Programming exercise:

Nicknames

In the `main`-method create a new `HashMap<String,String>` object. Store the names and nicknames of the following people in this hashmap so, that the name is the key and the nickname is the value. Use only lower case letters.

- matthew's nickname is matt
- michael's nickname is mix
- arthur's nickname is artie

Then get Matthew's nickname from the hashmap, and print it.

There is no automated tests for this exercise. Just submit the exercise when you think it works as it should.

Programming exercise:

Abbreviations

Create a class `Abbreviations` for managing common abbreviations. The class must have a constructor, which does not take any parameters. The class must also provide the following methods:

- `public void addAbbreviation(String abbreviation, String explanation)` adds a new abbreviation and its explanation.
- `public boolean hasAbbreviation(String abbreviation)` checks if an abbreviation has already been added; returns `true` if it has and `false` if it has not.
- `public String findExplanationFor(String abbreviation)` finds the explanation for an abbreviation; returns `null` if the abbreviation has not been added yet.

Example:

```
Abbreviations abbreviations = new Abbreviations();
abbreviations.addAbbreviation("e.g.", "for example");
abbreviations.addAbbreviation("etc.", "and so on");
abbreviations.addAbbreviation("i.e.", "more precisely");

String text = "e.g. i.e. etc. lol";

for (String part: text.split(" ")) {
    if(abbreviations.hasAbbreviation(part)) {
        part = abbreviations.findExplanationFor(part);
    }

    System.out.print(part);
    System.out.print(" ");
}

System.out.println();
```

Sample output

```
for example more precisely and so on lol
```

Programming exercise:

Print me my hash map

Exercise template contains a class `Program`. Implement the following class methods in the class:

- `public static void printKeys(HashMap<String,String> hashmap)`, prints all the keys in the hashmap given as a parameter.
- `public static void printKeysWhere(HashMap<String,String> hashmap, String text)` prints the keys in the hashmap given as a parameter, which contain the string given as a parameter.
- `public static void printValuesOfKeysWhere(HashMap<String,String> hashmap, String text)`, prints the values in the given hashmap whichs keys contain the given string.

Example of using the class methods:

```
HashMap<String, String> hashmap = new HashMap<>();
hashmap.put("f.e", "for example");
hashmap.put("etc.", "and so on");
hashmap.put("i.e", "more precisely");

printKeys(hashmap);
System.out.println("---");
printKeysWhere(hashmap, "i");
System.out.println("---");
printValuesOfKeysWhere(hashmap, ".e");
```

Sample output

```
f.e
etc.
i.e
---
i.e
---
for example
more precisely
```

NB! The order of the output can vary, because the implementation of hashmaps does not guarantee the order of the objects in it.

Programming exercise:

Print me another hash map

The exercise template contains the already familiar classes `Book` and `Program`. In the class `Program` implement the following class methods:

- `public static void printValues(HashMap<String,Book> hashmap)`, which prints all the values in the hashmap given as a parameter using

the toString method of the Book objects.

- `public static void printValueIfNameContains(HashMap<String,Book> hashmap, String text)`, which prints only the Books in the given hashmap which name contains the given string. You can find out the name of a Book with the method `getName`.

An example of using the class methods:

```
HashMap<String, Book> hashmap = new HashMap<>();
hashmap.put("sense", new Book("Sense and Sensibility", 1811, "..."));
hashmap.put("prejudice", new Book("Pride and prejudice", 1813, "...."));

printValues(hashmap);
System.out.println("---");
printValueIfNameContains(hashmap, "prejud");
```

Sample output

```
Name: Pride and prejudice (1813)
Contents: ...
Name: Sense and Sensibility (1811)
Contents: ...
---
Name: Pride and prejudice (1813)
Contents: ...
```

NB! The order of the output may vary. The implementation of a hashmap does not guarantee the order of the objects in it.

Programming exercise:

I owe you

Create a class called `IOU` which has the following methods:

- constructor `public IOU()` creates a new IOU
- `public void setSum(String toWhom, double amount)` saves the amount owed and the person owed to to the IOU.
- `public double howMuchDoIOweTo(String toWhom)` returns the amount owed to the person whose name is given as a parameter. If the person

cannot be found, it returns 0.

The class can be used like this:

```
IOU mattsIOU = new IOU();
mattsIOU.setSum("Arthur", 51.5);
mattsIOU.setSum("Michael", 30);

System.out.println(mattsIOU.howMuchDoIOweTo("Arthur"));
System.out.println(mattsIOU.howMuchDoIOweTo("Michael"));
```

The code above prints:

Sample output

```
51.5
30.0
```

Be careful in situations, when a person does not owe anything to anyone.

NB! The IOU does not care about old debt. When you set a new sum owed to a person when there is some money already owed to the same person, the old debt is forgotten.

```
IOU mattsIOU = new IOU();
mattsIOU.setSum("Arthur", 51.5);
mattsIOU.setSum("Arthur", 10.5);

System.out.println(mattsIOU.howMuchDoIOweTo("Arthur"));
```

Sample output

```
10.5
```