Implement a Phonebook Using a HashMap

Estimated time needed: 20 minutes

In this lab, you will learn how to handle HashMaps. You will learn to add items, search for items, and remove items.

You are currently viewing this lab in a Cloud-based Integrated Development Environment (Cloud IDE). It is a fully online, integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

Learning Objectives

After completing this lab, you will be able to:

- Understand HashMap as a collection
- Add new keys and respective values to a HashMap
- Search for existing values using the key
- Delete entries from HashMap
- View the entries in a sorted order of values

HashMap

A HashMap is like a dictionary or a lookup table. It stores data in pairs, where each pair consists of a key and a value. You can think of it as a real-life dictionary. The key is like the word you look up and the value is like the definition of that word.

In a HashMap, you use the key to quickly find the value associated with it. Each key must be unique (no duplicates), but values can be repeated.

To validate the phone numbers, there are many combinations of regex you will be able to find when you search. In this lab, you will use regex (Regular expression) patter matching.

```
\+?\d{1,4}?[-.\s]?\(?\d{1,3}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,9}
```

Regex explanation:

- `\+?` - Optional plus sign (+) for international numbers
- `\d{1,4}?` - Country code (1-4 digits)
- `[-.\s]?` - Optional separator (dash, dot, or whitespace)
- `\(?\d{1,3}??` - Optional area code (1-3 digits) enclosed in parentheses
- `[-.\s]?` - Optional separator
- `\d{1,4}[-.\s]?\d{1,9}` - Local phone number (1-4 digits, followed by an optional separator, and 1-9 digits)

As the regex has `\` which is an escape character, you will need to preceed it with another escape character.

```
"\\+?\\d{1,4}?[-.\\s]?\\(?\\d{1,3}?\\)?[-.\\s]?\\d{1,4}[-.\\s]?\\d{1,9}"
```

> An escape character is a special character that is used to change the meaning of the character that follows it. It is called an "escape" character because it allows you to "escape" the normal interpretation of the next character.

1. Create a project directory by running the following command.

   ```
   mkdir my_hashmap_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_hashmap_proj/src
   mkdir -p my_hashmap_proj/classes
   mkdir -p my_hashmap_proj/test
   cd my_hashmap_proj
   ```

3. Now create a file named `PhoneBookHashMap.java` inside the src directory.

```
touch /home/project/my_hashmap_proj/src/PhoneBookHashMap.java
```

4. Click the following button to open the file for editing.

Open **PhoneBookHashMap.java** in IDE

5. Copy and paste the code in `PhoneBookHashMap.java`. Read and the comments carefully to understand what the code does.

```
import java.util.HashMap;
import java.util.Scanner;
public class PhoneBookHashMap {
    // Method to validate if the name contains only letters, spaces, hyphens, or apostrophes
    private static boolean isNameValid(String name) {
        if (name.matches("^[a-zA-Z' -]+$") == false) {
            System.out.println("Invalid name!");
            return false;
        }
        return true;
    }
    // Method to validate if the phone number matches a specific format
    private static boolean isPhoneNumberValid(String phoneNumber) {
        if (phoneNumber.matches("\\+?\\d{1,4}?[-.\\s]?\\(?\\d{1,3}?\\)?[-.\\s]?\\d{1,4}[-.\\s]?\\d{1,9}") == false) {
            System.out.println("Invalid phone number!");
            return false;
        }
        return true;
    }
    public static void main(String s[]) {
        try {
            // Create a Scanner object to read user input
            Scanner scanner = new Scanner(System.in);
            // Create a HashMap to store names (keys) and phone numbers (values)
            HashMap<String, String> phonebook = new HashMap<>();
            // Infinite loop to keep the program running until the user chooses to exit
            while (true) {
                // Display the menu options to the user
                System.out.println(
                    "Press 1 to add an entry in the phonebook," +
                    "\n2 to view all the entries" +
                    "\n3 to search for entries with name" +
                    "\n4 to delete an entry" +
                    "\nAny other key to exit");
                // Read the user's choice
                String userAction = scanner.nextLine();
                // Option 1: Add a name-number entry to the phonebook
                if (userAction.equals("1")) {
                    // Prompt the user to enter the name
                    System.out.println("Enter a name");
                    String name = scanner.nextLine();
                    // Validate the name format
                    if (!isNameValid(name)) {
                        continue; // Skip to the next iteration if the name is invalid
                    }
                    // Check if the name already exists in the phonebook
                    if (phonebook.containsKey(name)) {
                        System.out.println("This name already exists! Do you want to replace the number? y/n");
```

```
                        String repChoice = scanner.nextLine();
                        // If the user chooses not to replace, skip to the next iteration
                        if (repChoice.equalsIgnoreCase("n")) {
                            continue;
                        }
                    }
                    // Prompt the user to enter the phone number
                    System.out.println("Enter the phone number");
                    String phoneNumber = scanner.nextLine();
                    // Validate the phone number format
                    if (!isPhoneNumberValid(phoneNumber)) {
                        continue; // Skip to the next iteration if the phone number is invalid
                    }
                    // Add the name and phone number to the HashMap
                    phonebook.put(name, phoneNumber);
                    System.out.println("The name and number have been added to the phonebook.");
                }
                // Option 2: View all the entries in the phonebook
                else if (userAction.equals("2")) {
                    // Iterate through the HashMap and print all key-value pairs
                    for (String name : phonebook.keySet()) {
                        System.out.println(name + ": " + phonebook.get(name));
                    }
                }
                // Option 3: Search for an entry by name
                else if (userAction.equals("3")) {
                    // Prompt the user to enter the name to search
                    System.out.println("Enter the name you want to search");
                    String keyName = scanner.nextLine();
                    // Check if the name exists in the phonebook
                    if (phonebook.containsKey(keyName)) {
                        // Display the phone number associated with the name
                        System.out.println("The phone number you are looking for is " +
                                phonebook.get(keyName));
                    } else {
                        System.out.println("No such name found in the phonebook.");
                    }
                }
                // Option 4: Delete an entry by name
                else if (userAction.equals("4")) {
                    // Prompt the user to enter the name to delete
                    System.out.println("Enter the name you want to delete ");
                    String keyName = scanner.nextLine();
                    // Check if the name exists in the phonebook
                    if (phonebook.containsKey(keyName)) {
                        // Remove the entry from the HashMap
                        phonebook.remove(keyName);
                        System.out.println("The entry has been removed.");
                    } else {
                        System.out.println("No such name found in the phonebook.");
                    }
                }
                // Exit the program if the user enters any other key
                else {
                    break;
                }
            }
        } catch (NumberFormatException nfe) {
            // Handle invalid number input (for example, non-integer input for priority or index)
            System.out.println("Invalid input. Please enter a valid number.");
        }
    }
}
```

The program is a phonebook management system implemented using a HashMap. `phoneBook` is HashMap that can hold entries, where, `Key` is the name of the person (unique) and `Value` is the phone number.
`isNameValid` checks if the name contains only letters, spaces, hyphens, or apostrophes. It is used to handle invalid name imput.
`isPhoneNumberValid` validates the phone number format using a regex pattern. It is used to handle invalid phone number imput.

6. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/PhoneBookHashMap.java
```

7. Set the `CLASSPATH` variable.

```
export CLASSPATH=$CLASSPATH:/home/project/my_hashmap_proj/classes
```

8. Run the program and test with variable combinations.

```
java PhoneBookHashMap
```

A sample output would look like

```
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
1
Enter a name
Lav
Enter the phone number
+1(201)920-2243
The name and number have been added to phonebook
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
1
Enter a name
Sam
Enter the phone number
+61498456533
The name and number have been added to phonebook
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
2
Lav: +1(201)920-2243
Sam: +61498456533
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
3
```

```
Enter the name you want to search
Sam
The phone number you are looking for is +61498456533
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
4
Enter the name you want to delete
Sam
The entry has been removed
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
2
Lav: +1(201)920-2243
Press 1 to add an entry in the phonebook,
2 to view all the entries
3 to search for entries with name
4 to delete an entry
Any other key to exit
```

Sort and Store

Now that you have added, updated, and deleted entries, in this section, you will sort the HashMap by name and store the content to a file.

1. Click the following button to open the file for editing, if it is not already open.

Open **PhoneBookHashMap.java** in IDE

2. Replace the existing code with the following code. This code has additional menu items to display names in sorted order using `TreeMap` and writing it on to a file using `java.io`.

   Sorting can implemented in more than one ways. It can be done using Comparator, as you did in the `ArrayListExample`. But you will use `TreeMap`, a class in the `java.util` package, which is a navigable map implementation that sorts its entries based on the natural ordering of its keys.

```
import java.util.HashMap;
import java.util.Scanner;
import java.util.TreeMap;
import java.util.Map;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
public class PhoneBookHashMap {
    // Method to validate if the name contains only letters, spaces, hyphens, or apostrophes
    private static boolean isNameValid(String name) {
        if (name.matches("^[a-zA-Z' -]+$") == false) {
            System.out.println("Invalid name!");
            return false;
        }
        return true;
    }
    // Method to validate if the phone number matches a specific format
    private static boolean isPhoneNumberValid(String phoneNumber) {
        if (phoneNumber.matches("\\+?\\d{1,4}?[-.\\s]?\\(?\\d{1,3}?\\)?[-.\\s]?\\d{1,4}[-.\\s]?\\d{1,9}") == false) {
            System.out.println("Invalid phone number!");
            return false;
        }
        return true;
    }
    public static void main(String s[]) {
        try {
            // Create a Scanner object to read user input
            Scanner scanner = new Scanner(System.in);
            // Create a HashMap to store names (keys) and phone numbers (values)
```

```
HashMap<String, String> phonebook = new HashMap<>();
// Infinite loop to keep the program running until the user chooses to exit
while (true) {
    // Display the menu options to the user
    System.out.println(
        "Press 1 to add an entry in the phonebook," +
        "\n2 to view all the entries" +
        "\n3 to search for entries with name" +
        "\n4 to delete an entry" +
        "\n5 to sort the entries by name" +
        "\n6 to write the entries onto a file" +
        "\nAny other key to exit");
    // Read the user's choice
    String userAction = scanner.nextLine();
    // Option 1: Add a name-number entry to the phonebook
    if (userAction.equals("1")) {
        // Prompt the user to enter the name
        System.out.println("Enter a name");
        String name = scanner.nextLine();
        // Validate the name format
        if (!isNameValid(name)) {
            continue; // Skip to the next iteration if the name is invalid
        }
        // Check if the name already exists in the phonebook
        if (phonebook.containsKey(name)) {
            System.out.println("This name already exists! Do you want to replace the number? y/n");
            String repChoice = scanner.nextLine();
            // If the user chooses not to replace, skip to the next iteration
            if (repChoice.equalsIgnoreCase("n")) {
                continue;
            }
        }
        // Prompt the user to enter the phone number
        System.out.println("Enter the phone number");
        String phoneNumber = scanner.nextLine();
        // Validate the phone number format
        if (!isPhoneNumberValid(phoneNumber)) {
            continue; // Skip to the next iteration if the phone number is invalid
        }
        // Add the name and phone number to the HashMap
        phonebook.put(name, phoneNumber);
        System.out.println("The name and number have been added to the phonebook.");
    }
    // Option 2: View all the entries in the phonebook
    else if (userAction.equals("2")) {
        // Iterate through the phonebook and print all name-number pairs
        for (String name : phonebook.keySet()) {
            System.out.println(name + ": " + phonebook.get(name));
        }
    }
    // Option 3: Search for an entry by name
    else if (userAction.equals("3")) {
        // Prompt the user to enter the name to search
        System.out.println("Enter the name you want to search");
        String keyName = scanner.nextLine();
        // Check if the name exists in the phonebook
        if (phonebook.containsKey(keyName)) {
            // Display the phone number associated with the name
            System.out.println("The phone number you are looking for is " +
                    phonebook.get(keyName));
        } else {
            System.out.println("No such name found in the phonebook.");
        }
    }
    // Option 4: Delete an entry by name
    else if (userAction.equals("4")) {
        // Prompt the user to enter the name to delete
        System.out.println("Enter the name you want to delete ");
        String keyName = scanner.nextLine();
        // Check if the name exists in the phonebook
        if (phonebook.containsKey(keyName)) {
            // Remove the entry from the HashMap
            phonebook.remove(keyName);
            System.out.println("The entry has been removed.");
        } else {
            System.out.println("No such name found in the phonebook.");
        }
    }
    // Option 5: Sort the entries by name
    else if (userAction.equals("5")) {
        // Sort the phoneBook by keys using TreeMap
        TreeMap phoneBookTreeMap = new TreeMap<String,String>(phonebook);
        for (Object keyName : phoneBookTreeMap.keySet()) {
            System.out.println(keyName + ": " + phoneBookTreeMap.get((String)keyName));
        }
    }
```

```
                // Option 6: Write the entries to a text file
                else if (userAction.equals("6")) {
                    // Write the Phonebook entries to a file
                    try (PrintWriter writer = new PrintWriter(new FileWriter("phonebook.txt"))) {
                        for (String name : phonebook.keySet()) {
                            writer.println(name + ": " + phonebook.get(name));
                        }
                        System.out.println("The entries are written to a file");
                    } catch (IOException e) {
                        System.err.println("Error writing to file: " + e.getMessage());
                    }
                }
                // Exit the program if the user enters any other key
                else {
                    break;
                }
            }
        } catch (NumberFormatException nfe) {
            // Handle invalid number input (for example, non-integer input for priority or index)
            System.out.println("Invalid input. Please enter a valid number.");
        }
    }
}
```

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/PhoneBookHashMap.java
```

4. Run the program and test with variable combinations.

```
java PhoneBookHashMap
```

The file will get written onto the same directory where you are running the code from. If you chose option `6` and wrote the phonebook onto a file, you should be able to see it when you click on the button below.

Open **phonebook.txt** in IDE

Practice Exercise

1. Create a HashMap which stores `Taskname` as key and the `Task` as value. The `Task` object should have the name, description, priority and status. You should be able to add, view, delete and update tasks.

    Hint: Trying using a combination of the labs you have worked on and Generative AI.

▶ Click here for sample code

Conclusion

In this lab, you learned how about HashMaps and how to manipulate them.

Author(s)

Lavanya