# COM410 Programming in Practice

## B3.3 Hashing

## Hashing

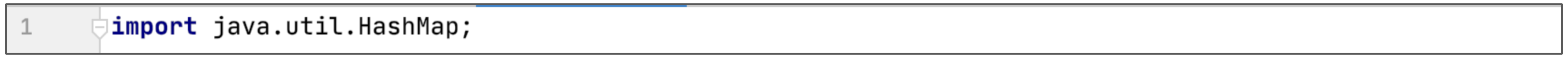
- The simplest possible search is when you don’t need to look for the item because you already know where it is located!
    - Each item in a collection is stored at a unique and known location, where the address of the location is a value that you specify
    - The unique address is known as the **hash**

## Hashing

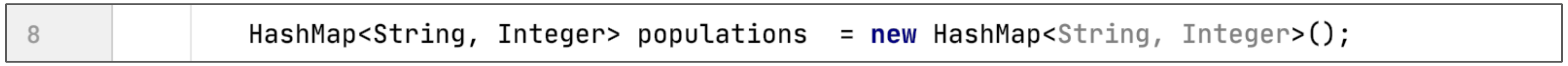
- Java provides a structure called a `HashMap` that allows us to specify the address of an item as a variable (must be an Object type)
    - For example, modules addressed by their modulecode (e.g. COM410), students addressed by their B-Number, mobile phones addressed by their unique number, etc.
    - No iteration or searching to locate an item – either it exists at the specified address or it doesn’t

## The Java HashMap

- Declare a `HashMap` with the object types of the key and the value



```java
import java.util.HashMap;
```

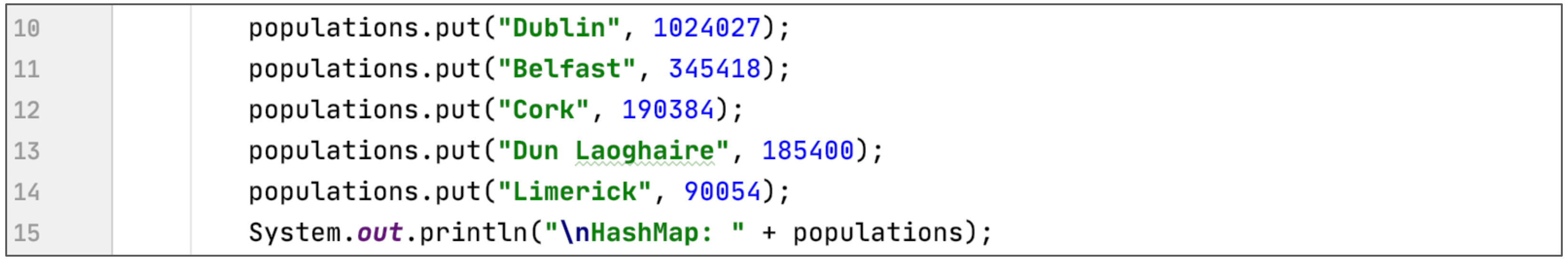


```java
        HashMap<String, Integer> populations  = new HashMap<String, Integer>();
```

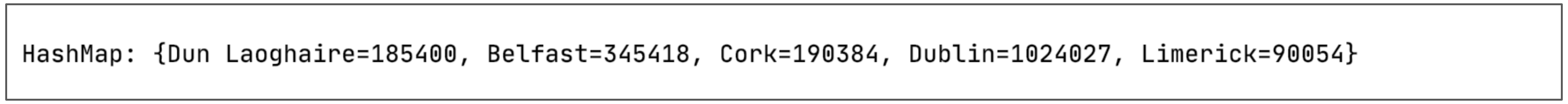
- Here, a `String` key and an `Integer` value

## The Java HashMap

- The `put()` method adds an element to the `HashMap`
    - Types of the parameters match those in the definition of the `HashMap`



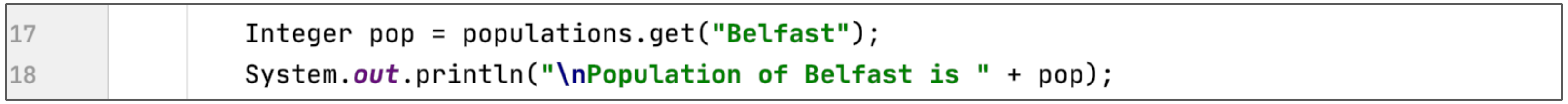
```java
        populations.put("Dublin", 1024027);
        populations.put("Belfast", 345418);
        populations.put("Cork", 190384);
        populations.put("Dun Laoghaire", 185400);
        populations.put("Limerick", 90054);
        System.out.println("\nHashMap: " + populations);
```



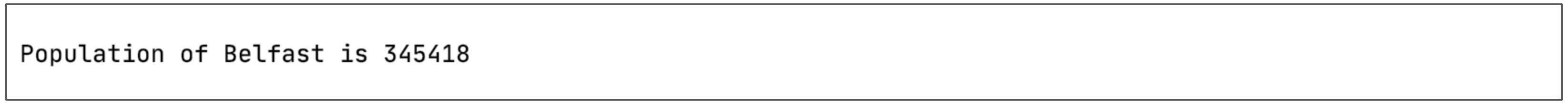
```
HashMap: {Dun Laoghaire=185400, Belfast=345418, Cork=190384, Dublin=1024027, Limerick=90054}
```

## The Java HashMap

- The `get()` method retrieves a value from the `HashMap` by addressing its key field



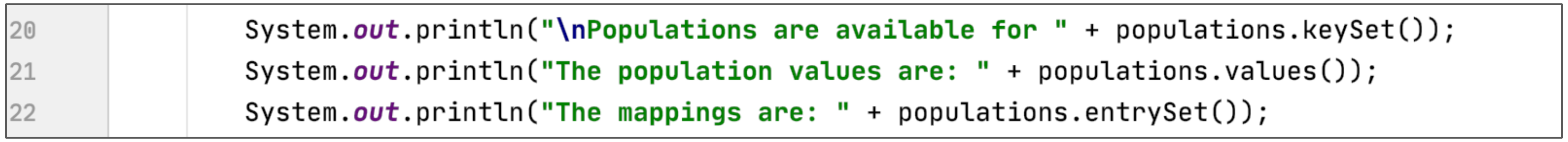
```java
        Integer pop = populations.get("Belfast");
        System.out.println("\nPopulation of Belfast is " + pop);
```



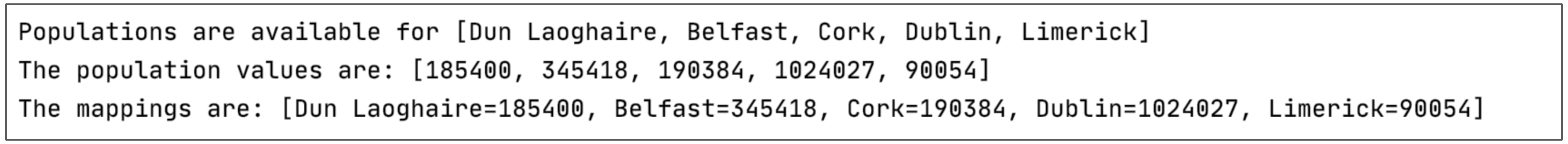
```
Population of Belfast is 345418
```

## The Java HashMap

- Retrieving data
  - The **keySet()** method retrieves the set of keys
  - The **values()** method retrieves the set of values
  - The **entrySet()** method retrieves the key=value pairs



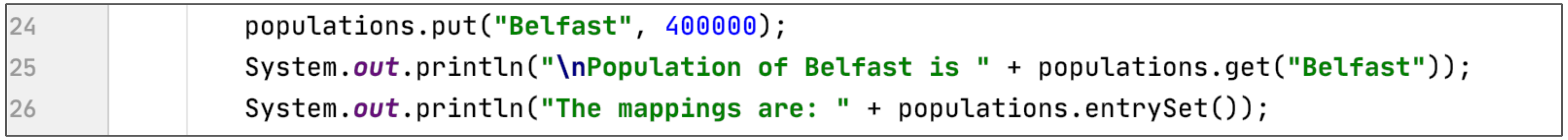
```java
System.out.println("\nPopulations are available for " + populations.keySet());
System.out.println("The population values are: " + populations.values());
System.out.println("The mappings are: " + populations.entrySet());
```



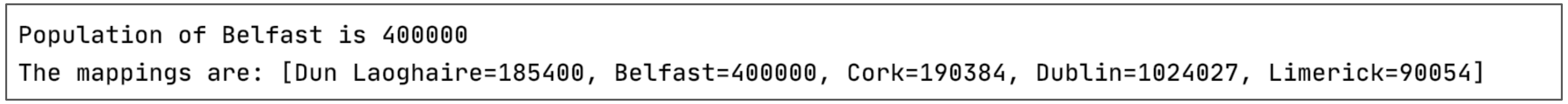
```
Populations are available for [Dun Laoghaire, Belfast, Cork, Dublin, Limerick]
The population values are: [185400, 345418, 190384, 1024027, 90054]
The mappings are: [Dun Laoghaire=185400, Belfast=345418, Cork=190384, Dublin=1024027, Limerick=90054]
```

# The Java HashMap

- One value per key
    - Values can be replaced by another `put()` operation



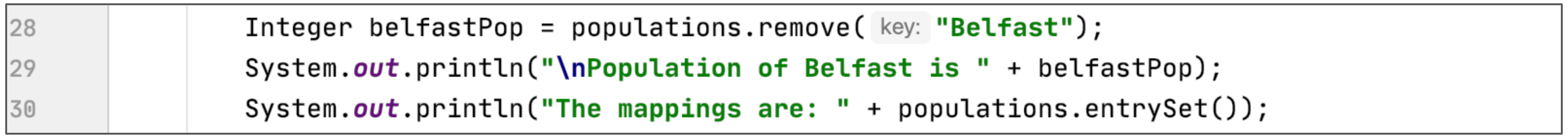
```java
        populations.put("Belfast", 400000);
        System.out.println("\nPopulation of Belfast is " + populations.get("Belfast"));
        System.out.println("The mappings are: " + populations.entrySet());
```



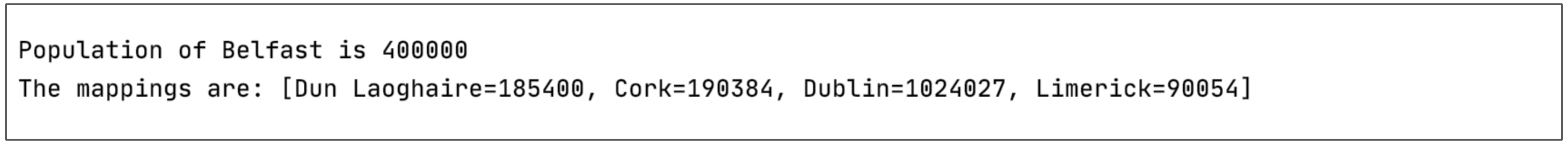
```
Population of Belfast is 400000
The mappings are: [Dun Laoghaire=185400, Belfast=400000, Cork=190384, Dublin=1024027, Limerick=90054]
```

# The Java HashMap

- Values can be deleted by the `remove()` method



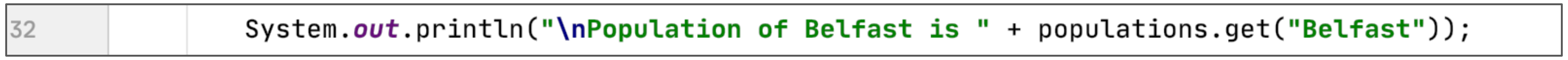
```java
        Integer belfastPop = populations.remove( key: "Belfast");
        System.out.println("\nPopulation of Belfast is " + belfastPop);
        System.out.println("The mappings are: " + populations.entrySet());
```



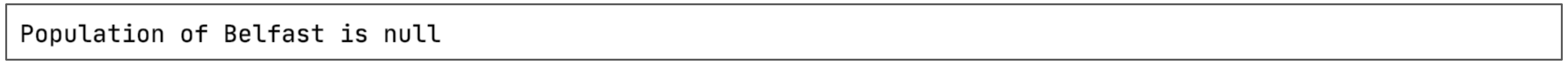
```
Population of Belfast is 400000
The mappings are: [Dun Laoghaire=185400, Cork=190384, Dublin=1024027, Limerick=90054]
```

# The Java HashMap

- Retrieving the value for a key that doesn't exist returns `null`



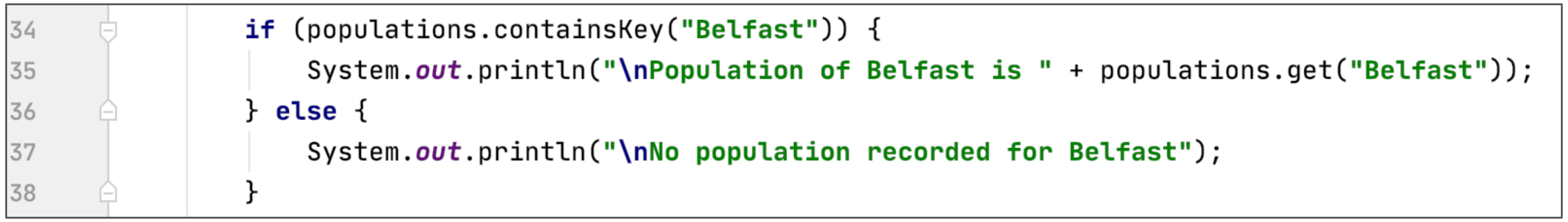
```
        System.out.println("\nPopulation of Belfast is " + populations.get("Belfast"));
```



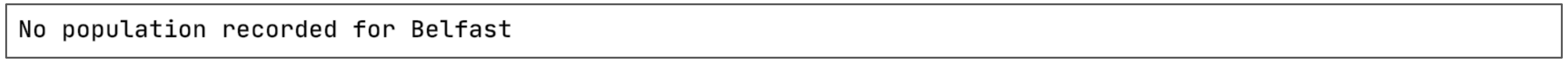
```
Population of Belfast is null
```

## The Java HashMap

- Use the `containsKey()` method to check that a key exists within the `HashMap`



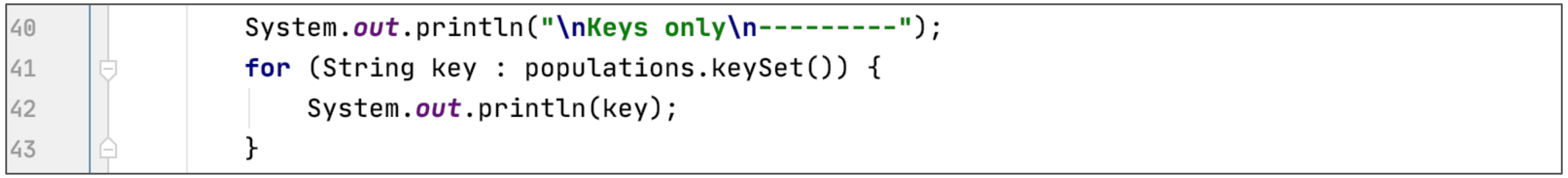
```
if (populations.containsKey("Belfast")) {
    System.out.println("\nPopulation of Belfast is " + populations.get("Belfast"));
} else {
    System.out.println("\nNo population recorded for Belfast");
}
```



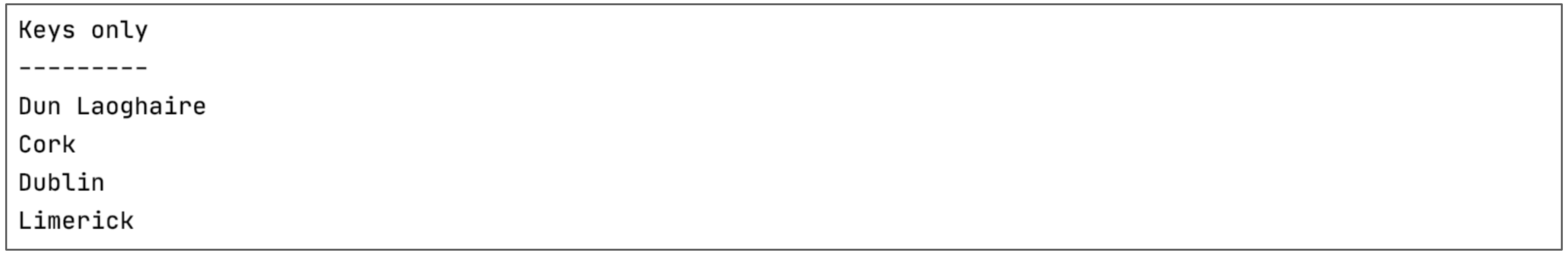
```
No population recorded for Belfast
```

## The Java HashMap

- Iterating over the `keySet()`



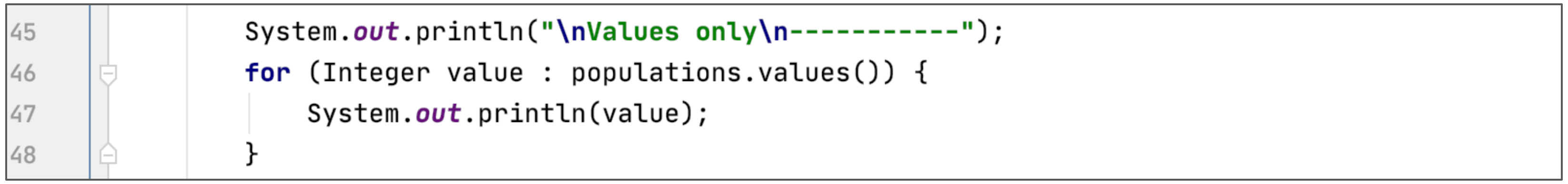
```java
System.out.println("\nKeys only\n---------");
for (String key : populations.keySet()) {
    System.out.println(key);
}
```



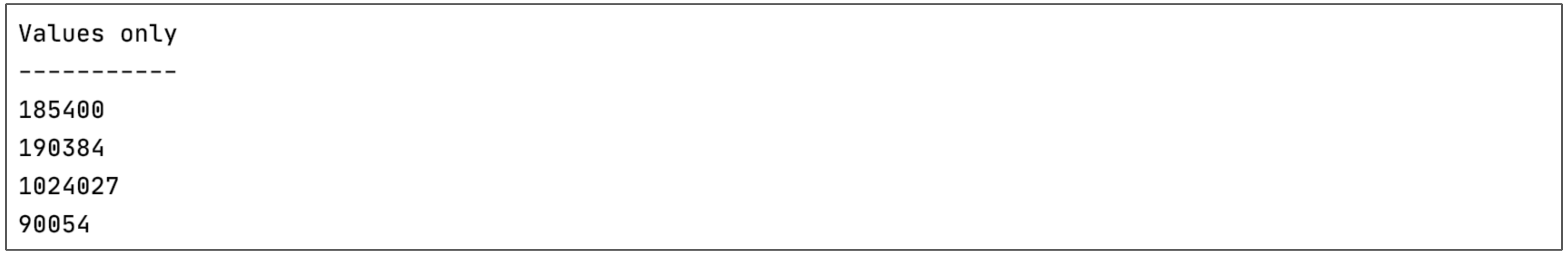
```
Keys only
---------
Dun Laoghaire
Cork
Dublin
Limerick
```

# The Java HashMap

- Iterating over the `values()`



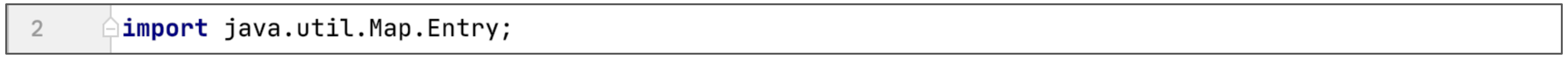
```java
System.out.println("\nValues only\n-----------");
for (Integer value : populations.values()) {
    System.out.println(value);
}
```



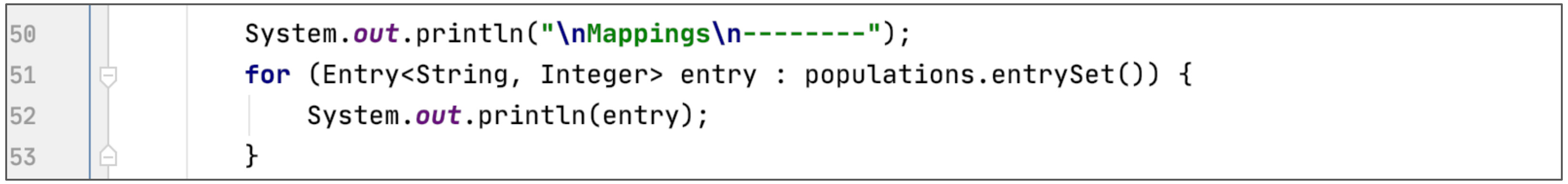
```
Values only
-----------
185400
190384
1024027
90054
```

## The Java HashMap

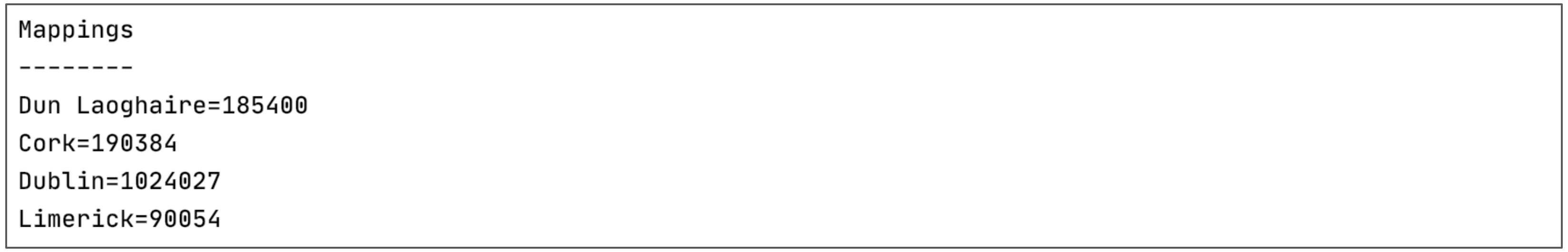
- Iterating over the `entrySet()`



```java
import java.util.Map.Entry;
```



```java
        System.out.println("\nMappings\n--------");
        for (Entry<String, Integer> entry : populations.entrySet()) {
            System.out.println(entry);
        }
```



```
Mappings
--------
Dun Laoghaire=185400
Cork=190384
Dublin=1024027
Limerick=90054
```

# Scenario

- Show how user-defined types can be stored in a `HashMap`
  - Add a `Person` class to your **Searching** project that defines a `Person` by a `String` name and an `int` age.
  - In a new class `Students`, create a `HashMap` where the key field is a `String` and the value field is a `Person`.
  - Add entries to the `HashMap` such that the key field is a student's B-Number and the value is a new `Person` object.
  - Demonstrate how you can retrieve details of each student by querying the `HashMap` on the student B-Number.