Table of Contents

HashMap

A mutable hash map based on a Robin Hood hash table.

Usage

Create

You can create an empty map using new() or construct it using from_array().

```
1
2   test {
3     let _map2 : @hashmap HashMap[String, Int] = @hashmap new()
4
5   }
```

Set & Get

You can use set() to add a key-value pair to the map, and use get() to get a value.

```
1
2   test {
3     let map : @hashmap HashMap[String, Int] = @hashmap new()
4     map set("a", 1)
5     assert_eq(map get("a"), Some(1))
6     assert_eq(map get_or_default("a", 0), 1)
7     assert_eq(map get_or_default("b", 0), 0)
8     map remove("a")
9     assert_eq(map contains("a"), false)
10  }
```

Remove

You can use remove() to remove a key-value pair.

```
1
2   test {
3     let map = @hashmap of([("a", 1), ("b", 2), ("c", 3)])
4     map remove("a") |> ignore
5     assert_eq(map to_array(), [("c", 3), ("b", 2)])
6   }
```

Contains

You can use contains() to check whether a key exists.

```
1
2   test {
3     let map = @hashmap of([("a", 1), ("b", 2), ("c", 3)])
4     assert_eq(map contains("a"), true)
5     assert_eq(map contains("d"), false)
6   }
```

Size & Capacity

You can use size() to get the number of key-value pairs in the map, or capacit
y() to get the current capacity.

```
1
2   test {
3     let map = @hashmap of([("a", 1), ("b", 2), ("c", 3)])
4     assert_eq(map size(), 3)
5     assert_eq(map capacity(), 8)
6   }
```

Similarly, you can use is_empty() to check whether the map is empty.

```
1
2   test {
3     let map : @hashmap HashMap[String, Int] = @hashmap new()
4     assert_eq(map is_empty(), true)
5   }
```

Clear

You can use clear to remove all key-value pairs from the map, but the allocated
memory will not change.

```
1
2   test {
3     let map = @hashmap of([("a", 1), ("b", 2), ("c", 3)])
4     map clear()
5     assert_eq(map is_empty(), true)
6   }
```

Iteration

You can use each() or eachi() to iterate through all key-value pairs.

```
1
2   test {
3     let map = @hashmap of([("a", 1), ("b", 2), ("c", 3)])
4     let arr = []
5     map each((k, v) => arr push((k, v)))
6     let arr2 = []
7     map eachi((i, k, v) => arr2 push((i, k, v)))
8   }
```

Or use iter() to get an iterator of hashmap.

```
1
2   test {
3     let map = @hashmap of([("a", 1), ("b", 2), ("c", 3)])
4     let _iter = map iter()
5
6   }
```