Collections (Part 2) tutorial

In this tutorial, you'll create and use a `Map` collection object from the Java Collections library.

A `Map` is a key-value pair data structure. `Map`s are also called **associative arrays** because they're collections of objects that have a relation between the key and the value stored in them.

Like `List`s, the data types for the keys and values in a `Map` have to be defined. The keys and values don't need to be the same data type.

To get started, import this project into IntelliJ. You'll write your code in the `src\main\java\com\techelevator\Tutorial.java` file.

In `Tutorial.java`, you'll see some comments where you can type your code for each step.

Step One: Declare a Map

Find the first comment in `Tutorial.java`. You'll add your code after this line:

```
// Step One: Declare a Map
```

You'll create a `Map` that has a `String` for the key and a `String` for the value. First, declare the variable that holds the `Map`:

```
Map<String, String> projects
```

Here, you specify two types in the `<>`. The first is the type of the key, and the second is the type of the value. You can mix these depending what you want to do with the structure.

To continue with the theme from yesterday, you'll add the names of some famous programmers as the key and their most famous project as the value, so you'll use two `String`s to represent them.

Next, you need to create a new `HashMap` and assign it to the variable:

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

Recall what the student book said about *programming to an interface* and `HashMap` fulfilling `Map`'s contract.

Step Two: Add items to a Map

Find the second comment in `Tutorial.java`. You'll add your code after this line:

```
// Step Two: Add items to a Map
```

To add an element to the end of the `Map`, use the `put()` method of the `Map` object. The `put()` method requires two arguments. The first is the key. The second is the value:

```
projects.put("Ada", "Inventing Computer Programming");
projects.put("Grace", "COBOL");
projects.put("Margaret", "Apollo Guidance Computer");
projects.put("Adele", "Graphical User Interfaces");
```

This links the four names, or keys, to the values that describe their projects.

Step Three: Loop through a Map

Find the third comment in `Tutorial.java`. You'll add your code after this line:

```
// Step Three: Loop through a Map
```

You can print out each of the key-value pairs by looping through all the elements in the `Map`.

Using a for-each loop, you can assign each element to a temporary `Map.Entry<String, String>` variable, and use the `getKey()` and `getValue()` methods to access the key and value:

```
for (Map.Entry<String, String> project : projects.entrySet()) {
    System.out.println("The key " + project.getKey() + " is linked to the value " 
+ project.getValue());
}
```

> Note: Your `Map.Entry` must have the same types as the `Map` that you're looping through.

If you run your code again, you'll see this output:

```
The key Ada is linked to the value Inventing Computer Programming
The key Grace is linked to the value COBOL
The key Margaret is linked to the value Apollo Guidance Computer
The key Adele is linked to the value Graphical User Interfaces
```

> Note: Since `Map` is an unordered data structure, the order of the items displayed may not be in the same order as the example.

Step Four: Remove an item from a Map

For this step, you'll add your code between Steps Two and Three so you can see the difference in the loop you wrote in Step Three.

For both `List`s and `Map`s, the `remove()` method removes an item from the collection. For `List`s, you pass the list item to the `remove()` method; for `Map`s, you pass the key to the method.

Add this line before your for-each loop from Step Three:

```
projects.remove("Grace");
```

Notice that when you added items to the `Map`, you had to provide both the key and value, but to remove an item, you only need the key.

If you run your code now, you'll notice that "Grace" and her project is no longer printed:

```
The key Ada is linked to the value Inventing Computer Programming
The key Margaret is linked to the value Apollo Guidance Computer
The key Adele is linked to the value Graphical User Interfaces
```

> Note: Since `Map` is an unordered data structure, the order of the items displayed may not be in the same order as the example.

Summary

After completing this tutorial, you should understand:

- The common operations of a `Map` and how to use them.
- How to use the for-each loop to iterate through a collection.