

Assignment 11: Input Validation Lab Report

**Submitted by: Begum Fatima Zohra
UTA ID: 1001880881**

Contents

- Description of how code works
- Compilation/build instructions
- Installation, setup, and execution instructions
- Assumptions I have made
- Pros/Cons of my approach
- Bonus Attempted
- PHONE NUMBER REGEX USED
- SCREENSHOTS TO SHOW THE WORKING OF THE CODE
- SCREENSHOT AFTER MAKING SETUID
- SCREENSHOT OF MY FOLDER

Description of how code works

One Java class used - > **NamePhone.java**

We have used 6 methods and 1 main function.

4 methods for CRUD:

Add(), DeleteName(), DeletePhone(), ListAll()

2 helper methods for regex check:

num_validation(), name_validation()

main()-> Connection to the database is established in the main(). The Connection.java class variable is sent as an argument to Add(), DeleteName(), DeletePhone(), ListAll(). switch-case for interactive running of the program.

P.S: Optimization used: Only one connection was reused throughout the program. If we establish connection in each of the functions then the SQLite database will get blocked.

name_validation()-> used for regex matching of name taken as input from the user in the command line. Takes name in String format as an argument. returns 0 if successful and 1 if not.

num_validation()-> used for regex matching of telephone number taken as input from the user in the command line. Takes the phone number in String format as an argument. returns 0 if successful and 1 if not.

Add() -> used to take name and phone from the user in the command line, matches the regex in name_validation() and num_validation() then INSERT in the database if the data is correct.

ListAll()->used to see all the data present in the table using the SELECT query.

In order to display the data, we have provided the executed query to ResultSet class and read the derived result. Next, we have used the HashSet class to make

a new set of the derived result and put them in order. This helped in display of name and phone in the command line.

DeleteName()->used to take name from the user in the command line, matches the regex in name_validation() and then DELETE the record based on the name from the database if the data is correct.

DeletePhone()->used to take a phone number from the user in the command line, matches the regex in num_validation() and then DELETE the record based on the phone number from the database if the data is correct.

Compilation/build instructions

1. Open terminal in Ubuntu and go to the folder where project materials are downloaded.
2. Command for compilation

javac NamePhone.java

3. Command for running the program

java -classpath “.:sqlite-jdbc-3.34.0” NamePhone

4. TYPE A to insert, B to Delete by name, C to Delete by phone number and D to see the List

Installation, setup, and execution instructions

For this assignment, we have installed java, sqlite and sqlite browser.

For Java:

```
$ sudo apt-get update && apt-get upgrade
```

```
$ sudo apt-get install default-jdk
```

To access the java file:

```
$ vi NamePhone.java
```

For sqlite:

```
$ sudo apt-get update
```

```
$ sudo apt install sqlite3
```

```
$sqlite3 Validate.db
```

```
sqlite>CREATE TABLE Assign11(name TEXT NOT NULL, phone text NOT NULL);
```

For sqlite browser:

```
$ sudo apt-get update
```

```
$ sudo apt-get install sqlitebrowser
```

To access the DB manager GUI of sqlite, you can simply go to the apps section and double click it. Connect it with Validate.db database. You will now be able to see the Assign11 table and its tuples.

Assumptions I have made:

1. You have Ubuntu20.04(64 bit) installed in your computer.
2. You have downloaded sqlite jdbc jar file and kept in the same location as the java program.
3. You know at least American area codes for phone numbers.
4. You know how to use terminal (e.g. accessing a particular folder)

Pros/Cons of my approach

Pros

1. Menu driven switch-case is used to interact with the user easily.
2. Since our project is related to phone directory management, we have used regex to facilitate better processing of different patterns of inputs and reject the invalid ones.
3. For easy and hassle-free implementation of CRUD operations, we have used SQLite.

Cons

1. Regex is not easy to understand by everyone. It's difficult to implement all the scenarios. Some acceptable inputs provided in the instruction were rejected.

Bonus Attempted:

1. **Yes, a database was used** for storing the input data.
2. **Yes, prepared statements was used.**(API that supports parameterized queries)

PHONE NUMBER REGEX USED

String phonePattern =

```
"^(?:\\+?1[-.●]?)?\\((?([0-9]{3})\\|)?[-.●]?([0-9]{3})[-.●]?([0-9]{4})$"
+ "\\d{3}-\\d{4}$"
+ "\\^(\\+\\d{1,3}( )?)?(\\(\\d{3}\\|)\\d{3})[-. ]?\\d{3}[-. ]?\\d{4}$"
+ "\\^\\d{5}"
+ "\\^\\d{3}.\\d{3}.\\d{4}$"
+
"\\^(?:\\+?011[-.●]?)?\\((?([0-9]{3})\\|)?[-.●]?([0-9]{3})[-.●]?([0-9]{4})$"
+ "\\^(?:\\+?32[-.●]?)?\\((?([0-9]{2})\\|)?[-.●]?([0-9]{3})[-.●]?([0-9]{4})$"
+ "\\^(?:(:\\+|0{0,2})91(\\s*[\\-]\\s*)?|[0])?[789]\\d{9}$"
+ "\\^\\((?([0-9]{3})\\|)?[-.\\s]?([0-9]{3})[-.\\s]?([0-9]{4})$";
```

SCREENSHOTS TO SHOW THE WORKING OF THE CODE

1. INSERT into the Assign11 table

```
[05/08/21]seed@VM:~/.../Lab$ javac NamePhone.java
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite
-jdbc-3.34.0" NamePhone
Enter
  A to insert(name,phone)
  B to delete (name)
  C to delete (phone)
  D to see the list
A
Enter the name:
Tube
Enter the phone number:
111-9999
Valid data
Valid data
Insert successful
```

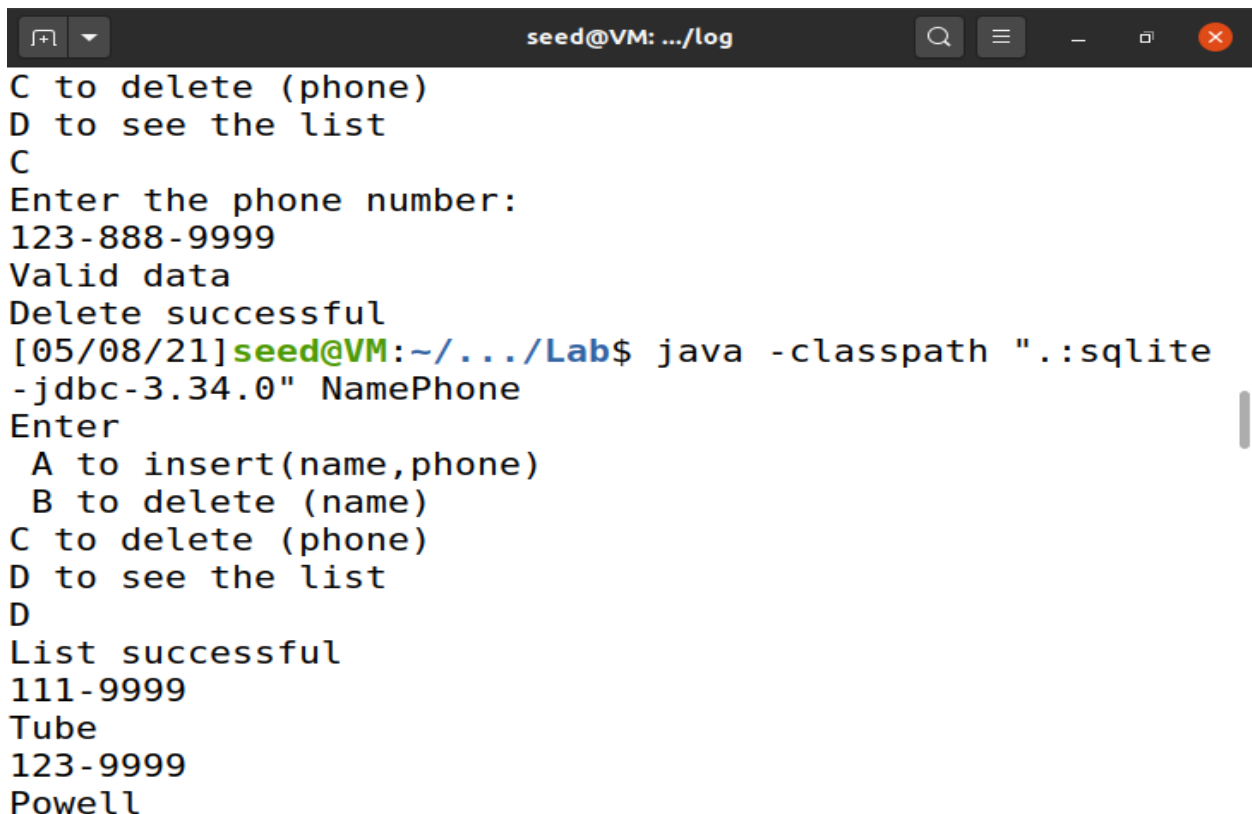
2. Check if the INSERT was successful or not(SELECT all the records)

```
seed@VM: .../log
Insert successful
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite
-jdbc-3.34.0" NamePhone
Enter
  A to insert(name,phone)
  B to delete (name)
  C to delete (phone)
  D to see the list
D
List successful
111-9999
Tube
Fatima
123-9999
123-888-9999
Powell
```

3. DELETE a record using phone as a parameter

```
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite
-jdbc-3.34.0" NamePhone
Enter
  A to insert(name,phone)
  B to delete (name)
  C to delete (phone)
  D to see the list
C
Enter the phone number:
123-888-9999
Valid data
Delete successful
```

4. Check if the DELETE was successful or not(SELECT all the records)

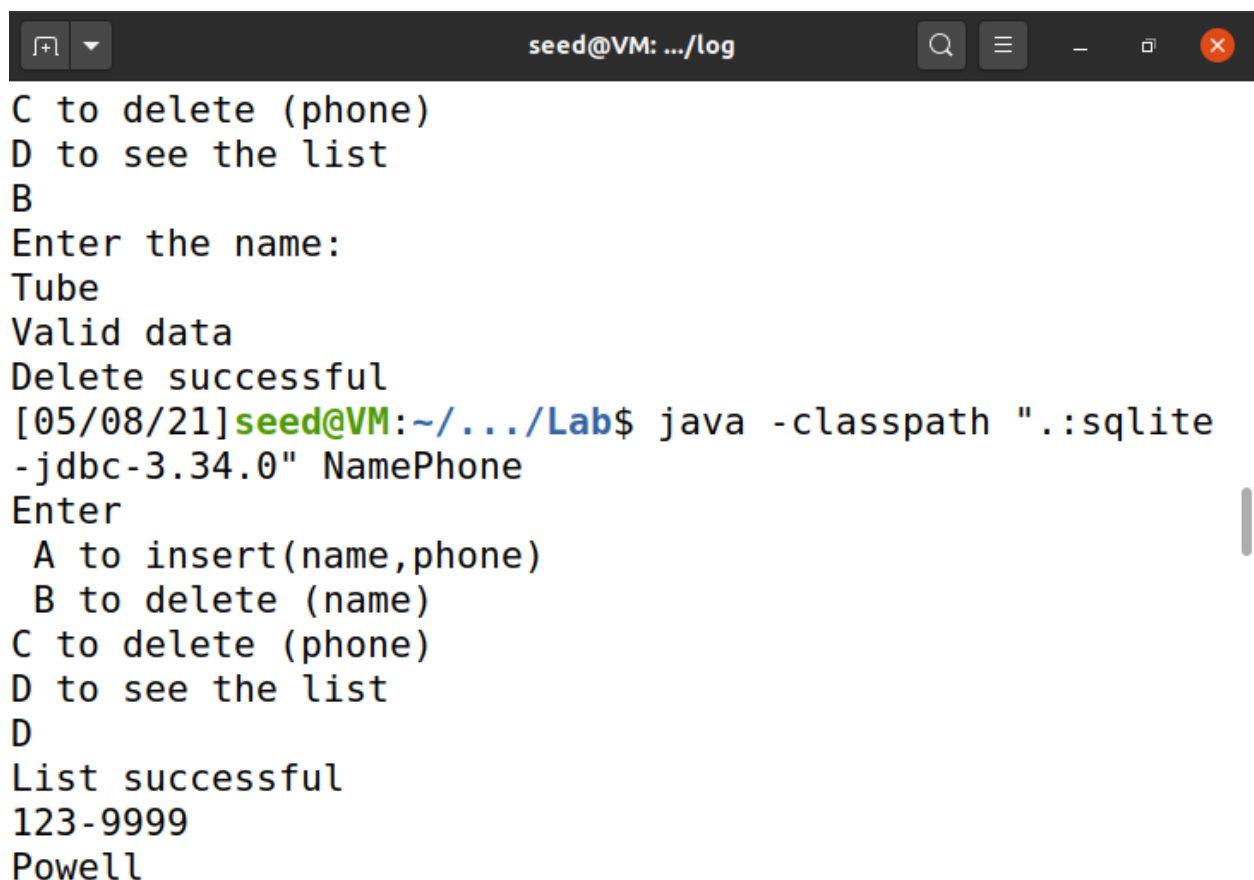


```
seed@VM: ~/log
C to delete (phone)
D to see the list
C
Enter the phone number:
123-888-9999
Valid data
Delete successful
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite
-jdbc-3.34.0" NamePhone
Enter
  A to insert(name,phone)
  B to delete (name)
  C to delete (phone)
  D to see the list
D
List successful
111-9999
Tube
123-9999
Powell
```

5. DELETE a record using name as a parameter

```
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite
-jdbc-3.34.0" NamePhone
Enter
A to insert(name,phone)
B to delete (name)
C to delete (phone)
D to see the list
B
Enter the name:
Tube
Valid data
Delete successful
```

6. Check if the DELETE was successful or not(SELECT all the records)

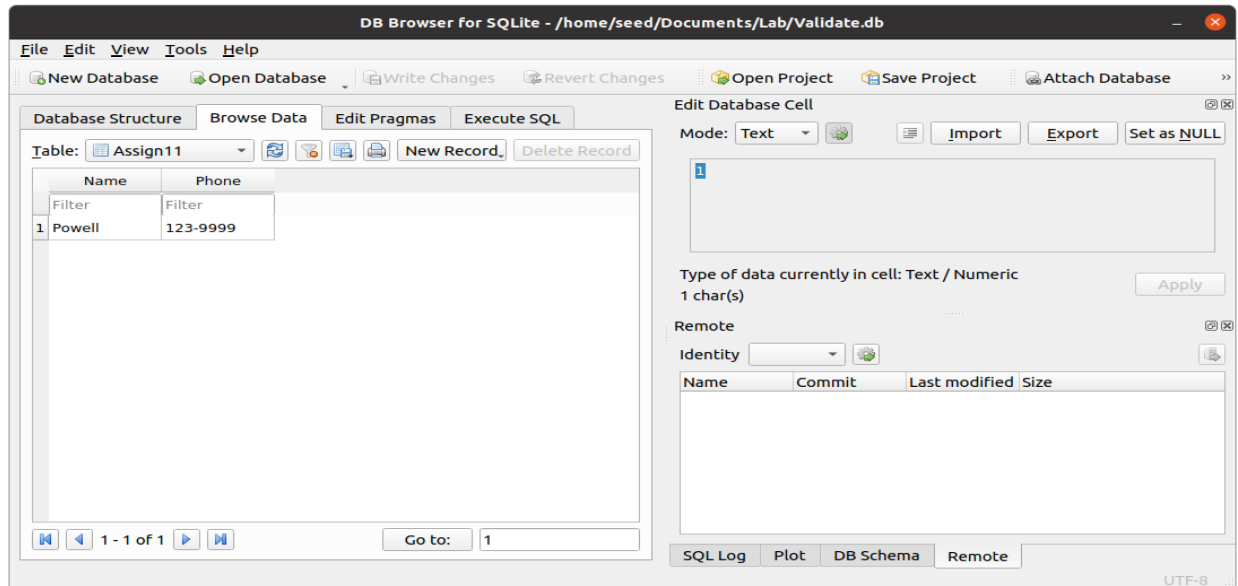


```
seed@VM: .../log
C to delete (phone)
D to see the list
B
Enter the name:
Tube
Valid data
Delete successful
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite
-jdbc-3.34.0" NamePhone
Enter
A to insert(name,phone)
B to delete (name)
C to delete (phone)
D to see the list
D
List successful
123-9999
Powell
```


7. Check if sqlite is giving the same result or not

```
sqlite> select * from Assign11;  
Powell|123-9999  
sqlite> █
```

8. Check if sqlite browser is giving the same result or not



SCREENSHOT AFTER MAKING SETUID

```
seed@VM: ~/.../Lab  
[05/08/21]seed@VM:~/.../Lab$ sudo chown root NamePhone  
.java  
[05/08/21]seed@VM:~/.../Lab$  
[05/08/21]seed@VM:~/.../Lab$ sudo chmod 4755 NamePhone  
.java  
[05/08/21]seed@VM:~/.../Lab$ java -classpath ".:sqlite  
-jdbc-3.34.0" NamePhone  
Enter  
A to insert(name,phone)  
B to delete (name)  
C to delete (phone)  
D to see the list  
D  
List successful  
123-9999  
Powell  
[05/08/21]seed@VM:~/.../Lab$ ls -l NamePhone.java  
-rwsr-xr-x 1 root seed 7311 May  8 11:57 NamePhone.jav  
a  
[05/08/21]seed@VM:~/.../Lab$ █
```

SCREENSHOT OF MY FOLDER

