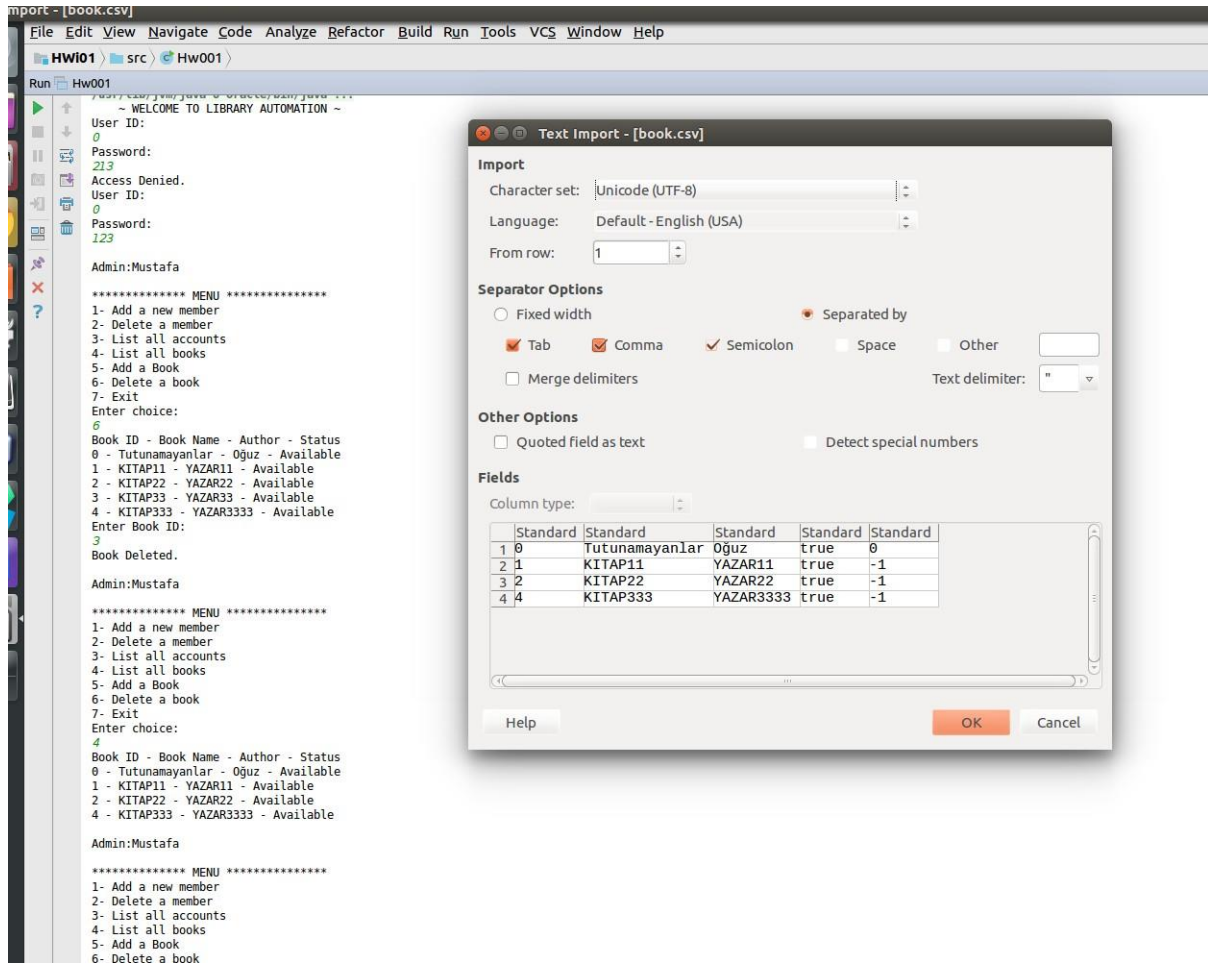
6.3-Olmayan kullanıcıyı silememe başarılı.



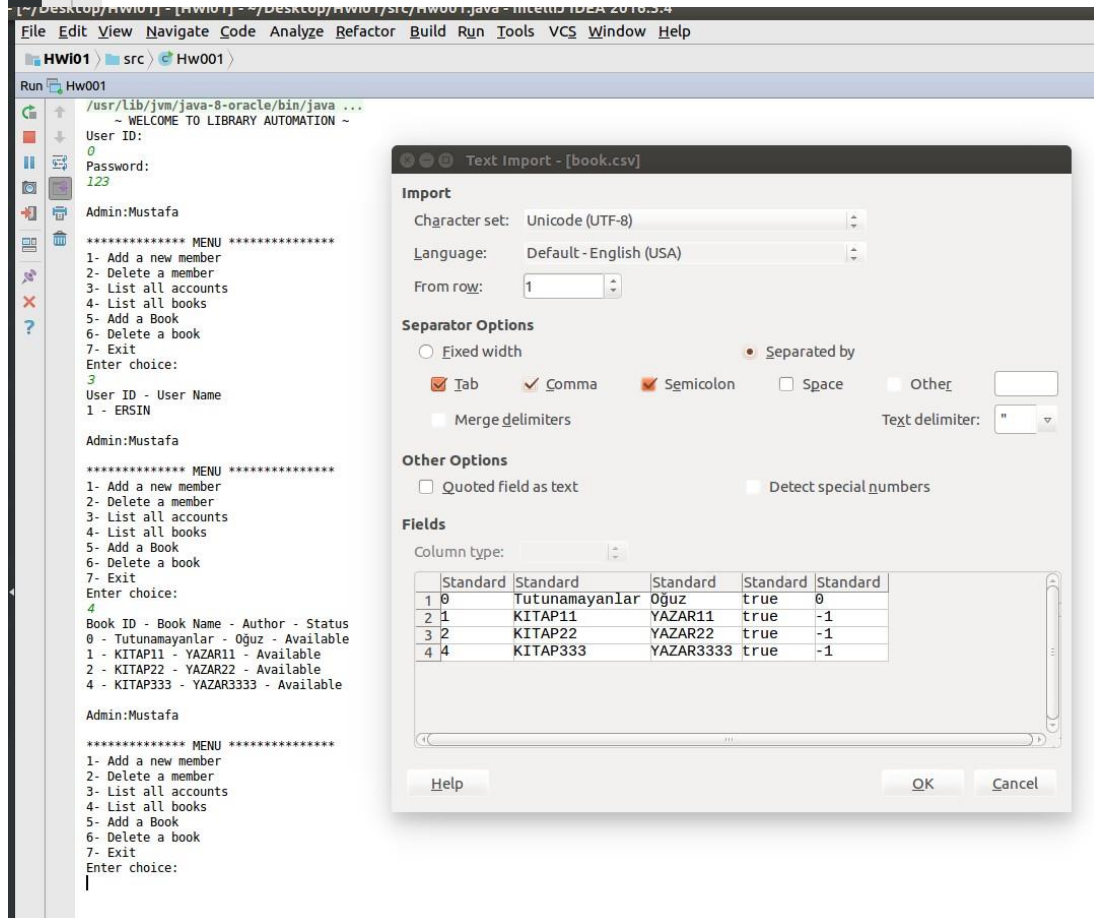
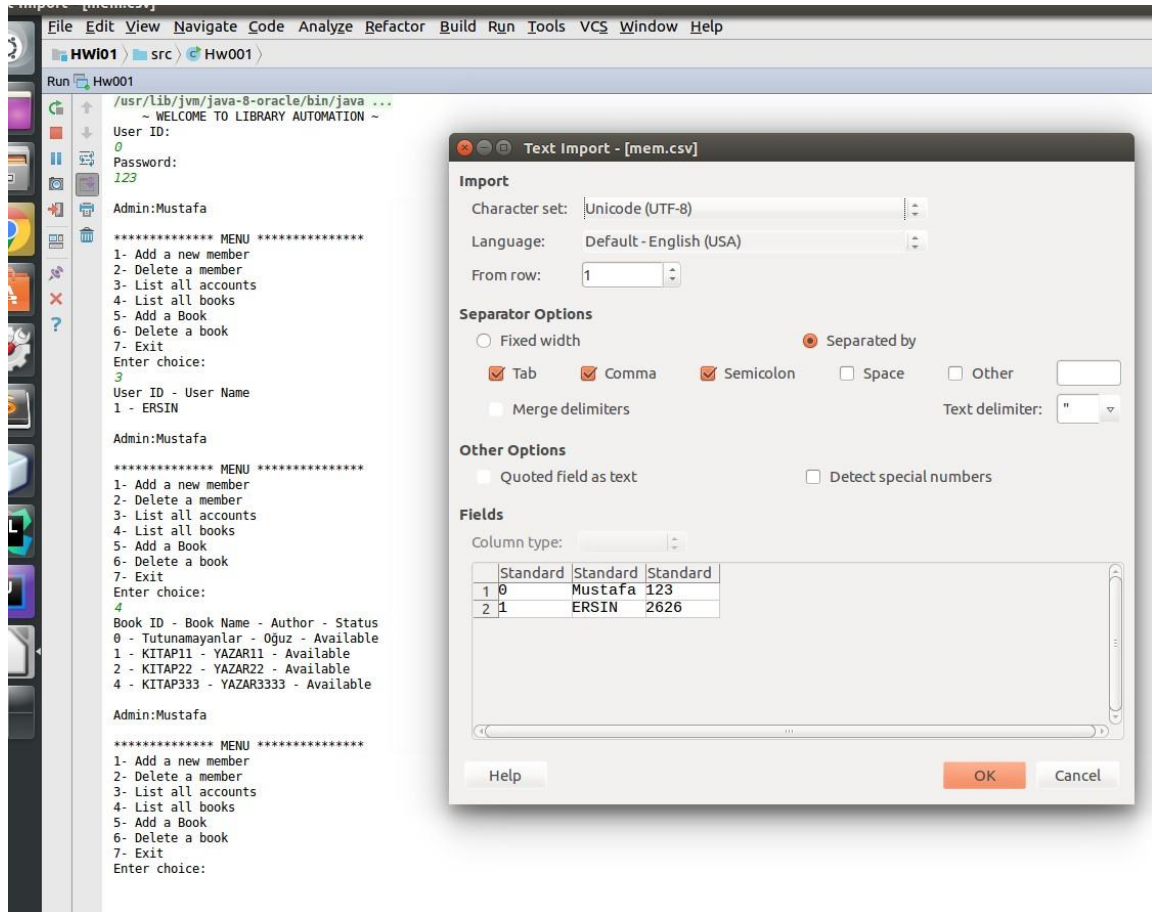
6.4-Sadece veritabanındaki kitaplar listelenmektedir.



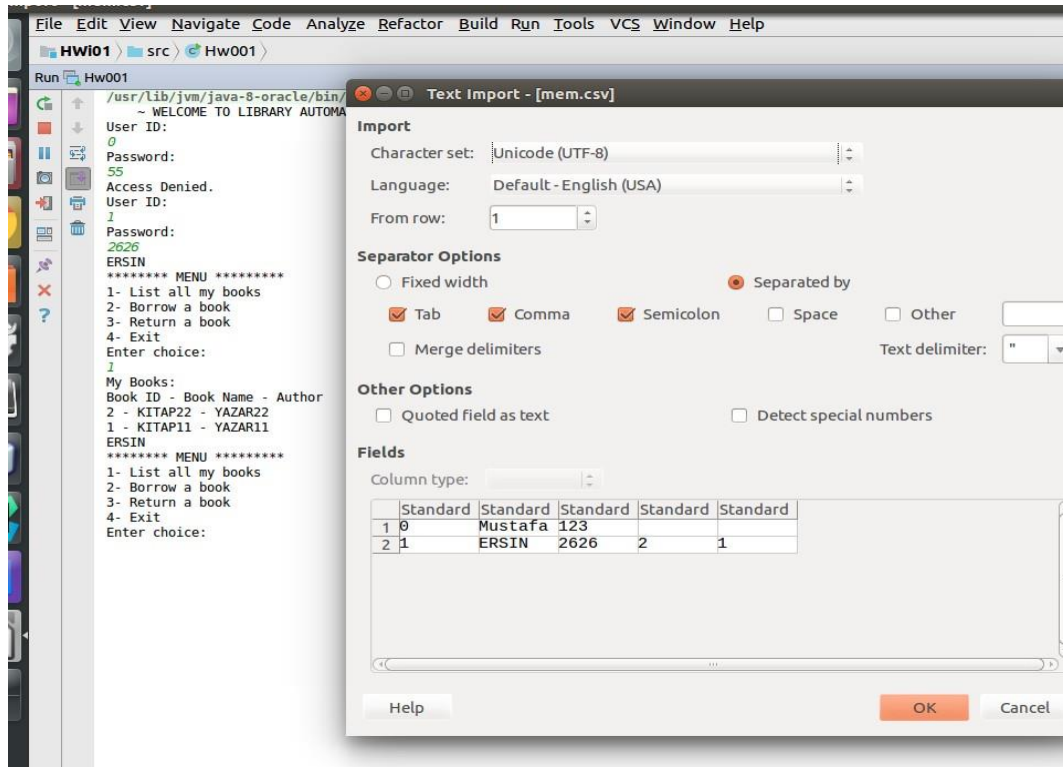
6.5-Kitap ekleme başarılı.



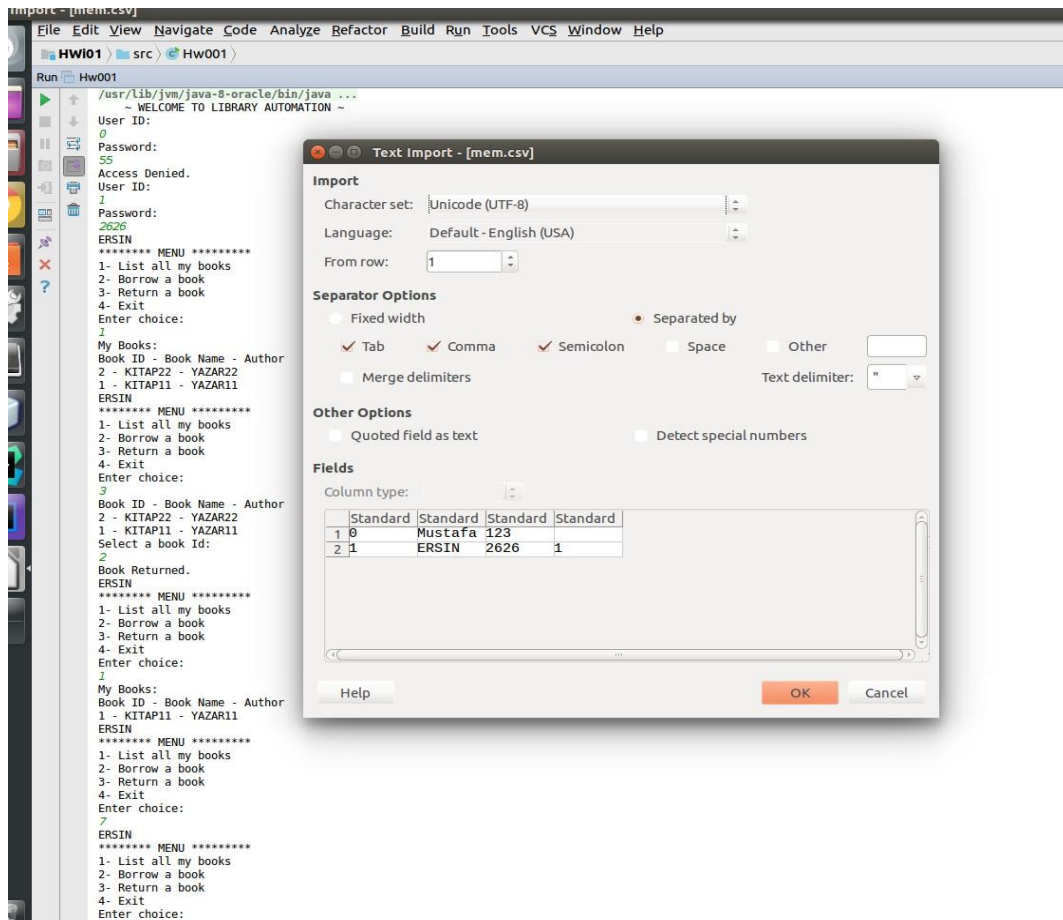
6.6-Kitap silme başarılı.



6.7-Tüm kitapları ve kullanıcıları listeleme başarılı.



6.8-Sadece kullanıcının sahip olduğu kitapları sergileme başarılı.



6.9-Ödünç alınan kitabı geri verme başarılı.

7. Running and Results

Bir yönetici vardır. (User ID:0 Password:123) Program çalıştırıldığından doğrulama gerçekleşir doğrulama ID ye göre gerçekleşir ve doğrulamadan sonra yönetici veya kullanıcı olarak devam eder. Ve program kullanıcılara id numaralarını otomatik atamaktadır. Eğer program yöneticiyi doğrularsa karşısına yönetici menüsü gelir. Ve yönetici; kitap ve kullanıcı ekleme, silme yetkilerine sahiptir aynı zamanda kütüphanedeki tüm kitapları ve tüm kullanıcıları görebilmektedir. Eğer program kullanıcıyı doğrularsa kullanıcı menüsü gelir. Ve kullanıcı; kitap ödünç alabilir ve geri verebilir aynı zamandan sadece ödünç aldığı tüm kitapları görebilir.

https://github.com/mstfbngl/HW02_141044077