COURSEWARE

HashSets

Contents

Overview

HashSets 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*.

HashSets, like HashMaps, which we've covered here, work slightly differently to these.

A `HashSet` works by storing information through *hashing*.

Every object stored inside a `HashSet` has its own unique *hash code* associated with it.

(Because of this, duplicate values are not allowed within a `HashSet`.)

That hash code is then used as the index for the *value* of the data stored.

This means that every `HashSet` is backed by its own `HashMap`, where:

- the *hash code* is stored as the *key*
- the *actual object you're storing* in the `HashSet` is used as the *value*

Usage

You would likely use a `HashSet` instead of an `ArrayList` to store objects if you:

- don't have duplicate objects to store in the collection
- don't care about the order the objects are stored in the collection

HashSets in action

Let's take a look at how it works.

Instantiation

Instantiating a `HashSet` is similar to any other collection type. `HashSet` inherits from the `Set` interface, so we should code to the interface:

▶ HashSet instantiation

`add()`

Let's store some letters in our `HashSet` before printing it out.

Sets work slightly differently from *Maps* in that they don't allow duplicate values:

▶ HashSet add()

`size()` and `isEmpty()`

We can check whether a `HashSet` is empty or not by using `isEmpty()`:

▶ HashSet isEmpty()

We can also use `size()`, just like with other collection types:

▶ HashSet size()

`remove()`

`HashSet` comes with a handy utility method for removing *Entries*:

▶ HashSet remove()

`iterator()`

We could also use an `Iterator` to remove an object from the `HashSet` while iterating over it:

▶ HashSet iterator

`clear()`

Clearing the `HashSet` works in a similar way to other collection types, too:

▶ HashSet clear()

`contains()`

Finally, we can check whether an object exists within a `HashSet` by using `contains()`.

(`contains()` uses the `equals()` method to check if something exists within the `HashSet` - this is why it is important to generate `@Override` methods for the `equals()` and `hashCode()` methods when creating new object classes.)

▶ HashSet contains

Tutorial

There is no tutorial for this module.

Exercises

There are no exercises for this module.