Exercise: Address Book

We want to write an address book and you are going to implement it in differnt ways:

Address book with if statements

- Use `Scanner` to take in a name
- Use `if` statements to print out
  - 0845 50 50 50 if the name is “Aaron”
  - 00 49 12345 if the name is “Fred”
  - 00 49 11111 if the name is “Jonas”
- Add another name and number to our program
- And then add another
- And then another

Main file:

```
package com.redi;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String name = scanner.next();

        if (name.equalsIgnoreCase("Aaron")) {
            System.out.println("0845 50 50 50");
        } else if (name.equalsIgnoreCase("Fred")) {
            System.out.println("00 49 5000");
        } else if (name.equalsIgnoreCase("Jonas")) {
            System.out.println("666");
        }

        // ADD MORE!

    }
}
```

Address book with normal arrays and normal loops

- Create a normal array that stores three names as `String`s
- Create a normal array that stores three numbers as `String`s
- Use `Scanner` to take in a name
- Use a `for` loop to loop over the names array
    - If the name in the array equals what you found from `Scanner` then print that out

Main file:

```
package com.redi;

import java.util.ArrayList;
import java.util.Map;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String name = scanner.next();

        String[] names = new String[3];
        String[] numbers = new String[3];
        names[0] = "Aaron";
        numbers[0] = "0845 50 50 50";
        names[1] = "Fred";
        numbers[1] = "00 49 12345";
        names[2] = "Jonas";
        numbers[2] = "666";
        for (int i = 0; i < names.length; i++) {
            if (names[i].equalsIgnoreCase(name)) {
               System.out.println("Number: " + numbers[i]);
            }
        }
    }
}
```

Address book with ArrayList

- Do as above, but with `ArrayList` instead of normal arrays

Main file:

```
package com.redi;

import java.util.ArrayList;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        String name = scanner.next();

        ArrayList<String> names = new ArrayList<>();
        ArrayList<String> numbers = new ArrayList<>();
        names.add("Aaron");
        numbers.add("0845 50 50 50");
        names.add("Fred");
        numbers.add("00 49 12345");
        names.add("Jonas");
        numbers.add("666");

        for (int i = 0; i < names.size(); i++) {
            if (names.get(i).equalsIgnoreCase(name)) {
                System.out.println("Number: " + numbers.get(i));
            }
        }
    }
}
```

Address book with TelephoneEntry object

- Replace the two arrays with one array
- Instead of using `String`s use an `TelephoneEntry` class.
    - This class should have a constructor that takes in a name and number.
- Ensure the program works as above.

Main file:

```
package com.redi;


import java.util.ArrayList;

import java.util.Scanner;


public class Main {


    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        String name = scanner.next();


        ArrayList<TelephoneEntry> phonebook = new ArrayList<>();


        phonebook.add(new TelephoneEntry("Aaron", "111"));

        phonebook.add(new TelephoneEntry("Fred", "222"));

        phonebook.add(new TelephoneEntry("Jonas", "666"));


        for (int i = 0; i < phonebook.size(); i++) {

            if (phonebook.get(i).getName().equals(name)) {

                System.out.println("Number: " + phonebook.get(i).getNumber());

            }

        }
    }
}
```

New file `TelefonyEntry.java`:

```
package com.redi;

public class TelephoneEntry {

    String name;
    String number;

    public TelephoneEntry(String name, String number) {
       this.name = name;
       this.number = number;
    }

    public String getNumber() {
        return number;
    }

    public String getName() {
        return name;
    }
}
```

Address book with for-each

Do as above, but with a for each loop instead of a normal for loop.

Main file:

```
for (TelephoneEntry entry : phonebook) {
    if (entry.name.equals(name)) {
        System.out.println(entry.number);
    }
}
```

Moving to Object Oriented Programming (Address book as a class)

Right now we have written all our code in the main class and the main method. That is fine for very small projects but in real programs we want to have as little logic as possible in the main class and method. Thefore we should write a class that abstracts the written logic for us.

Let’s identify what the program does so far and how it translates to methods:

- add a contact => `void addContact(TelephoneEntry)`
- get a number via a name => `String getNumber(String name)`

What attributes do we need:

- `ArrayList<TelephoneEntry> entries`

Should the scanner be a attribute too? No every class (and every method) should only deal with exactly one thing. Our class allready deals with storing data, therefore it should not care about where the data comes from.

Implement the class and write some code in the main method to check if your code is correct.

Address book with Maps instead of loops

To get a feeling of the speed of a method it is often a good idea to count how many statements are executed for a given input.

How many statements are executed in this code we have written:

```
String getNumber(String name) {
    for (TelephoneEntry entry : entries) {
        if (entry.name.equalsIgnoreCase(name)) {
            return entry.number;
        }
    }
}
```

If the name we are seaching for is at the end of the list we are have to search the complete list. Right now this is not so bad because we only have a few entries but if we want to do a phone book for all germans it might get slow.

An array or `ArrayList` maps from an integer to an value. It would be nicer if we could directly map from a string to a value. This mapping between two objects is called key-value-pair. In Java we can use a `Map` to express exactly that:

To create a new map that maps from `String` keys to `String` values:

```
Map<String, String> hm = new HashMap<>();
```

How would a map creation look like that maps from `Integer`s to `Boolean`s?

```
Map<Integer, Boolean> hm = new HashMap<>();
```

To add an entry:

```
map.put("Jonas","666");
```

To get an value by key:

```
String number = map.get("Jonas");
```

Rewrite the `AddessBook` class with a map instead of an array.

Made with ❤ by teachers at ReDI School.