COURSEWARE

Professional Skills

Agile Fundamentals

Jira

Git

Databases Introduction

Java Beginner

Maven

Testing (Foundation)

Java Intermediate

- Optionals
- JDBC CRUD
- Exceptions
- SOLID Principles
- Single Responsibility
- Open/Closed
- Liskov Substituiton
- Interface Segregation
- Dependency Inversion
- Best Practice
- Design Patterns
- Creational Design Patterns
- Structural Design Patterns
- Behavioural Design Patterns
- Collection & Map
- HashSets
- HashMaps
- Enums
- Logging
- Generics
- Lambda Expressions
- Streams
- Complexity
- Input and Output
- Local Type Inference

HTML

CSS

HashMaps

Contents

- Overview
- HashMaps in action
- Tutorial
- Exercises
  - Morse Code Translator

Overview

HashMaps are a type of *collection* in Java.

You may have already encountered several different *collections* in Java, such as *Arrays*, *Lists*, and *ArrayLists* - all of which contain data which is accessed by an *index*.

HashMaps work differently because they allow you to *map* information *from an index* (key) *to the data you want* (value).

HashMaps in action

Let's look at a `HashMap` to see how they work in practice.

Here, we'll create a `HashMap` object which stores capital cities.

Remember to *code to an interface* - we'll use the interface `Map` to generate a new `HashMap` implementation:

▶ HashMaps

Just like other collection types, `HashMap` contains useful methods for adding and removing stuff from the collection.

Let's add some cities to our `HashMap` by using the `put()` method:

▶ Add to a HashMap

We can also see these by using `get()` - but for a `HashMap` we pass in a key rather than some pre-defined index:

▶ Get from a HashMap

Other methods, like `remove()`, `clear()` and `size()`, all work in the same way that they do for other collection types:

▶ HashMap methods

We can loop through a `HashMap` in 3 different ways:

- through the *keys* (on the left side) using `keySet()`
- through the *values* (on the right side) using `values()`
- through the *entire entry* (both sides) using `entrySet()`

▶ keySet

▶ values

▶ entrySet

Because keys and values inside a `HashMap` are both objects, we can define a `HashMap` for other object types too.

Let's define a `HashMap` object called `people` that maps names to ages:

▶ People

Tutorial

Javascript
Spring Boot
Selenium
Sonarqube
Advanced Testing (Theory)
Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

There is no tutorial for this module.

Exercises

Morse Code Translator

Consider the following class, which takes in a new `MorseTranslator` object and uses its `translate()` function on a String passed into it:

```
public class Runner {
    public static void main(String[] args) {
        MorseTranslator translator = new MorseTranslator();
        System.out.println(translator.translate(".--- .- ...- .- / .. ... / -.-. --- --- .-.. / --- -.-"));
    }
}
```

Write the `MorseTranslator` class, using a `HashMap`, to complete this exercise.

▶ Show solution